Conceptos y Métodos en
VISIÓN POR COMPUTADOR

Editado por Enrique Alegre,
Gonzalo Pajares y Arturo de la Escalera

Grupo de Visión del
Comité Español de Automática (CEA)

Conceptos y Métodos en Visión por Computador

Editado por Enrique Alegre, Gonzalo Pajares y Arturo de la Escalera

Junio 2016

Conceptos y métodos en Visión por Computador

Varios autores

Editado por: Alegre Gutiérrez, Enrique; Pajares Martinsanz, Gonzalo; de la Escalera Hueso, Arturo

ISBN: 978-84-608-8933-5

Materias: Ingeniería de instrumentos e instrumentación, Ingeniería de instrumentos e instrumentación, Ingeniería de control automático, Procesamiento de imágenes

España, Junio 2016

ÍNDICE

PRÓLOGO

Para los editores de este libro, *Conceptos y métodos en Visión por Computador*, ha sido un placer coordinar todas las tareas relacionadas con la publicación de esta obra, iniciativa del Grupo Temático Visión por Computador perteneciente al Comité Español de Automática (www.ceautomatica.es) y en la que han participado 39 profesores pertenecientes a 15 departamentos de 10 universidades españolas y una británica. Aunque cuando nos embarcamos en la tarea de sacar adelante la publicación de este libro no éramos conscientes de las dificultades que tendríamos que sortear, a la vista del resultado pensamos sinceramente que el tiempo, y la espera, han merecido la pena ya que la comunidad científica de habla castellana tendrá a partir de ahora una publicación de público acceso donde se detallan las ideas básicas donde se asientan los impresionantes desarrollos que se están observando en la actualidad en esta tecnología.

Ése es el objetivo principal que pretende este libro, familiarizar al lector con la Visión por Computador, el análisis de imágenes a través de computadores. Tradicionalmente, esta tecnología ha estado implantada en el ambiente industrial en dos campos principales:
1.- Lograr una mayor interactuación entre las máquinas y el entorno que las rodea.
2.- Conseguir un control de calidad total de los productos fabricados.

Sin embargo, la aparición de nuevos algoritmos y cámaras ha propiciado que el rango de aplicaciones se haya expandido en los últimos años. Un ejemplo de este tipo de aplicaciones son las cámaras y aplicaciones fotográficas, donde se detecta de forma automática las caras de las personas que aparecen en la imagen. Otros ejemplos clásicos son el reconocimiento automático de las matriculas de los vehículos o incluso las modernas cámaras 3D, capaces de dar información, no solo del color, sino también de la distancia a la que se encuentran los objetos que aparecen en la imagen. Estos avances posibilitan nuevos campos de aplicación como la seguridad, o la conducción automática, o nuevas formas de interactuar con los computadores, por ejemplo con los gestos de la mano, o movimientos de los ojos por no decir la creciente demanda proveniente de lo que se conoce como internet de las cosas (IoT, Internet of Things)

Sinceramente pensamos que los conocimientos transmitidos en los 19 capítulos de este libro permitirán a los lectores desarrollar sus propias aplicaciones de análisis de imágenes. Los tres primeros capítulos están centrados en los primeros bloques de una aplicación. Así se ven los diversos elementos que se pueden utilizar para capturar una imagen en una aplicación industrial: cámaras, lentes, iluminación, para pasar después a describir elementos importantes como el histograma de una imagen y sus diversas aplicaciones para mejorar el contraste o realce en ella. En tercer lugar se describen diversas formas de definir y tratar el color en las imágenes a través de los denominados espacios de color. El procesamiento de imágenes es la finalidad de los dos siguientes capítulos donde se presentan diversas transformaciones como las lógicas, geométricas o las convoluciones para detenerse con más detalles en el procesamiento morfológico, tanto en imágenes binarias como en niveles de gris. El capítulo sexto se centra en la segmentación, etapa en la que ya se empieza a extraer información de alto nivel de las imágenes. Para reconocer objetos se necesita poder describirlos, que es el objeto de dos temas, en el séptimo se describen la textura de la superficies de los objetos mientras que en el siguiente se describe en profundidad uno de los detectores y descriptores de características más importantes de los últimos

años, el SIFT o Scale Invariant Feature Transform. Los temas nueve y diez están dedicados a la clasificación y reconocimiento de patrones, bien usando técnicas clásicas como K-medias o el modelo *Bag of Visual Words*. Los cinco siguientes capítulos están relacionados con la adquisición, procesamiento y reconocimiento de objetos cuando se tiene información tridimensional (3D). Se empieza explicando los diversos modelos, paramétricos o no, de los que se dispone para representar la información 3D. Se explica después cómo se puede inferir la información 3D a partir de la que proporcionan las cámaras, así como los principios básicos de la visión estereoscópica, para pasar entonces al reconocimiento de objetos tridimensionales usando descriptores de superficie. Para terminar se explica con detalle el algoritmo RANSAC, muy utilizado cuando se quiere ajustar la información de los datos a alguno de los modelos explicados. Los siguientes dos capítulos, dieciséis y diecisiete, describen dos aplicaciones de la Visión por Computador. Primero se explica cómo mediante el control visual se usan las cámaras para poder guiar a los robots para llegar después a la introducción de la visión en los dispositivos móviles, uno de los campos de mayor crecimiento hoy en día, incluyendo el IoT como se ha mencionado previamente. Los dos últimos capítulos están dedicados a conceptos básicos, de índole matemática, existentes detrás de algunos de los aspectos explicados antes, como el uso del álgebra lineal y la geometría proyectiva.

Esperamos sinceramente que este libro sea de interés tanto al lector que ya tiene ciertos conocimientos sobre la Visión por Computador como al que se inicia en esta apasionante tecnología.

Enrique Alegre Gutiérrez
Gonzalo Pajares Martin-Sanz
Arturo de la Escalera Hueso

Junio de 2016

CAPÍTULO 1

SISTEMA DE CAPTURA DE IMÁGENES

Antonio J. SÁNCHEZ-SALMERÓN[1], Carlos RICOLFE-VIALA[1]

[1] Instituto de Automática e Informática Industrial, Universitat Politècnica de València, Valencia, España

La visión artificial consiste básicamente en la deducción automática de la estructura y propiedades de un mundo tridimensional, posiblemente dinámico, a partir de una o varias imágenes bidimensionales de ese mundo. En esta área de conocimiento se aúnan conceptos de la física del color, óptica, electrónica, geometría, algorítmica, sistemas de computación, etc.

El propósito de este capítulo es introducir los conceptos básicos sobre los dispositivos y tecnologías que componen un sistema de visión. Empezando por los diferentes tipos de fuentes de iluminación, describiendo brevemente los diversos dispositivos que permiten controlar la transmisión de la luz, además de los dispositivos de transducción y posteriormente de transmisión de información durante el proceso de formación y captura de imágenes.

1.1. Introducción

Una imagen es una representación bidimensional de una escena del mundo tridimensional. La imagen es el resultado de la adquisición de una señal proporcionada por un sensor, que convierte la información del espectro electromagnético en codificaciones numéricas. Y su transformación en el formato elegido de representación de imágenes, que constituye información discreta tanto en los valores que puede tomar, como en los parámetros que la definen. De forma genérica, una imagen digital se define como una matriz (o vector) de dimensiones N x M, conteniendo en cada elemento de la matriz un valor discreto que cuantifica el nivel de información del correspondiente elemento, representado con un número finito de bits (q). En este sentido, se puede formalizar el concepto de imagen como una función discreta de dos dimensiones de la siguiente forma:

$$I(x,y) \;/\; 0 <= x <= N-1, 0 <= y <= M-1 \qquad (1.1)$$

donde N y M pueden ser cualesquier valor numérico perteneciente al conjunto de números naturales y donde los valores de cada elemento han de ser múltiplos de 2:

$$0 <= I(x,y) <= p-1 \,/\, p = 2^q \qquad (1.2)$$

Los valores de cada elemento de la imagen representan desde niveles oscuros de luminosidad hasta valores claros. El nivel más oscuro es el valor más bajo del intervalo y viene representado por el negro y el nivel más claro se representa con el blanco y es el valor más alto.

Así, una imagen es una función bidimensional que proporciona cierta información electromagnética para cada uno de sus valores. A cada uno de estos elementos discretos se le denomina punto o píxel y generalmente contiene el nivel de iluminación o el color de un punto en la escena. El conjunto de puntos o pixeles forman la imagen o fotografía de la escena. Se dice que una imagen es fruto de la representación espectral recibida por el sensor. En este sentido por ejemplo el canal rojo en una imagen RGB correspondería a la representación espectral de las correspondientes longitudes de onda. El color se genera por la superposición de tres componentes espectrales.

El proceso de formación y captura de una imagen depende de un conjunto de elementos cada uno de los cuales cumple una misión determinada. Básicamente, el proceso consiste en generar una señal electromagnética en forma de luz que se transmite, refracta y refleja en la escena hasta que alcanza el área del sensor de la cámara. En el sensor se produce la transducción a una señal eléctrica y a partir de ese momento se dispone de un conjunto de elementos electrónicos que digitalizan, formatean, transmiten y almacenan las imágenes digitales. Dentro del primer conjunto de elementos se encuentran las fuentes de iluminación, el medio de la escena (generalmente aire), los objetos de la escena, los filtros o difusores y las ópticas. Los elementos electrónicos, que se encuentran a partir del sensor, son la electrónica de la cámara, los cables de transmisión para un determinado protocolo de comunicación y las tarjetas de adquisición conectadas al bus del procesador.

1.2. Iluminación

En el diseño de sistemas de visión industriales la iluminación no controlada del entorno no suele ser aceptable ya que se obtienen imágenes con bajo contraste, reflexiones especulares, sombras y destellos. Un sistema de iluminación bien diseñado proporciona luz a la escena de forma que la imagen que se obtiene favorezca el posterior proceso sobre la misma, manteniendo e incluso mejorando la información necesaria para la detección y extracción de los objetos y las características de interés.

1.2.1. Tipos de iluminación artificial

En lo referente a los tipos de iluminación que podemos utilizar para resolver una aplicación de visión artificial, se puede decir que dependiendo del fenómeno físico que produce la transformación de energía eléctrica en fotones existen diferentes tipos de iluminación con características propias. Las características principales que definen un tipo de iluminación son el rango de longitudes de onda en la que emite la luz, la durabilidad de la misma, sus variaciones a lo largo del tiempo y la temperatura que

pueda alcanzar. En la tabla 1.1 se clasifican los tipos de iluminación en función del fenómeno físico que produce la luz valorando las características de cada una de ellas.

Tabla 1.1. Características de cada uno de los tipos de iluminación artificial.

Tipo de iluminación	Ventajas	Inconvenientes
Incandescente/Halógena	• Bajo coste y fáciles de utilizar. • Permiten ajustar la intensidad de luz. • Oscila 50 veces por segundo.	• Desprenden una gran cantidad de calor • Su espectro se centra en el rojo siendo deficiente para azules verdes o amarillos
Fluorescentes	• Se calienta menos que el incandescente • Su espectro se centra en los colores del ojo humano • La duración está estimada en torno a 10.000 horas	• La longitud de onda de la luz cambia con el uso • Para que sean válidos en aplicaciones industriales tienen que trabajar a una frecuencia del orden de 25 KHz
Led	• Gran durabilidad (100.000 horas) • Posibilidad de encender y apagar solamente en el tiempo de captura de la imagen • Fácil elección de la longitud de onda de la fuente de luz dentro del espectro visible e infrarrojo • Las fuentes de luz se pueden construir en multitud de formas	• Precio
Láser	• Se utilizan para generar luz estructurada con forma diversas tales como líneas, líneas paralelas, líneas cruzadas, retículas, puntos y matriz de puntos. Para generar las formas se utilizan ópticas específicas. • Están disponibles en multitud de longitud de ondas desde el visible al infrarrojo cercano. • Dado que el ojo humano es muy sensible al verde, un diodo láser en esta longitud de onda genera un mejor contraste en los bordes especialmente sobre superficies rojas.	• Precio
Fibra óptica	• Se utiliza para llevar la luz a cualquier punto distante de la fuente de luz. • Permite iluminar pequeñas áreas. • Proporciona luz fría, es decir, no se calienta.	• Precio. • Sólo sirve para iluminar pequeñas áreas.

La figura 1.1 muestra cuatro ejemplos de distintos sistemas de iluminación.

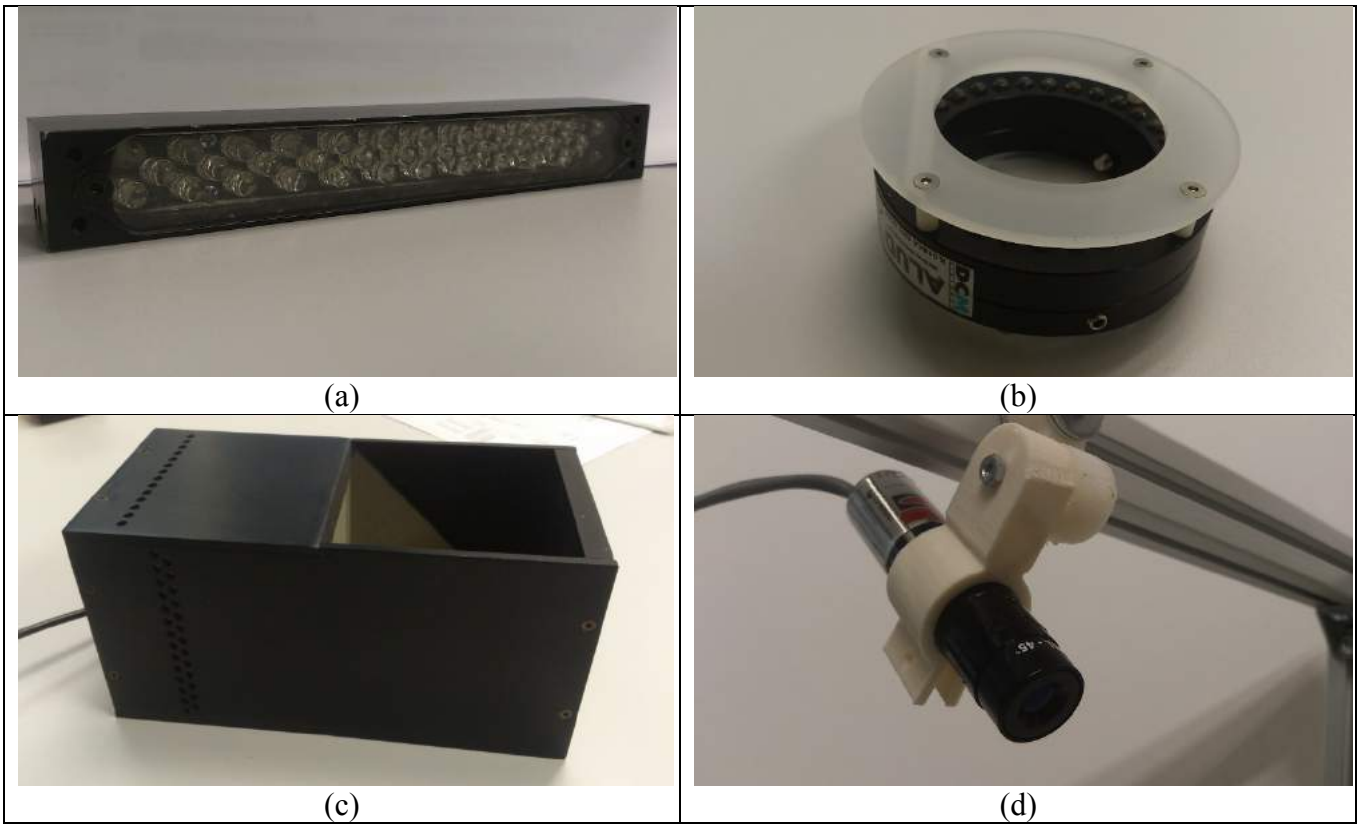


Figura 1.1. Distintos sistemas de iluminación. (a) Lineal con barra de leds. (b) Anillo de leds con filtro difusor. (c) DOAL. (d) Láser.

1.2.2. Técnicas de iluminación

El éxito de una aplicación de visión está altamente condicionado por la iluminación que se utilice. En el caso de no utilizar una iluminación adecuada, aparecen problemas de contrastes, brillos y sombras que complican el algoritmo de inspección o que incluso pueden llegar a que el algoritmo no pueda obtener una solución. Para mejorar notablemente la eficiencia del sistema de visión es necesario utilizar la técnica de iluminación adecuada que permita obtener una imagen correcta para ser procesada. Una imagen correcta para el procesado es aquella en la que los píxeles que representan los objetos de interés en la misma tienen características de luminosidad parecida y son muy distintas de los píxeles que no representan objetos de interés. Una iluminación apropiada es crítica para obtener una imagen correcta en la que no aparecen zonas saturadas o sombras que oculten información dentro de la imagen. Además las sombras causan falsas detecciones de bordes resultando en medidas incorrectas. También, una iluminación pobre puede resultar en una baja relación señal/ruido, lo que puede provocar una imagen con píxeles ruidosos. Igualmente, una iluminación no uniforme puede además dificultar notablemente operaciones de segmentación.

Para elegir una correcta iluminación también es necesario conocer el papel que juega cada componente del sistema de visión en la captura de la imagen. Cada componente influye en la cantidad de luz que llega al sensor y en consecuencia en la calidad de la imagen capturada. La apertura del

diafragma de la óptica afecta directamente a la cantidad de luz que llega al sensor. Si el diafragma se cierra debe ser aumentada la cantidad de luz que llega de la escena o aumentar el tiempo de exposición si se quiere conseguir una imagen con los mismos valores de luminosidad. También es importante el área que se está inspeccionando. Un área pequeña refleja menos luz que un área grande y en consecuencia esto tiene que tenerse en cuenta desde el punto de vista de la iluminación. La sensibilidad mínima de la cámara también es importante para determinar la cantidad mínima de luz que necesitamos para el sistema. Además el tiempo de exposición, la ganancia de la cámara también afectan directamente a la sensibilidad del sensor.

En la siguiente tabla se muestra a modo de resumen los pros y los contras de cada técnica de iluminación y las posibles aplicaciones en las que puede ser útil.

Tabla 1.2. Descripción de las diferentes técnicas de iluminación y sus posibles aplicaciones.

Técnica	Descripción	Pros	Contras
Direccional	La iluminación de la escena se realiza con uno o varios puntos de luz en la que el ángulo de incidencia no es paralelo ni perpendicular al eje de la cámara.	• Flexible • Adaptable • Barata	• Produce brillos • Genera sombras
Lateral o darkfield	Se utiliza luz direccional en la que el ángulo de incidencia es paralelo a la superficie a inspeccionar y perpendicular al eje de la cámara. Se puede utilizar en objetos opacos y transparentes.	• Resalta la textura de la superficie del objeto • Descubre grietas, burbujas incluso en el interior del objeto si es transparente	• Aparecen zonas quemadas y sombras • Poco contraste del borde
Difusa	Iluminando la escena de forma indirecta se consigue una luz suave.	• Reduce brillos • Reduce sombras • Iluminación suave	• Sistema de iluminación de gran tamaño • Dificultad para encajar en pequeños espacios • Las características de la superficie se difuminan
Anillo	Luz direccional con ángulo de incidencia en la misma dirección que el eje de la cámara. La lente se coloca en el centro del anillo.	• Reduce sombras • Se puede conseguir una iluminación suave si se utiliza un filtro en el anillo de luz.	• Reflejos con forma del anillo circular en la escena.
Difusa axial	Luz difusa alineada con la el eje óptico de la cámara. Se utiliza un cristal polarizado.	• No existen sombras. • Iluminación suave.	• Poca intensidad.
Estructurada	Se proyecta un patrón en la escena tipo línea, matriz de puntos o círculos generados por láser.	• Se obtiene la estructura del objeto	• No se distinguen colores
Contraluz	El objeto a inspeccionar se sitúa entre la fuente de luz y la cámara.	• Se obtiene una imagen del borde bien definida	• Elimina los detalles de la superficie

En las siguientes figuras se muestran diversos pares de imágenes (Infaimon, 2013) que permiten comparar los resultados de utilizar la técnica de iluminación directa frente a otras técnicas.

Figura 1.2. Imágenes capturadas con iluminación directa (columna de la izquierda) en comparación con imágenes capturadas con iluminación darkfield (columna de la derecha).

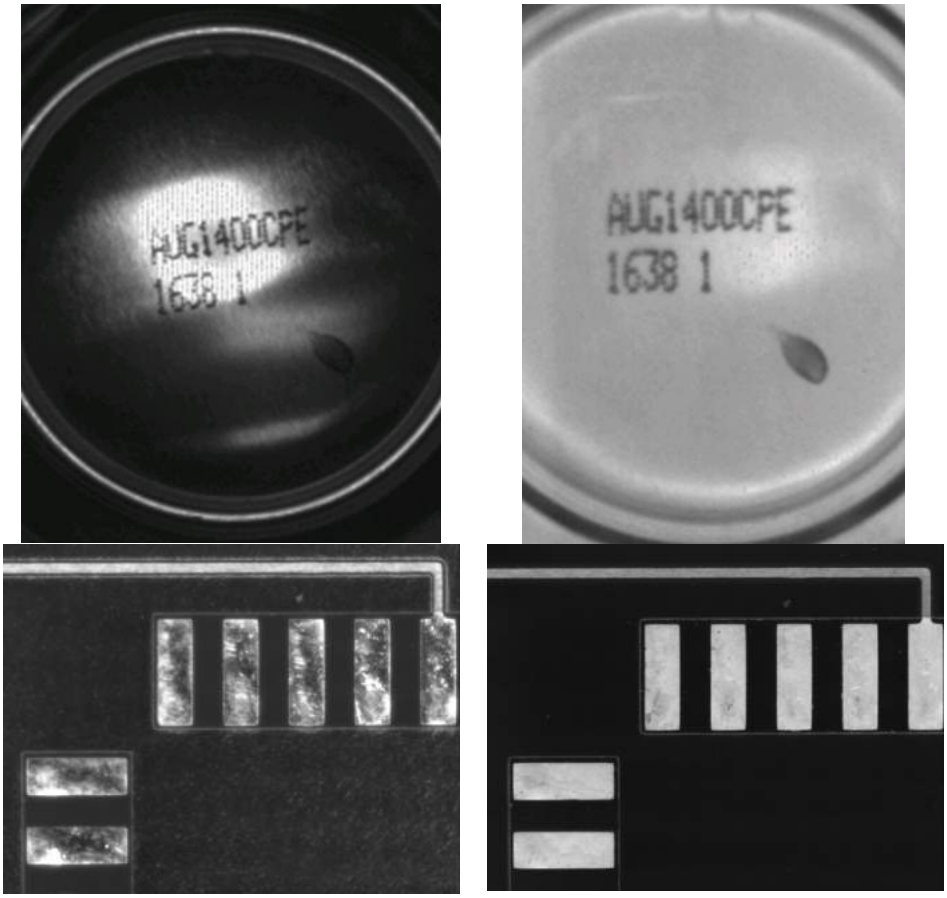


Figura 1.3. Imágenes capturadas con iluminación directa (columna de la izquierda) en comparación con imágenes capturadas con iluminación difusa (columna de la derecha).

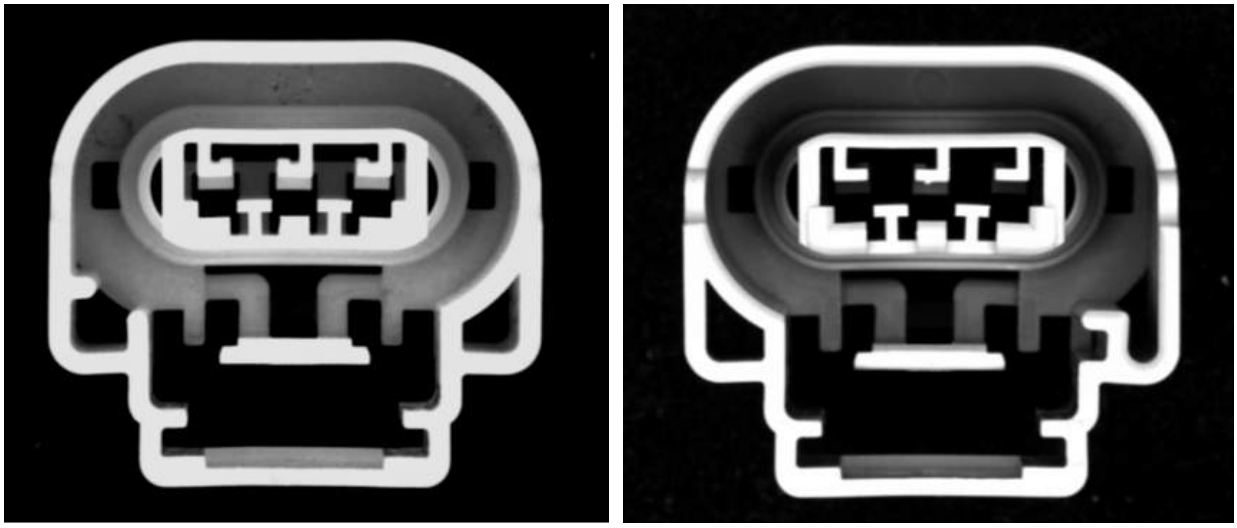


Figura 1.4. Imagen capturada con iluminación directa (izquierda) en comparación con una imagen capturada con iluminación difusa axial (derecha).

Figura 1.5. Imagen capturada con iluminación directa (izquierda) en comparación con una imagen capturada con iluminación estructurada (derecha).

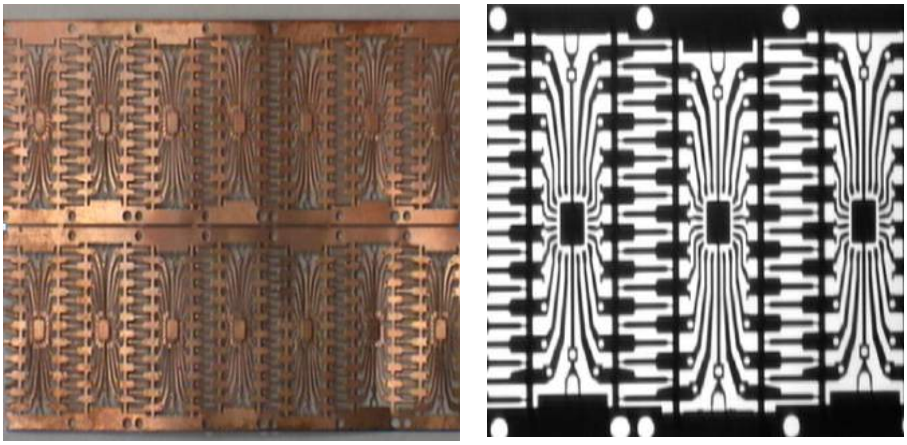


Figura 1.6. Imagen capturada con iluminación directa (izquierda) en comparación con una imagen capturada con iluminación a contraluz (derecha).

1.2.3. Filtros y difusores

Es interesante observar que si bien el ojo humano es sensible a todas las radiaciones luminosas del espectro visible, es sin embargo, incapaz de aislar una de ellas a partir de un flujo luminoso blanco. Para poder conseguir esto el ojo necesita un filtro coloreado que le impida ver el resto de los colores. Un filtro es un dispositivo fabricado con un determinado tipo de material en el que se producen los efectos de absorción y transmisión de las longitudes de onda espectrales. Por un lado transmite las longitudes de onda que equivalen al pigmento del mismo y por otro absorbe la parte del flujo luminoso restante.

Las características que diferencian los filtros entre sí son el color que transmiten; el material con el que se fabrican; la montura que incorporan y el factor o coeficiente de un filtro, que indica de manera aproximada el grado de opacidad o la capacidad de absorción del mismo al paso de un haz de luz blanca. Existen otras clases de filtros no coloreados y de los cuales cabe mencionar los filtros infrarrojos.

Los filtros pasabanda transmiten selectivamente una porción del espectro y rechazan el resto de longitudes de onda. Existen en multitud de anchos de banda.

Por otra parte los difusores reflejan o transmiten, la luz que les llega en una determinada dirección, en una gran variedad de direcciones diferentes minimizando de esta manera la generación de reflexiones especulares en la escena iluminada.

1.3. El sistema óptico

Los sistemas ópticos están formadas por lentes que modifican la dirección de los fotones cuando la luz atraviesa los mismos. Dependiendo del material del cristal que se utiliza para construir los sistemas ópticos, las características de la misma cambian.

1.3.1. Lentes

Una lente se caracteriza por un índice de refracción, una reflexión y una dispersión definida según el índice de Abbe. El índice de refracción n, describe la capacidad que tiene la lente para reducir la velocidad de la luz cuando pasa a través de ella. Se define como un cociente entre la velocidad de la luz en el vacío y la velocidad de la luz al cuando pasa a través de la lente.

$$n = velocidad\ de\ la\ luz\ en\ vacio\ /\ velocidad\ de\ la\ luz\ en\ el\ material\ =\ c/v \qquad (1.3)$$

siendo $c=2.998 \cdot 10^8\ m/s$.

Las lentes con un bajo índice de refracción tienen menos reflexión de forma natural. Para aumentar el índice de reflexión en las lentes con un alto índice de refracción, se utilizan recubrimientos que permiten reducir la reflexión. La dispersión describe la variación del índice de refracción con la

longitud de la onda de la luz que atraviesa la lente. Se mide utilizando el índice de Abbe representado en 1.4, y permite conocer la separación de colores que producirá la lente cuando la luz atraviese la misma. Un índice de Abbe pequeño indica una alta dispersión y una significativa separación de los colores, mientras que un índice de Abbe grande indica una baja dispersión y una menor separación de los colores.

$$vd = (nd - 1)/(nf - nc) \tag{1.4}$$

siendo:
nd=índice de refracción en 587.6nm (verde)
nf=índice de refracción en 486.1nm (azul)
nc=índice de refracción en 656.3nm (rojo)

Finalmente, la reflexión es el fenómeno que se produce cuando la luz cambia de medio y una parte de ella es devuelta al medio de procedencia. Por ejemplo, cuando la luz pasa del aire al cristal de la lente, se tiene entre un 8% y un 10% de pérdidas, las cuales dependen de la longitud de onda de la misma. Para reducir la reflexión, se añaden recubrimientos a la lente que permiten tener unas pérdidas de hasta el 0.25% o menos. Sin embargo, aunque el recubrimiento anti reflexión mejora significativamente el comportamiento de la lente, es necesario tener en cuenta, que esta mejora sólo se produce para las longitudes de onda para las cuales el recubrimiento anti reflexión ha sido diseñado. Fuera de estas longitudes de onda el comportamiento de la lente puede ser diferente al esperado. Este efecto se utiliza para construir lentes que filtren unas determinadas longitudes de onda.

1.3.2. Ópticas

La óptica se encuentra dispuesta en el objetivo, que es un conjunto complejo de lentes y mecánica dispuestas en un cilindro metálico y que dispone de una rosca o bayoneta que permite su incorporación al cuerpo de la cámara y su intercambio.

Las ópticas permiten enfocar luz procedente de un punto en un lado de la óptica en otro punto detrás de la misma. De esta forma, la óptica permite determinar la escena que se va a capturar en forma de imagen de un objeto situado a una distancia finita de la misma y la enfoca en el sensor que también está a una distancia fija al otro lado. Las ópticas pueden estar formadas por una única lente o varias lentes. La distancia focal *f* de las ópticas formadas por una única lente coincide con la distancia focal de la misma. En el caso de utilizar varias lentes el coste aumenta y la distancia focal de la óptica depende de las distancias focales de las lentes y de la distancia que separa las mismas. Las ópticas también se definen en base al ángulo de visión que abarcan. Generalmente se utiliza el ángulo de la diagonal, aunque se suele incluir el horizontal que resulta mucho más expresivo para el ser humano, acostumbrado a una visión de tipo panorámico en sentido horizontal. El campo de visión más amplio corresponde a los objetivos denominados gran angular y el más pequeño a los teleobjetivos. En consecuencia al cambiar la distancia focal, lo que se hace en la práctica es variar el tamaño en la imagen de un determinado objeto. Así, por ejemplo, si el objeto tiene un tamaño de 1 milímetro en la

imagen capturada con un objetivo de una focal de 25 mm, con un objetivo de una focal cuádruple (100 mm) tendrá un tamaño de 4 milímetros.

Los objetivos se pueden clasificar en objetivos normales, de focal larga y de focal corta, siendo la focal normal la que capta imágenes dentro de un ángulo similar al de la visión humana, con una perspectiva de los objetos equivalente a la que percibe el ser humano con sus ojos.

Existen diversos tipos de ópticas cada una de las cuales es útil para unas determinadas aplicaciones. Las ópticas con distancia focal fija tienen como característica un ángulo de campo de visión único definido por una distancia focal fija. Tienen una mínima distancia al objeto (MOD) a partir de la cual es posible enfocar el mismo. Los objetos que están más cercanos a la lente aparecen más grandes que los objetos que se sitúan más lejos. Se caracterizan porque son ligeras y pequeñas, por lo que son ampliamente utilizadas en aplicaciones de visión artificial.

Teniendo en cuenta la perspectiva, un objeto que está más cerca de la lente aparece más grande en la imagen que uno que está más lejos. Este hecho puede producir un error de perspectiva que es apreciable si la relación entre la distancia del objeto a la lente y la profundidad del mismo es del mismo orden. Para solucionar este error, en lugar de utilizar ópticas convencionales, se utilizan ópticas telecéntricas que evitan este error de perspectiva y las hacen ideales para aplicaciones de medición. Las ópticas telecéntricas ofrecen una muy buena calidad de imagen y el orden de amplificación de las mismas es de 0.06x hasta 6x.

Las ópticas con zoom o varifocales permiten cambiar la distancia focal o el ángulo del campo de visión de la lente. Los sistemas ópticos con zoom mantienen enfocado el objeto mientras la distancia focal se modifica. Por el contrario, con las ópticas varifocales, es necesario reenfocar de nuevo el objeto cada vez que se cambia la distancia focal. El cambio de distancia focal permite diferentes campos de visión para ser capturados sin necesidad de cambios en el resto del sistema de visión. Las ópticas con zoom son ideales para aplicaciones donde es necesario inspeccionar con un único sistema de visión objetos grandes teniendo en cuenta algunos pequeños detalles.

Las ópticas disponen de un diafragma que permiten regular la cantidad de luz que llega al sensor. La apertura del diafragma condiciona la profundidad de campo, la sensibilidad del sensor y la cantidad de luz necesaria en la escena para capturar una imagen de calidad. El número *f* máximo, que es la mayor apertura del diafragma en la escala F. Ésta es: 1, 1.4, 2, 2.8, 4, 5.6, 8, 11, 16, 22, 32 y cada una de las cifras de esta escala (de izquierda a derecha) indican que el objetivo deja pasar el doble de luz que la cifra anterior. En dicha escala el paso de una cifra a la siguiente, en cualquier sentido, se denomina un punto del diafragma o stop.

Los objetivos y las lentes para microscopio tienen un nivel de amplificación elevado ofreciendo una calidad de imagen superior. Estos objetivos pueden trabajar en un amplio rango de distancias al

objeto de trabajo. Además estos objetivos tienen la posibilidad de fácil integración con el sistema de iluminación o filtros específicos.

Como resumen de este punto, a la hora de seleccionar la óptica adecuada para una aplicación dada, cabe considerar:

a) Montura C y CS. Las cámaras con montura CS permiten acoplar tanto ópticas CS como C, pero en este último caso requieren de un anillo adaptador.

b) Distancia focal. Determina el ángulo de visión y por tanto el tamaño de cuadro de la escena que se proyectará como imagen. Valores pequeños equivalen a ópticas de gran angular (ángulos de visión grandes), mientras que los valores grandes equivalen a teleobjetivos (ángulos de visión pequeños). Se mide en mm.

c) Formato del sensor de la cámara. Se refiere al tamaño del sensor de la cámara, con lo cual para una misma distancia focal de óptica, a menor formato de cámara el ángulo de visión es más pequeño y viceversa: una óptica de 8 mm. proporciona mayor ángulo de visión en una cámara de 2/3" que en una de 1/3". Por tanto indica el ángulo de visión y los valores típicos son 2/3", 1/2" y 1/3".

d) Apertura. Determina la cantidad de luz que deja pasar la cámara y, por tanto, la sensibilidad. Se representa por un número (F) y que es el coeficiente entre la distancia focal *f* y el diámetro efectivo de la óptica. Números de F menores corresponden a ópticas que dejan pasar más luz.

Figura 1.7. Sistema de captura de imágenes formado por una cámara y una óptica

1.4. Cámaras

La cámara es el dispositivo que, utilizando un objetivo formado por un juego de lentes y el diafragma, construye una imagen sobre el plano del sensor compuesto de elementos fotosensibles, la digitaliza y la transmite hacia la tarjeta de adquisición del procesador. Están compuestas por un sensor y la electrónica asociada. Las cámaras proporcionan una señal de vídeo en un formato estándar para su digitalización (en el caso analógico) o directamente la información en formato digital que constituye la imagen captada por la misma (en el caso de cámaras digitales).

Montura entre la óptica y el cuerpo de cámara:
La montura de la cámara puede ser C, CS o F. La montura C es la más común la cual deja 17,5 mm de separación entre la óptica y el sensor. La montura CS es compatible con la C si se utiliza un separador de 5mm ya que la distancia entre el sensor y la óptica con esta montura es de 12,5 mm. La montura F se utiliza en cámara con sensores grandes. La distancia entre la óptica y el sensor con este tipo de monturas es de 46,5 mm la cual la hace ideales para cámaras con sensores lineales.

Formato del sensor:
CCD (Charge Coupled Device) y CMOS (Complementary Metal Oxide Semiconductor) son diferentes tecnologías utilizadas en la construcción del sensor para convertir la luz en señales eléctricas. Los sensores CMOS se utilizan en aplicaciones donde es necesario un bajo consumo de energía o en las que el espacio es reducido. El nivel de ruido en una imagen capturada con tecnología CMOS es mayor que una captura con tecnología CCD. Además, el rango dinámico del sensor es notablemente menor que el de un sensor de las mismas características desarrollado con tecnología CCD. Por el contrario, los sensores CMOS son más rápidos y tienen un menor consumo que los sensores desarrollados con tecnología CCD. Los sensores CCD se recomiendan en aplicaciones donde es necesaria una mayor calidad de imagen.

El elemento sensible de la inmensa mayoría de las cámaras actuales se compone de una matriz de sensores de acoplamiento de carga o CCDs. Cuando un CCD se ve expuesto a la luz, cada elemento del mismo absorbe una cierta cantidad de fotones, los cuales provocan la aparición de cargas eléctricas que se almacenan en un condensador MOS. Una propiedad importante de un sensor CCD es la posibilidad de transferir la carga en el momento que se decida, obteniéndose así un efecto de obturación electrónica, ya que sólo se convertirá en señal la luz durante el tiempo de exposición. Estas cargas se transmiten inmediatamente a través de la puerta (gate) a un registro de desplazamiento que se encargará de almacenarla y luego volcarla secuencialmente al exterior.

El sensor puede tener formato de área o lineal. Los sensores con formato de área son rectangulares y suelen tener un ratio típico 4:3(H:V). Los sensores de área son más grandes que los sensores lineales, ya que los sensores lineales capturan simplemente una línea de píxeles. El tamaño de la línea medida en píxeles es el que define la resolución del sensor. Los sensores de área son mucho más baratos que los sensores lineales ya que se utilizan en multitud de aplicaciones y son muy fáciles de configurar. Los sensores lineales se utilizan para capturar imágenes de objetos grandes ya que para capturar una imagen se van acumulando líneas sucesivas hasta que se forma la imagen. Para esto se necesita un movimiento relativo entre la cámara y el objeto de forma que el escaneo sea posible, lo que complica notablemente la captura de la imagen. Por el contrario, dado que sólo es necesario iluminar una línea de la escena, el apartado de iluminación es mucho más simple y las imágenes que se obtienen tienen una iluminación más homogénea.

Para obtener una imagen en color o monocroma existen diversas configuraciones. La tabla 1.3 muestra los pros y los contras de cada una de ellas.

Tabla 1.3. Técnicas para obtener una imagen monocromo o color y sus características.

MONOCROMO	COLOR CON UN SOLO SENSOR	COLOR CON 3 SENSORES
• Un solo sensor que recoge imágenes en escala de grises. • 10% más de resolución si se compara con una imagen en color capturada con un solo sensor. • Mejor relación señal/ruido. • Mayor contraste. • Mejor sensibilidad en condiciones de poca luz. • Ideal para aplicaciones de medida	• Usa un filtro de Bayer color RGB. • Más barato. • Menor resolución de imagen para un sensor con los mismos elementos sensibles (son necesarios más elementos sensibles para reconocer el color de un píxel). 	• Utiliza un prisma para separar la luz blanca en tres sensores. • Más caro. • Mejor resolución para una imagen en color. • Menor sensibilidad • Menos ópticas disponibles para este tipo de sensores.

El tamaño del sensor determina el campo de visión del sistema y el zoom necesario para poder capturar una imagen de la escena. El tamaño puede variar desde ¼ de pulgada (3,6x2,7 mm) hasta de 35 mm (36x24 mm) pasando por diferentes medidas tales como 1/2 pulgada (6,4x4,8 mm), 1 pulgada (12,8x9,6 mm), etc.

Para poder seleccionar la cámara y la óptica que compongan el sistema de visión que mejor se adecue a la aplicación que se desea resolver es necesario tener en cuenta los siguientes parámetros:

1. Campo de visión (Field Of View, FOV): área del objeto o escena del que se desea capturar una imagen.
2. Resolución: tamaño de la característica más pequeña del objeto que se desea que se vea en la imagen.
3. Distancia de trabajo: separación que existe entre el objeto que se captura la imagen y la óptica.
4. Profundidad de campo: máxima profundidad del objeto necesaria para conseguir un enfoque adecuado.

Como resumen de este punto, vamos a señalar los factores que son convenientes comprobar a la hora de seleccionar una determinada cámara:

a) Iris electrónico (Electronic Light Controller (ELC)). Esta función en una cámara, permite el ajuste a variaciones de luz ambientales, mediante el uso de un obturador electrónico automático. En la práctica actúa como un auto iris y permite usar ópticas de iris fijo o manual con el mismo efecto que si se usaran ópticas autoiris.

b) Auto Iris (AI) - AI vídeo y AI DC (DD). Las cámaras que incorporan esta función permiten trabajar con ópticas de iris automático. Se utilizan dos métodos de control: a) Autoiris vídeo, en el que la cámara envía a la óptica una señal de vídeo de referencia que se convierte en la tensión necesaria para mover el mecanismo del iris y b) Autoiris DC (o DD), que se diferencia del anterior en que se envía directamente la tensión correspondiente a la óptica.

c) Sincronización a línea (Line Lock). Si se tienen varias cámaras que obtienen al mismo tiempo varias imágenes de la escena desde diferentes puntos de vista, es usual que se produzcan saltos y oscilaciones (jitter) de la imagen. Para evitar este problema es necesario sincronizar las cámaras tomando como referencia una señal común. En las cámaras con sincronización de línea se toma como referencia la alimentación alterna (AC) de las cámaras.

d) Sincronización externa (Gen Lock). Este método de sincronización requiere una generación externa de la señal de sincronismo. Esta señal externa se distribuye a cada cámara a través de un cable coaxial independiente. La señal externa de referencia puede provenir de un generador de señal independiente o de la salida de vídeo de una cámara. Éste es el único método que proporciona una sincronización perfecta (vertical y horizontal).

e) Resolución. Es la capacidad de una cámara de producir una imagen detallada. La resolución está determinada por la composición espacial del número de líneas y columnas del sensor. Se expresa en números de elementos sensibles y dependerá, en definitiva, del número de píxeles del CCD.

f) Sensibilidad. Es el límite de trabajo de una cámara en condiciones de baja iluminación. Se expresa en Lux e indica el nivel de iluminación mínimo de la escena con el que puede trabajar una cámara. Es importante tener en cuenta que la medida en Lux se refiere siempre a una determinada apertura de óptica (F.1.2, F.1.4, etc.) y valores referidos, por ejemplo a F.1.4 serán menores que a F.1.2.

1.4.1. Estándares de vídeo

Las cámaras digitales disponen de muchas opciones de comunicación, cada una de las cuales se debe ajustar a las necesidades de la aplicación. Algunos formatos como por ejemplo Firewire permiten obtener una salida de vídeo y alimentación en un solo conector, lo que posibilita reducir notablemente los elementos que componen el sistema de visión. La velocidad de transferencia de datos afecta

directamente al número de capturas o *frames* por segundo que se pueden obtener de la cámara. Este dato también se ve condicionado por la resolución de la imagen y la profundidad de color (número de bits con el que se representa la información de cada píxel). La siguiente tabla muestra una comparativa de los diferentes tipos de interface de comunicaciones que existen en la actualidad.

Tabla 1.4. Comparación entre los diferentes tipos de interface de comunicaciones.

	FireWire 1394.a	**FireWire 1394.b**	**Camera Link**	**USB 2.0 (USB 3.0)**	**GigE(POE)**
Tasa Transferencia de datos	400Mb/s	800Mb/s	Hasta 3.6Gb/s	480Mb/s (5Gb/s)	1000Mb/s
Longitud máxima de cable	4,5 m	100 m	10m	5m	100m
Número de cámaras	Hasta 63	Hasta 63	1	Hasta 127	Ilimitada
Conector	6 pines-6 pines	9 pines-9 pines	26 pines	USB	RJ45/CAT5e ó 6
Tarjeta de captura	Opcional	Opcional	Necesaria	Opcional	No es necesaria
Alimentación	Opcional	Opcional	Necesaria	Opcional	Necesaria (Opcional)

1.4.2. Tarjetas de adquisición

Es la parte encargada de recibir la transmisión de la señal digital de vídeo que proporciona la cámara. En su funcionamiento cabe destacar como parámetros:

a) La resolución o definición espacial, esto es, el número de puntos discretos por imagen o cuadro (píxeles).
b) La resolución digital, que es el número de bits que se utilizan para codificar los niveles de intensidad luminosa de cada punto resultante de la digitalización.

Estas tarjetas (*frame grabbers*) agrupan los elementos de decodificación electrónica del protocolo de comunicación junto con una memoria local. Permiten básicamente recibir y almacenar imágenes, pudiéndose encontrar incluida en ellas varias funciones, como son:

a) Adquisición de secuencias de imágenes.
b) Aceleración del proceso de visualización en pantalla.
c) Procesado de imágenes.

1.5. Sistemas comerciales y aplicaciones

A la hora de seleccionar los componentes que integran un sistema de visión hay que tener en cuenta que existe una gran cantidad de fabricantes de sistemas de iluminación, filtros, cámaras, ópticas, tarjetas de adquisición, procesadores y software en el mercado.

Los componentes *hardware* más importantes de los sistemas de visión se pueden conseguir a través de una gran cantidad de fabricantes o distribuidores. A continuación se enumeran algunos de los fabricantes y distribuidores más importantes a nivel mundial:

- Cognex (Cognex, 2015): multinacional líder a nivel mundial en ventas de sistemas de visión para la industria.
- Festo (Festo, 2015) multinacional multisectorial que dispone de una serie de sistemas de visión en su catálogo de productos.
- Omron (Omron, 2015): multinacional multisectorial que disponen de diversos sistemas de visión.
- Infaimon (Infaimon, 2015): distribuidor especialista en todos los componentes de un sistema de visión en el ámbito de iberoamérica.
- Teledyne Dalsa (Teledyne, 2015): fabricante a nivel mundial de componentes de visión.
- Alava Ingenieros (Alava Ingenieros, 2015) ingeniería multinacional multisectorial que desarrollan y venden sistemas de visión. Dispone de una amplia gama de tecnologías de visión e imágenes.
- DCM Sistemes (DCM Sistemes, 2015): fabricante español de sistemas de iluminación.
- Bitmakers (Bitmakers, 2015): distribuidor de sistemas de visión. Importador exclusivo en España de Keyence.
- Keyence (Keyence, 2015): fabricante de sistemas de visión.
- Point Grey (Point Grey, 2015): fabricante de sistemas de visión.
- Etc.

Por otra parte también existe una gran cantidad de librerías y plataformas software de tratamiento de imágenes. Entre todas las existentes se enumera a modo de ejemplo las siguientes:

- OpenCV (Opencv, 2015): amplia librería de código abierto de procesamiento de imágenes. Esta librería planteada para programar en C++ o Python.
- Halcon (Halcon, 2015): Librería propietaria de procesamiento de imágenes.
- Matrox MIL Library (Matrox, 2015) Librería propietaria de procesamiento de imágenes.
- National Instruments (National Instruments, 2015): La plataforma de programación Labview dispone de adquisición y procesamiento de imágenes con un entorno de programación gráfico.
- MathWorks (Mathworks, 2015): La plataforma de programación Matlab dispone de paquetes de adquisición y procesamiento de imágenes que son interesantes para realizar pruebas de análisis de viabilidad de un sistema de visión.
- Etc.

A continuación se analiza qué sistema de iluminación se podría utilizar para facilitar la solución de cuatro ejemplos de aplicaciones de visión.

1.- Un fabricante de teclados de PC necesita asegurarse de que ninguna de las teclas de la calculadora se ha quedado atascada por debajo de la posición esperada antes de la fase de empaquetado. ¿Qué tipo de iluminación se puede utilizar para resolver este problema? En esto caso sería interesante utilizar un sistema de luz estructurada ya que la deformación del patrón proyectado depende de la altura de las teclas.

2.- Un fabricante de tarjetas de circuitos impresos necesita determinar la posición de los agujeros perforados en la etapa de taladrado para insertar los componentes de la placa automáticamente. ¿Qué tipo de iluminación optimiza la solución de este problema? Aprovechando que las tarjetas de circuitos impresos son opacas, sería interesante utilizar un sistema de iluminación a contraluz. De este modo se capturarán imágenes de los agujeros muy contratadas ya que estos no impedirán que la luz llegue directamente a la cámara.

3.- Un fabricante de botellas de cristal desea detectar posibles muescas en las bocas de las botellas. ¿Qué tipo de iluminación será mejor para eliminar los reflejos especulares que se producen sobre el cristal? En este caso puede ser interesante utilizar un sistema de iluminación lateral.

4.- Un fabricante de champú desea comprobar la calidad de la etiqueta impresa en blanco sobre una botella opaca de color verde. ¿Qué tipo de iluminación utilizarías para obtener una imagen más contrastada? En este caso podría ser interesante utilizar una iluminación que no emitiera las longitudes de onda de color verde (por ejemplo una luz roja o azul) de esta manera la imagen de la botella tendrá un color más oscuro que el texto de la etiqueta. Por el contrario una iluminación de color verde nos dará una imagen poco contratada.

En Internet se puede encontrar una gran cantidad de ejemplos de aplicaciones de visión, por ejemplo en el siguiente enlace http://www.infaimon.com/es/menu/aplicaciones se dispone de una gran cantidad de ejemplos por sectores.

1.6. Bibliografía

Alava Ingenieros (2015). Disponible online: http://www.alava-ing.es/ (accedido 8 Octubre 2015).
Bitmakers (2015). Disponible online: http://www.bitmakers.com (accedido 8 Octubre 2015).
Burke, M.W. (1996). Handbook of Machine Vision Engineering, Vol. I: Image Acquisition, Chapman & Hall, 1996.
Cognex (2015). Disponible online: http://www.cognex.com (accedido 8 Octubre 2015).
DCM Sistemes (2015). Disponible online: http://www.dcmsistemes.com/ (accedido 8 Octubre 2015).
Festo (2015). Disponible online: http://www.festo.com (accedido 15 Mayo 2015).

Halcon (2015). Disponible online: http://www.halcon.com/ (accedido 8 Octubre 2015).
Infaimon. (2013). Catálogo de productos. Disponible online:
http://www.infaimon.com/es/menu/publicaciones (accedido 8 Octubre 2015).
Infaimon (2015). Disponible online: http://www.infaimon.com/ (accedido 8 Octubre 2015).
Keyence (2015). Disponible online: http://www.keyence.com/ (accedido 8 Octubre 2015).
MathWorks (2015). Disponible online: http://es.mathworks.com/ (accedido 8 Octubre 2015).
Matrox (2015). Disponible online: http://www.matrox.com/ (accedido 8 Octubre 2015).
National Instruments (2015). Disponible online: http://spain.ni.com/ (accedido 8 Octubre 2015).
Omron (2015). Disponible online: http://www.omron.com/ (accedido 8 Octubre 2015).
OpenCV (2015). Disponible online: http://opencv.org/ (accedido 8 Octubre 2015).
Point Grey (2015). Disponible online: http://www.ptgrey.com/ (accedido 8 Octubre 2015).
Teledyne Dalsa (2015). Disponible online: http://www.teledynedalsa.com (accedido 8 Octubre 2015).

CAPÍTULO 2

OPERACIONES SOBRE EL HISTOGRAMA Y FILTRADO DE LA IMAGEN

José L. ESPINOSA-ARANDA, Milagro FERNÁNDEZ-CARROBLES, Noelia VÁLLEZ

Grupo VISILAB, ETSI Industriales, Universidad de Castilla-La Mancha Ciudad Real, España

Las principales técnicas de procesamiento digital de imágenes mejoran y realzan sus características. Las operaciones que son llevadas a cabo por este tipo de técnicas implican la alteración del histograma de la imagen o el procesamiento del valor de sus píxeles. En este capítulo se describen las principales operaciones sobre el histograma junto con los principales métodos de filtrado frecuencial y espacial.

2.1. El histograma de la imagen

Dada una imagen cuyos píxeles se encuentren definidos en escala de grises con valores entre 0 y 255, se llama histograma de la imagen al gráfico que se obtiene de la representación de la frecuencia de aparición de cada uno de los valores (Fig. 2.1). Recordando que el 0 representa el negro y el 255, en este caso, representa el blanco, el histograma muestra desde los píxeles oscuros a los claros.

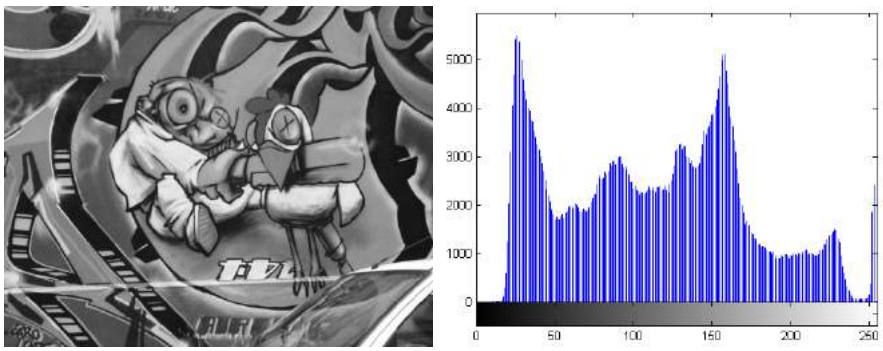


Figura 2.1. Imagen junto a su correspondiente histograma.

Si cada frecuencia es dividida entre el número total de píxeles de la imagen se obtiene el histograma normalizado. En ese caso cada valor obtenido representa la probabilidad de obtener un píxel con el valor de intensidad al que está asociada la frecuencia, (Freeman, 2005).

2.2. Umbralización del histograma

El proceso de umbralizado obtiene, generalmente, una imagen binarizada a partir de la imagen original (Bueno, 2015). Sean I e I' la imagen original y su imagen umbralizada, U el valor de intensidad establecido como umbral y L-1 el mayor nivel de intensidad posible, cada nuevo píxel I'_{ij} se obtiene mediante:

$$I'(\mathrm{i},\mathrm{j}) = \begin{cases} 0, & I(\mathrm{i},\mathrm{j}) < U \\ L-1, & I(\mathrm{i},\mathrm{j}) \geq U \end{cases} \tag{2.1}$$

El ejemplo de la Figura 2.2 muestra la imagen de una moneda en la que, al aplicar un umbralizado binario con U=60, es posible separar la moneda del fondo.

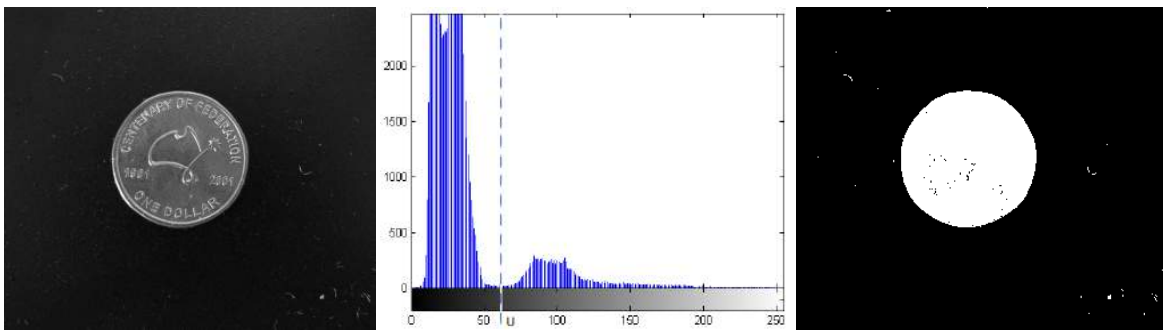


Figura 2.2. Imagen junto al resultado de su umbralización con U=60.

Existen otros métodos de umbralización que sólo alteran parte de los valores de la imagen. Un ejemplo de ellos es la umbralización por truncamiento que establece el valor umbral para todos aquellos píxeles que tengan un valor mayor a él.

Desafortunadamente, la elección del valor umbral no es trivial. Algunos métodos de umbralizado seleccionan este valor en base a una serie de características de la imagen o al resultado que se desea obtener. Uno de los métodos más utilizados es el llamado *método de Otsu*.

En 1979 Otsu propuso utilizar la varianza para calcular de forma automática el valor umbral (Fig. 2.3). En concreto se calculan las varianzas de las dos partes que resultan de dividir el histograma en dos. Como la varianza indica la dispersión de los valores, el valor umbral seleccionado será aquel que haga que la varianza dentro de cada parte sea mínima. Los pasos para calcular el umbral de Otsu son, (Otsu, 1979)):

a) Calcular el histograma normalizado H(x).

b) Establecer la probabilidad de pertenecer a la primera y a la segunda parte del histograma, $w_1(k)$ y $w_2(k)$ respectivamente, para cada valor k desde 1 hasta L-1.

$$w_1(k) = \sum_{i=0}^{k} H(k) \quad \text{y} \quad w_2(k) = \sum_{i=k+1}^{L-1} H(k) = 1 - w_1(k) \tag{2.2}$$

c) Calcular las medias $\mu_1(k)$ y $\mu_2(k)$ de cada parte del histograma:

$$\mu_1(k) = \sum_{i=0}^{k} i \cdot \frac{H(k)}{w_1(k)} \quad \text{y} \quad \mu_2(k) = \sum_{i=k+1}^{L-1} i \cdot \frac{H(k)}{w_2(k)}\,. \tag{2.3}$$

d) Calcular las varianzas $\sigma_1^2(k)$ y $\sigma_2^2(k)$ dentro de cada parte:

$$\sigma_1^2(k) = \sum_{i=0}^{k}\big(i - \mu_1(k)\big)\frac{H(k)}{w_1(k)} \quad \text{y} \quad \sigma_2^2(k) = \sum_{i=k+1}^{L-1}\big(i - \mu_2(k)\big)\frac{H(k)}{w_2(k)} \ . \tag{2.4}$$

e) Por último, el umbral U es seleccionado como:

$$U = \arg min_k\, \sigma^2(k) = \arg min_k\, w_1(k)\sigma_1^2(k) + w_2(k)\sigma_2^2(k) \ . \tag{2.5}$$

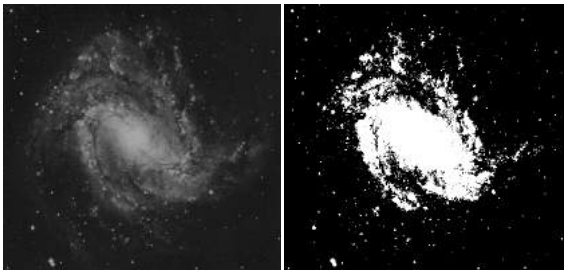


Figura 2.3. Imagen junto al resultado de su umbralización mediante el método de Otsu con *U*=68.

2.3. Negativo, brillo y contraste

A través del histograma de la imagen se pueden alterar ciertas características que ésta posee. Las más comunes son el brillo y el contraste. Además de estas dos, la obtención del negativo ha sido ampliamente utilizada en fotografía (Fig. 2.4).

- **Brillo**. Para modificar el brillo de una imagen se añade una cierta cantidad al valor de intensidad de cada uno de los píxeles. Si esta cantidad es positiva se aumenta el brillo de la imagen. Por el contrario, si es negativa, se disminuye. Si los valores obtenidos sobrepasan los extremos del intervalo de posibles niveles de intensidad, se establecen el máximo y el mínimo de dicho intervalo como nuevos valores en cada caso. Cuando se aplica la operación de brillo sobre una imagen se produce un desplazamiento del histograma.
- **Contraste**. El contraste mide la diferencia de intensidades en los colores de una imagen. El aumento o disminución del contraste de una imagen se traduce como la compresión o expansión del histograma respectivamente. Para ajustar el contraste de una imagen se han de seleccionar dos valores de intensidad entre los cuales se ajustará el nuevo histograma, $\mathrm{C_{max}}$ y $\mathrm{C_{min}}$ y aplicar las siguientes fórmulas según se desee disminuir o aumentar el contraste respectivamente.

$$I'(\mathrm{i,j}) = \left(\frac{\mathrm{C_{max}} - \mathrm{C_{min}}}{I_{\max} - I_{\min}}\right)(I(\mathrm{i,j}) - I_{\min}) + \mathrm{C_{min}} \tag{2.6}$$

$$I'(\mathrm{i,j}) = \left(\frac{I(\mathrm{i,j}) - I_{\min}}{I_{\max} - I_{\min}}\right)(C_{\max} - C_{\min}) + C_{\min} \tag{2.7}$$

  Donde I_{min} e I_{max} representan los valores de intensidad mínimo y máximo de la imagen.
- **Negativo**. El negativo de una imagen se obtiene restando el valor de cada píxel al mayor valor de intensidad posible *L-1*.

2.4. Ecualización del histograma

La ecualización del histograma de una imagen pretende obtener una distribución uniforme de los píxeles, es decir, que cada nivel de gris tenga la misma probabilidad de aparecer en la imagen. Este

método es muy utilizado para mejorar el contraste de las imágenes (Fig. 2.5). En ella se puede ver una distribución más uniforme de los niveles de intensidad en el nuevo histograma además de una mejora en la apreciación de los detalles. Para obtener la ecualización de una imagen se siguen los siguientes pasos:

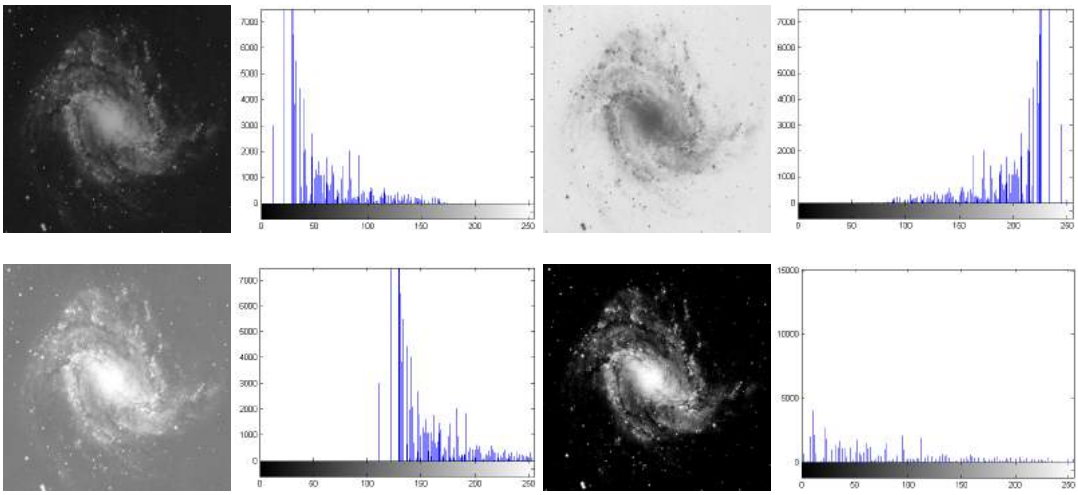


Figura 2.4. Operaciones básicas del histograma. La primera fila muestra la imagen original y su imagen en negativo mientras que la segunda contiene los resultados de aumentar el brillo y el contraste respectivamente. Todas las imágenes van acompañadas de su correspondiente histograma.

1. Se obtiene el histograma normalizado de la imagen, H, y se multiplican todas las frecuencias por el máximo valor de intensidad L-1.
2. Se calcula el histograma de frecuencias acumuladas, H'. En este caso, cada posición contiene la suma de todos los valores hasta dicha posición incluyendo el de ella misma. Posteriormente se redondean los resultados al entero más próximo.
3. El valor de cada nuevo píxel I'_{ij} es reemplazado por el valor de H' en la posición indicada por el valor de intensidad del píxel original I_{ij}, es decir, $I'_{ij} = H'(I_{ij})$.

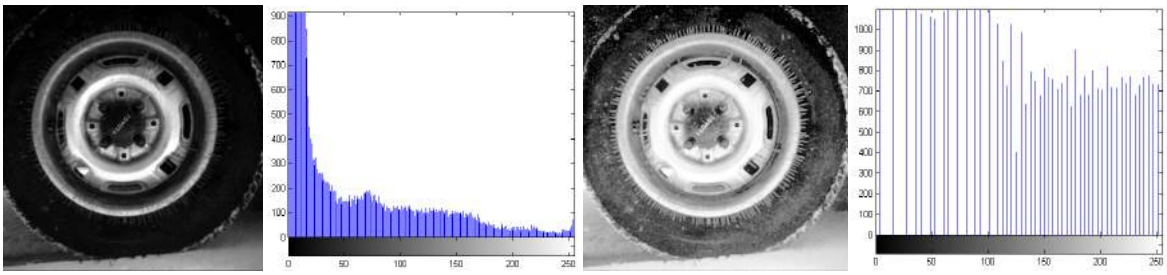


Figura 2.5. Ecualización del histograma de una imagen.

2.5. Mejora y Realce

2.5.1. Mejora del contraste

El contraste es también una forma de resaltar la textura de una imagen. La ausencia de contraste en una imagen puede ser provocada por las condiciones en las que fue tomada. La ausencia de

luminosidad es una de las posibles causas. Cuando el contraste es bajo, las líneas y bordes de la imagen se suavizan. Si queremos dar más profundidad a la imagen y definir más sus líneas y bordes deberemos subir su nivel de contraste. La Figura 2.6 es un ejemplo donde se muestra el histograma en intensidad de las imágenes.

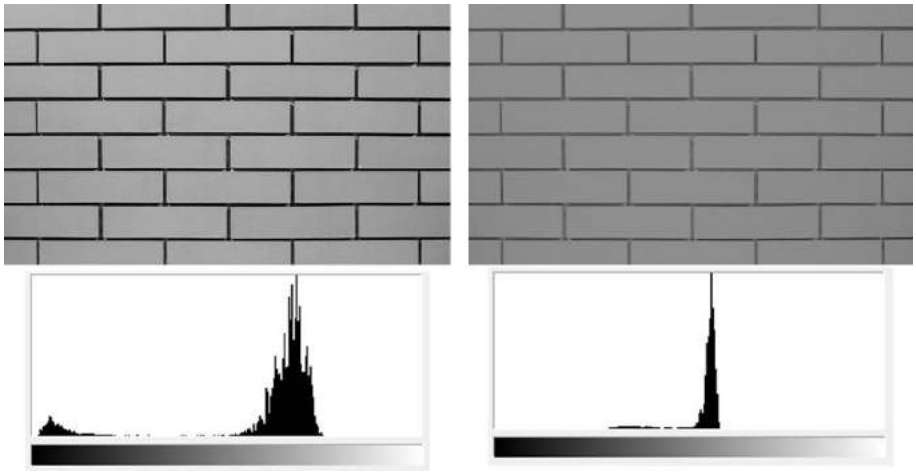


Figura 2.6. (*a*) Imagen de ladrillos de obra con alto contraste a la izquierda. (*b*) Imagen de ladrillos de obra con bajo contraste a la derecha.

2.5.2. Compresión del rango dinámico

El rango dinámico de una imagen se refiere a la relación que existe entre el mayor y menor valor de luminosidad en una imagen, Figura 2.7.

Debido al avance de la tecnología, hoy en día se pueden obtener imágenes que permiten captar de una forma más real lo que el ojo humano consigue ver. Este tipo de imágenes son las denominadas de alto nivel dinámico (*High Dynamic Range*, HDR). Las imágenes HDR tienen el objetivo de capturar todo el rango dinámico posible lo cual se consigue aumentando el número de bits por píxel para representar la imagen y con ello por tanto representando con más exactitud la luminosidad e intensidad de la imagen. Sin embargo, la mayor parte de las veces este tipo de imagen es visualizada por dispositivos con menor rango dinámico como son las pantallas de ordenador. Es necesario por tanto conseguir algún tipo de transformación de la imagen que permita comprimir su rango dinámico de la mejor forma posible. Una de las técnicas más usadas es mediante una curva de conversión que consigue comprimir el contraste en los puntos con mayor y menor luminosidad dejando el resto de puntos casi intactos.

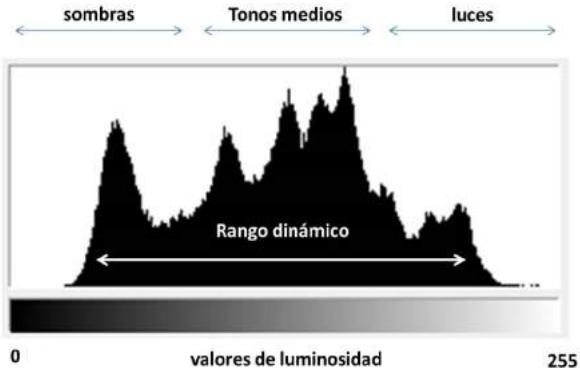


Figura 2.7. Definición visual del rango dinámico.

Las transformaciones logarítmicas y raíz cuadrada (ver ecuación 2.8 y 2.9) son las más utilizadas para la compresión del rango dinámico, donde *r* es el valor del píxel correspondiente en escala de grises, *c* es una constate de escala y *s* es el nuevo valor de dicho píxel (también a escala de grises). La imagen resultante es llamada imagen LDR (Bajo Rango Dinámico).

$$s = c \log\ (1 + |r|) \tag{2.8}$$

$$s = c \quad (|r|)^{1/2} \tag{2.9}$$

2.5.3. Realce de rangos de intensidad

Se puede dar el caso de que en una imagen únicamente nos interese quedarnos o resaltar un determinado rango de niveles de grises y ocultar el resto. Podremos utilizar las operaciones de resta y promedio de imágenes para localizar dichos valores.

2.5.3.1. Resta de imágenes

La resta de imágenes es un método sencillo pero a la vez muy utilizado en la visión por computador para realizar operaciones como la substracción de fondo o la detección de objetos en video. En este último caso, imaginemos que tenemos una cámara de video que constantemente está grabando una escena, una de las formas de saber si en algún momento alguien o algo entran en dicha escena es mediante la resta entre dos *frames* consecutivos.

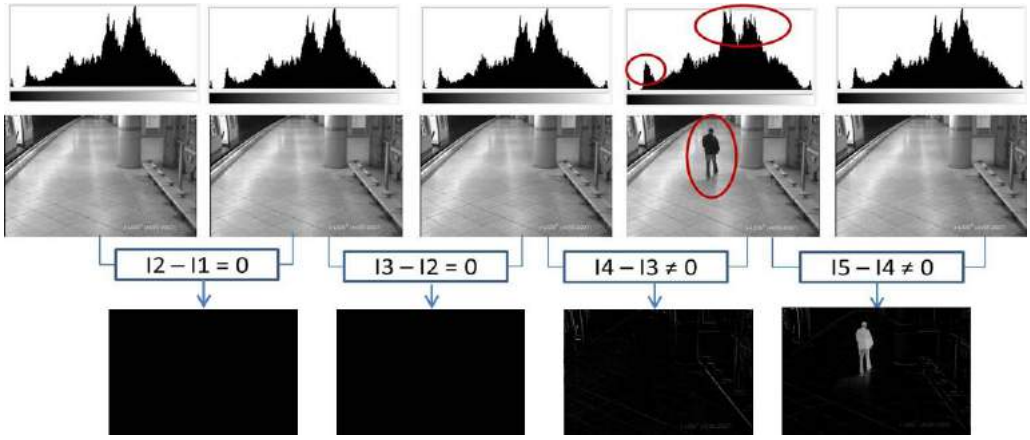


Figura 2.8. *Frames* de un video donde se capta la presencia de un objeto en la escena.

Este método es muy utilizado por ejemplo en las aplicaciones de videovigilancia. La Figura 2.8 muestra cómo la diferencia entre los *frames* detecta un cambio, que en este caso es la persona que entra en la escena.

2.5.3.2. Promedio de imágenes

Las señales se promedian para eliminar el ruido, lo mismo puede ser aplicado a las imágenes. El campo de la microscopía es uno de los que más utilizan el promedio de imágenes. Cuando se extrae una pila de imágenes seguida desde el microscopio se pueden producir desajustes en la iluminación que son provocadas por el sensor de la cámara. Esto provoca que posteriormente puedan existir problemas cuando se quieran realizar procesamientos sobre dichas imágenes, en especial segmentaciones o umbralizaciones. El promedio de imágenes es comúnmente aplicado en el mismo momento en el que se adquieren las imágenes y consiste en realizar la suma de las imágenes adquiridas y dividir el resultado entre el número total de imágenes que se sumaron (Pertusa, 2010).

2.5.4. Filtrado en el dominio de la frecuencia

Cuando se habla del filtrado en el dominio de la frecuencia, en lo primero que se piensa es en la transformada de Fourier. La transformada de Fourier de una función representa el espectro o distribución de frecuencias de la misma. Por tanto, cuando la función tiene cambios bruscos, sus componentes son de alta frecuencia y cuando no los tiene, es relativamente homogénea, tiene componentes de baja frecuencia. En el ámbito del tratamiento de imágenes, la función *f(x,y)* representa los niveles de intensidad de los píxeles a tratar y se define la transformada discreta de Fourier en 2D como se muestra en la ecuación (2.10), (González y Woods, 2007).

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x,y)\, e^{-j2\pi(\frac{ux}{N}+\frac{vy}{M})} \tag{2.10}$$

Por tanto, en la transformada discreta de Fourier las variables *x* e *y* representan las columnas y filas de la imagen, mientras que las variables *u* y *v* denotan las frecuencias verticales y horizontales. Estas frecuencias verticales y horizontales son representadas en el espacio transformado de magnitud de Fourier (|*F(u,v)*|) como distribuciones horizontales y verticales respectivamente. El punto central en la imagen de magnitud es el denominado componente *DC* (*direct current*) que representa la intensidad media de toda la imagen, Figura 2.9. Cuando la transformada discreta 2D de Fourier es aplicada en una imagen, es posible distinguir los bordes principales de la imagen como líneas de frecuencia en la imagen de magnitud de Fourier, Figura 2.10 (Fernández-Carrobles, 2015).

El filtrado de las imágenes en el domino de la frecuencia seguirá el proceso mostrado en la Figura 2.11, donde *H(u,v)* representa el tipo de filtro de frecuencia (Gaussiano, Laplaciano, segunda derivada Gaussiana, etc)

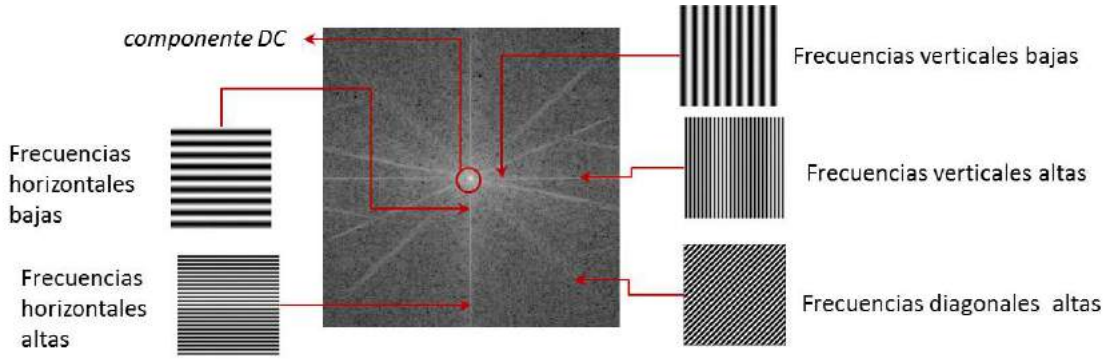


Figura 2.9. Frecuencias y orientaciones en el espacio transformado de Fourier.

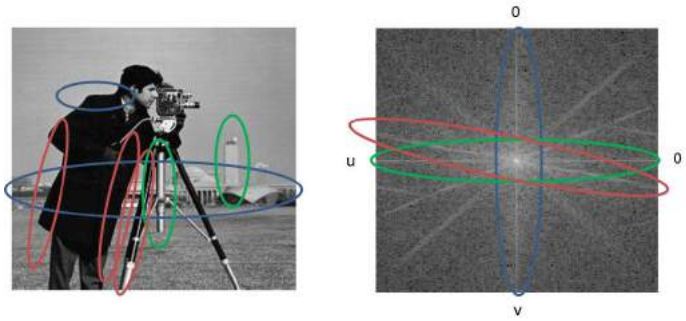


Figura 2.10. Líneas verticales, horizontales y diagonales de la imagen representadas como frecuencias horizontales, verticales y diagonales en la imagen de magnitud de Fourier.

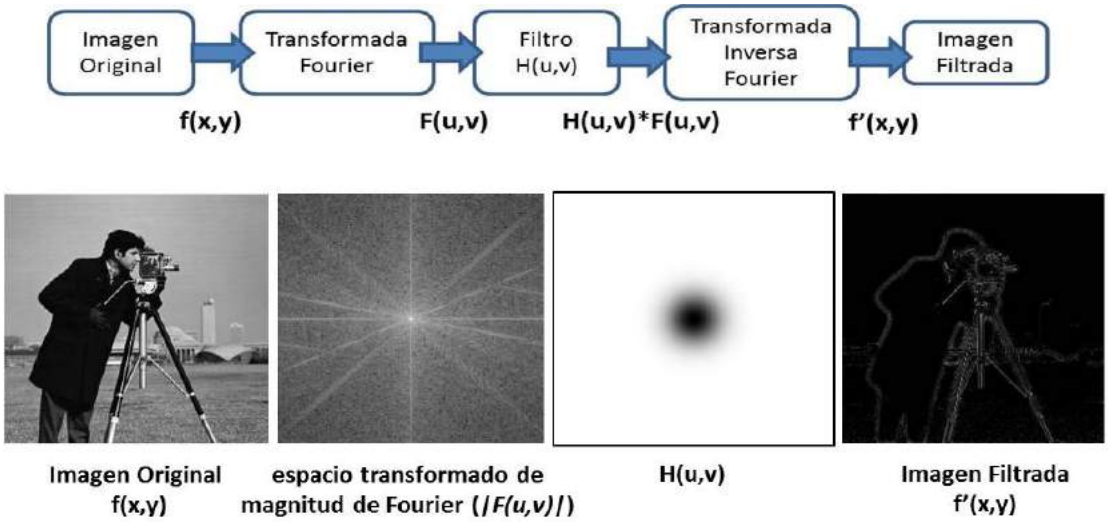


Figura 2.11. Filtrado de imágenes en el dominio de la frecuencia

2.5.4.1. Filtros paso bajo

El filtro paso bajo es aquel que deja pasar las bajas frecuencias y atenúan o eliminan las altas frecuencias. Esto se traduce en el dominio del espacio como un suavizado de la imagen, es decir, la eliminación de ruido y otros pequeños detalles en la imagen. Uno de los filtros paso bajo más utilizado es el filtro paso bajo Gaussiano, Figura 2.12, donde $D(u,v)$ es la distancia euclídea de (u,v) al origen del plano de frecuencias y D_0 es la denominada frecuencia de corte (debe ser un número no negativo).

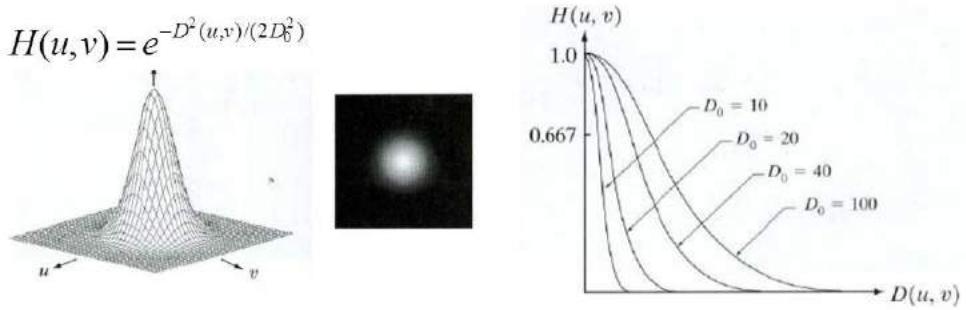


Figura 2.12. Filtro paso bajo Gaussiano

La Figura 2.13 muestra un ejemplo de su utilización con diferentes valores de radio.

Figura 2.13. Filtro paso bajo Gaussiano con diferentes radios

2.5.4.2. Filtros paso alto

Los filtros paso alto son justo lo contrario que los paso bajo, dejan pasar las altas frecuencias y atenúan o anulan las bajas frecuencias. Esto provoca que se resalten las zonas donde existen cambios bruscos en los niveles de intensidad, como son los bordes. Igualmente, uno de los filtros más utilizados para estos casos es el filtro de paso alto Gaussiano (inversión del filtro de paso bajo Gaussiano),

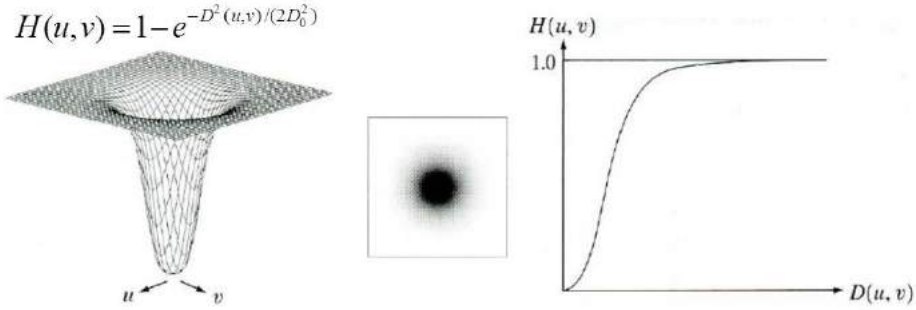


Figura 2.14. Filtro paso alto Gaussiano

La Figura 2.15 muestra un ejemplo de la utilización de un filtro paso alto Gaussiano con diferentes valores de radio.

Figura 2.15. Filtro paso alto Gaussiano con diferentes radio

2.5.4.3. Filtros paso banda

Los filtros paso banda son aquellos que atenúan o eliminan las bajas y altas frecuencia dejando pasar aquellas que pertenecen a una banda determinada. En el lado contrario a los filtros paso banda encontramos los filtros de rechazo de banda, que realizan la operación contraria, no permiten el paso de las frecuencias pertenecientes a una determinada banda dejando el resto de frecuencias intactas.

2.5.5. Filtrado en el dominio del espacio

En el caso de realizar el filtrado de las imágenes en el dominio del espacio las operaciones se llevan a cabo directamente sobre los píxeles de la imagen y no sobre su transformada en el dominio frecuencial. En concreto, para obtener el nuevo valor de cada uno de los píxeles se utiliza el píxel referencia y una serie de píxeles localizados dentro de una ventana alrededor de éste. Los filtros en el espacio pueden ser lineales o no lineales. Los filtros lineales están basados en los llamados *kernels* o máscaras de convolución (Acharya y Ray, 2005).

Un *kernel* no es más que una matriz de coeficientes que asigna una serie de valores a los píxeles vecinos del píxel de referencia $I(i,j)$. Los filtros basados en *kernels* pueden ser representados como una ventana móvil de coeficientes que recorre toda la imagen centrada en cada uno de sus píxeles. El nuevo valor de cada píxel se obtiene en función de los valores del *kernel*, del propio píxel y de sus vecinos. Por tanto, la aplicación de un *kernel* sobre una imagen equivale a realizar la convolución de ambas matrices.

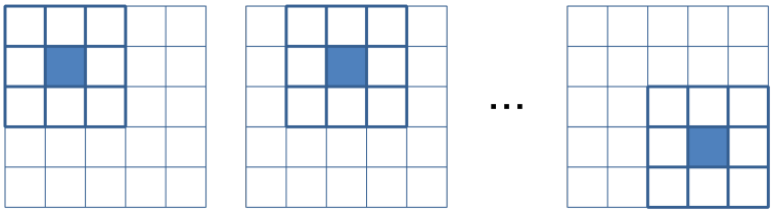


Figura 2.16. Representación de un *kernel* y el movimiento que realiza para recorrer toda la imagen.

Si K es un *kernel* de 3x3, I es la imagen de entrada e I' es la imagen obtenida al aplicar el *kernel*, cada píxel nuevo en la posición *(i,j)* se calcula como:

$$I'(i,j) = \sum_{\substack{m=-1 \\ n=-1}}^{\substack{m=1 \\ n=1}} I(i+m, j+n) \cdot K(m+1, n+1) \tag{2.11}$$

Dependiendo de si el filtro suaviza los pequeños detalles o los realza, puede clasificarse como filtro paso bajo o paso alto. Desde este punto de vista no se considera el punto de vista de la frecuencia, sino el espacial.

2.5.5.1. Filtros paso bajo

Los filtros paso bajo realizan un suavizado de la imagen. Este tipo de filtros elimina ruido y otros pequeños detalles que no son de interés. Afecta, por tanto, a las zonas que presentan muchos cambios de intensidad. Entre los filtros de suavizado destacan los de media, media ponderada, mediana, adaptativo y Gaussiano.

- **Media.** Cada píxel en la imagen de salida es el resultado de la media aritmética de los píxeles del vecindario del píxel de referencia. Por tanto, cada celda de una ventana o *kernel* de 3x3 cuyo valor de la celda central puede ser 0 ó 1 dependiendo de si su valor es utilizado o no para el cálculo y el resto de valores es 1. Una vez multiplicados todos los coeficientes de las celdas de la ventana por los correspondientes valores en la imagen, el resultado es dividido por el número total de coeficientes distintos de 0 utilizados. A continuación se muestra una máscara o *kernel* de ejemplo, *M*.

$$M = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} /9 \tag{2.12}$$

- **Media ponderada**. Se trata de un filtro de media donde los coeficientes son distintos de 1. En este caso, cada píxel tiene un peso en función de la posición que ocupa y el resultado de multiplicarlos por los correspondientes valores en la imagen es dividido por la suma de los coeficientes.

$$M = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} /10 \tag{2.13}$$

- **Mediana**. El filtro de mediana es similar al de media. Este filtro asigna la mediana de los valores que caen dentro de la ventana al píxel destino. Utilizar la mediana y no la media tiene como ventaja la obtención de una imagen menos borrosa. Por último, el filtro de la mediana es menos sensible a valores extremos por lo que obtiene muy buenos resultados eliminando ruido de tipo *salt-and-pepper* y *speckle*.
- **Adaptativo.** Es considerablemente más complejo que los anteriores ya que los coeficientes de la ventana o *kernel* son recalculados para cada uno de los píxeles. Un ejemplo de este tipo de filtros consistiría en calcular los coeficientes del *kernel* en función de la varianza de los valores de los píxeles que están dentro de la ventana en ese momento. Así, el filtro podría

realizar un mayor o menor suavizado dependiendo de la zona. Al igual que el filtro de mediana obtiene buenos resultados eliminando el ruido.

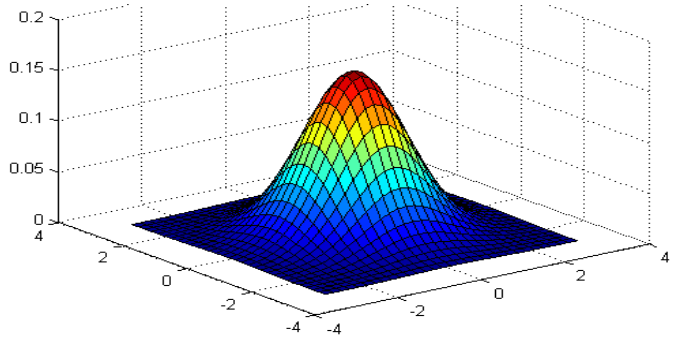
- **Gaussiano**. El *kernel* utilizado en este tipo de filtros contiene como coeficientes valores que siguen una distribución Gaussiana bivariante. Si la media y la varianza son las mismas para los dos ejes los valores pueden ser obtenidos mediante:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.14}$$

El valor máximo aparece en el píxel de referencia y disminuye en función de la varianza establecida σ^2. Por tanto, la contribución de cada píxel al valor final disminuye con la distancia. En la práctica, este tipo de filtros se utilizada como etapa de preprocesado para la detección de bordes u otras operaciones.

2.5.5.2. Filtros paso alto

Los filtros paso alto realizan un realzado de los detalles de la imagen. Este tipo de filtros atenúa las zonas uniformes y enfatiza los detalles de interés. Al contrario que los filtros paso bajo, eliminan la componente media. El *kernel* utilizado suele dar un peso alto al punto de referencia y valores bajos y negativos a la vecindad según la ecuación (2.15). Así, si el nivel de intensidad del píxel de referencia es mucho mayor que el de sus vecinos el efecto de éstos es bajo y el valor de salida acentúa esa diferencia. Por el contrario, si los valores son similares, el resultado del filtro es similar a un promedio. Existen dos tipos de procedimientos: basados en la sustracción de la media y basados en derivadas.



$$M = \begin{bmatrix} 0.153 & 0.156 & 0.153 \\ 0.156 & 0.159 & 0.156 \\ 0.153 & 0.156 & 0.153 \end{bmatrix}$$

Figura 2.17. Representación de la función Gaussiana en 2D con $\mu = 0$ y $\sigma = 1$ y la máscara de 3x3 que se obtiene, *M*.

$$\mathrm{M}= \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{2.15}$$

- **Sustracción de la media**. Este tipo de métodos suma a la imagen original la diferencia entre ella misma y el resultado de aplicarle un filtro paso bajo.
- **Derivadas**. Ya que una imagen es la discretización de una función y = f(x,y), su segunda derivada da información acerca de cómo son los cambios que se producen entre zonas

contiguas de una imagen. Para ello se utiliza el operador laplaciano. La ecuación (2.16) muestra un par de ejemplos.

$$Laplaciano \rightarrow M = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$
$$Laplaciano + Imagen \rightarrow M = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad (2.16)$$

2.5.6. Realce y detección de bordes

El realce de bordes de una imagen se suele realizar a través de filtros espaciales. A continuación se exponen los métodos para realce de bordes más utilizados (Aldalur y Santamaría, 2002).

- **Desplazamiento y sustracción**. Si dos píxeles adyacentes tienen valores de intensidad muy diferentes, lo que sucede con un píxel situado en un borde y alguno de sus vecinos, su diferencia será alta. Por tanto, si a una imagen se le resta ella misma pero desplazada horizontal o verticalmente se obtiene como resultado una imagen con los bordes en el eje perpendicular al desplazamiento.
- **Gradiente**. El módulo del gradiente de la imagen es comúnmente utilizado para realzar los bordes. En concreto, el valor utilizado es el módulo del gradiente obtenido a partir de las derivadas discretas en las direcciones *x* e *y* de la imagen.

$$|\nabla I(x,y)| = \sqrt{\left(\frac{\partial}{\partial x} I(x,y)\right)^2 + \left(\frac{\partial}{\partial y} I(x,y)\right)^2} \quad (2.17)$$

- **Gradientes direccionales**. Además de las direcciones de los dos ejes principales de la imagen, se pueden realizar filtrados en otras direcciones. Los filtros direccionales se utilizan para realzar estructuras que siguen una determinada dirección en el espacio. Estos filtros actúan con diferente valor ponderado según la dirección en la que busquen las variaciones. Los coeficientes del *kernel* utilizado anulan la componente en una determinada dirección para realzar la componente perpendicular a ésta, ecuación (2.18).
- **Kirsch**. El operador de Kirsch aplica cada una de las orientaciones de una máscara direccional y establece el máximo como valor de salida.

$$\underset{NO}{\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & -1 \\ 1 & -1 & -1 \end{bmatrix}} \quad \underset{N}{\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}} \quad \underset{NE}{\begin{bmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{bmatrix}}$$
$$\underset{O}{\begin{bmatrix} 1 & 1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & -1 \end{bmatrix}} \quad \underset{E}{\begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}} \quad (2.18)$$

$$\underset{SO}{\begin{bmatrix}1 & -1 & -1\\ 1 & -2 & -1\\ 1 & 1 & 1\end{bmatrix}} \quad \underset{S}{\begin{bmatrix}-1 & -1 & -1\\ 1 & -2 & 1\\ 1 & 1 & 1\end{bmatrix}} \quad \underset{SE}{\begin{bmatrix}-1 & -1 & 1\\ -1 & -2 & 1\\ 1 & 1 & 1\end{bmatrix}}$$

- **Laplaciano**. Los filtros laplacianos están basados en la segunda derivada. Estos filtros también pueden ser considerados como filtros direcciónales que actúan en todas las direcciones posibles. La ecuación (2.19) muestra su máscara más común.

$$M = \begin{bmatrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{bmatrix} \tag{2.19}$$

  Sin embargo, este tipo de filtros es muy sensible al ruido. Al igual que los bordes, el ruido también se ve realzado tras aplicarlos.

- **Prewitt**, **Sobel** y **Frei-Chen**. Estos operadores calculan una aproximación de las primeras derivadas en los ejes *x* e *y*. Las máscaras de los tres solo difieren en un par de valores (ecuación 2.20). Para Prewitt $k = 1$, para Sobel $k = 2$ y para Frei-Chen $k = \sqrt{2}$. El operador de Sobel es algo más sensible que el de Prewitt a los bordes diagonales. Por otro lado, el operador de Frei-Chen presenta el mismo valor para los bordes verticales, horizontales y diagonales.

$$M_x = \begin{bmatrix}1 & 0 & -1\\ k & 0 & -k\\ 1 & 0 & -1\end{bmatrix} \quad M_y = \begin{bmatrix}-1 & -k & -1\\ 0 & 0 & 0\\ 1 & k & 1\end{bmatrix} \tag{2.20}$$

  Canny. Este método utiliza el gradiente de la imagen para realzar los bordes. Ya que el gradiente es sensible al ruido, el método de Canny utiliza dos umbrales para detectar bordes más o menos marcados. De esta manera, los bordes con valores por debajo del umbral más pequeño son descartados, los que tienen valores entre ambos umbrales son marcados como bordes débiles y los que tienen valores mayores que el mayor de los umbrales son marcados como bordes fuertes.

- **Roberts**. El operador de Roberts tiene como ventaja la obtención de una buena respuesta ante los bordes diagonales. Sin embargo, tiene como inconveniente su sensibilidad al ruido. La ecuación (2.21) muestra las máscaras utilizadas por este operador.

$$M_x = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 1\\ 0 & -1 & 0\end{bmatrix} \quad \boldsymbol{M}_y = \begin{bmatrix}-1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix} \tag{2.21}$$

La figura (2.18.) muestra el resultado de aplicar algunos de estos métodos para detección de bordes.

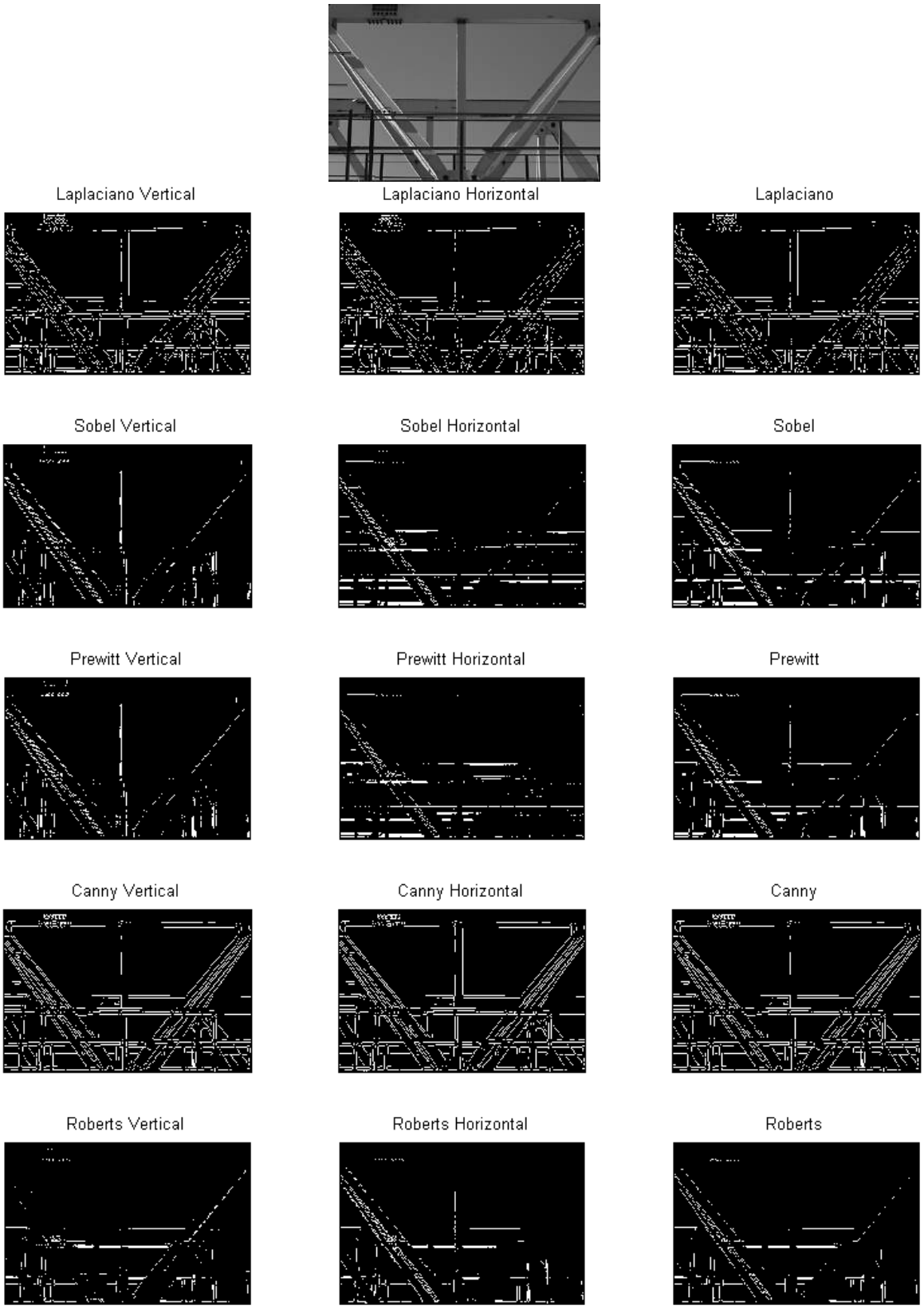


Figura 2.18. Resultado de algunos operadores de detección de bordes. Tanto el algoritmo de Canny como el Laplaciano han sido aplicados después de un suavizado Gaussiano.

2.6. Bibliografía

Acharya, T.; Ray, A. K. (2005). *Image processing: principles and applications*. John Wiley & Sons.
Aldalur, B.; Santamaría, M. (2002). Realce de imágenes: filtrado espacial. *Revista de Teledetecccción*, vol. 17, pp. 31-42. Disponible online: http://www.aet.org.es/revistas/revista17/AET17-04.pdf (accedido 29 Abril 2015).
Bueno, G.; Deniz, O.; Espinosa-Aranda, J.L.; Salido, J.; Serrano, I.; Vállez, N.; (2015). *Learning Image Processing with OpenCv.* Packt Publishing.
González, R.C.; Woods, R.E. (2007) *Digital Image Processing*, 3ª ed. Prentice Hall.
Fernández-Carrobles, M.M. (2015). Tesis doctoral: Computer Aided Breast Cancer Detection and Diagnosis System based on Histopathological Image Analysis of TMAs. Universidad de Castilla-La Mancha, E.T.S.I.Industriales.
Freeman, M. (2005*). The Digital SLR Handbook*. Ilex. ISBN 1-904705-36-7.
Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Trans. Sys., Man., Cyber*. vol. 9, pp. 62-66. doi:10.1109/TSMC.1979.4310076.
Pertusa, J.F. (2010) Técnicas de análisis de imagen. Aplicaciones en la biología, 2ª ed. Universitat de Valencia. Servei de publications.

CAPÍTULO 3

EL COLOR: MODELOS Y TRANSFORMACIONES DE LOS ESPACIOS DE COLOR

David MARTÍN[1], Fernando GARCÍA[1], José Mª ARMINGOL[1]

[1]Laboratorio de Sistemas Inteligentes (Dpto. de Ingeniería de Sistemas y Automática), Universidad Carlos III de Madrid, Leganés (Madrid), España

El color es una de las características que nos permite a los seres humanos identificar y clasificar los objetos. La percepción visual del color se genera en nuestro cerebro a partir de las ondas electromagnéticas reflejadas por los objetos y captadas por los ojos.

Desde el punto de vista del procesamiento de imágenes por computador, es preciso recurrir a los llamados espacios de color, se trata de conjuntos de fórmulas matemáticas que permiten describir los colores y descomponerlos en distintos canales. Los espacios de color más utilizados son el RGB y el CMYK, debido a que el modelo RGB se utiliza en periféricos como pantallas, cámaras y escáneres, y el modelo CMYK en impresoras. En este capítulo se analizará también el sistema de coordenadas tridimensional (tono, saturación e intensidad) del espacio de color HSI, donde cada color está representado por un punto en el espacio. Los dos espacios siguientes que se estudiarán fueron establecidos por la Comisión Internacional de Iluminación (CIE): el espacio de color XYZ se usa actualmente como referencia para definir los colores que percibe el ojo humano, mientras que el Lab se puede considerar como el más completo de los desarrollados por la CIE, ya que permite identificar cada color de una forma muy precisa mediante sus valores "a" y "b" y su brillo ("L"). Finalmente, se analizará el espacio de color YCbCr.

3.1. Introducción

Cualquier aspecto visual que podamos cuantificar de un objeto debe su existencia a la luminosidad y al color. Los límites que determinan la forma de los objetos se derivan de la capacidad del ojo para

distinguir entre sí zonas de luminosidad y color diferentes, sin embargo, aunque el color es una de las características más importantes que define a los objetos, hasta hace poco tiempo no se le prestaba mucha atención en la visión por computador, debido al coste computacional y a la memoria necesaria para el procesamiento de este tipo de imágenes (Forsyth, 2012).
Antes de empezar a trabajar con el color, dentro de los sistemas de visión, es necesario que introduzcamos algunas definiciones básicas imprescindibles para comprender el color, éstas son:
• Brillo: es la sensación que indica si un área está más o menos iluminada.
• Tono: es la sensación que indica si un área es similar al rojo, amarillo, verde o azul o a una proporción de dos de ellos.
• Colorido: disposición y grado de intensidad de los diversos colores.
• Luminosidad: es el brillo de una zona respecto a otra blanca en la imagen.
• Croma: es el colorido de un área respecto al brillo de un blanco de referencia.
• Saturación: es la relación entre el colorido y el brillo.

Los parámetros relacionados con la percepción humana son la luminancia, el tono y la saturación (Gevers y col., 2012). Para transmitir estos parámetros cromáticos, es preciso transformarlos antes en parámetros eléctricos, si la escena es capturada por una cámara, y en la recepción deshacer la conversión. Los parámetros eléctricos correspondientes pueden ser:
Tres señales de luminancia cromática: roja, verde y azul (R, G, B). Se trata del sistema más intuitivo y de hecho es en el que se basan las cámaras de vídeo para adquirir imágenes en color.
Una señal de luminancia monocroma o acromática, una señal de crominancia roja y una señal de crominancia azul (Y, R-Y, B-Y). La especificación de un color monocromático $x = C_\lambda$, caracterizado por su longitud de onda única λ, se efectúa al compararlo con el color blanco base o blanco referencia. En la señal Y se introduce una línea de retardo que pone en fase las señales de luminancia y crominancia.
Mediante la modulación en cuadratura de una subportadora, por las señales de crominancia roja y azul, aparecen dos nuevos parámetros eléctricos: la fase de la subportadora (que transmite el tono) y la amplitud de la subportadora (que transmite la saturación). Ambos parámetros desde el punto de vista de la colorimetría definen el aspecto cualitativo del color, estando el cuantitativo vinculado a la luminancia.

3.2. Espacios de Color

Existen diversos espacios para representar los colores. Un espacio de color es un método de representación por el que se pueda especificar, crear o visualizar cualquier color. La especificación numérica de un color se efectúa mediante tres cantidades (luminancias) que definen dicho color de forma cualitativa y cuantitativamente. Dependiendo del tipo de sensor y aplicación se han desarrollado diversos espacios de colores que facilitan el empleo de esta característica. Así el hombre distingue un color de otro dependiendo de su tono, brillo y colorido. En la televisión se trabaja con la componente roja, verde y azul, mientras que para la impresión se utiliza cían, magenta, amarillo y negro.

3.2.1. Espacio RGB.

Variando las cantidades de los colores primarios (rojo, verde y azul), se pueden obtener numerosos colores *x* por mezcla o suma aritmética, ecuación (3.1):

$$x = r + g + b \tag{3.1}$$

La naturaleza del color obtenido es trivariante, al ser los tres colores primarios independientes entre sí, es decir, un color *x* de luminancia L, se puede igualar mediante tres luminancias R, G, B, o cantidades adecuadamente dosificadas de tres componentes primarias (Szeliski, 2010).

Los colores rojo, verde y azul representan las tres componentes elementales y definen el espacio de color básico, figura X.1, donde cada color es un punto en tres dimensiones, perteneciente a la superficie o el interior del cubo. Cada una de estas tres componentes se obtiene por la proyección de la escena captada sobre el espacio 3-D del sensor. La ecuación que establece esta transformación es la siguiente:

$$C = \int_{\lambda} E(\lambda) S_C(\lambda) d\lambda \qquad para \quad C = (R, G, B) \tag{3.2}$$

Siendo E(λ) la intensidad de la luz y $S_C(\lambda)$ tres hipotéticos filtros de colores. En el Ejemplo 3.1. se muestra una imagen en color y su descomposición en el espacio RGB.

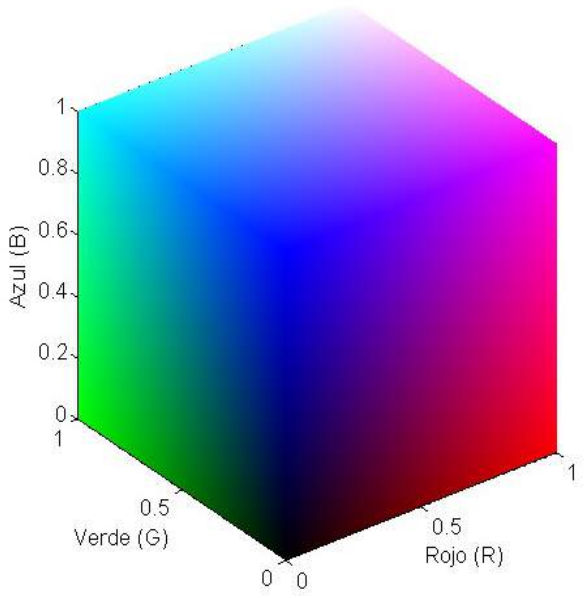


Figura 3.1. Espacio de color RGB representado en forma de cubo.

El principal problema que presenta el espacio RGB es que la segmentación de imágenes debe ser desarrollada en el espacio 3-D, como consecuencia de la dependencia existente a las variaciones uniformes de los niveles de luz. Para evitar este problema surge la necesidad de normalizar el espacio de color RGB, la transformación de normalización viene dada por la ecuación (3.3):

$$N_C = \frac{C}{(R + G + B)} \qquad para \qquad C = (R, G, B) \tag{3.3}$$

Considerando que la expresión anterior es redundante (N_B =1- N_R- N_G), el espacio de color RGB normalizado suele formularse como (3.4):

$$Y = c_1 R + c_2 G + c_3 B$$
$$T_1 = \frac{R}{(R+G+B)} \qquad (3.4)$$
$$T_2 = \frac{G}{(R+G+B)}$$

Donde c_1, c_2 y c_3 son constantes que deberán cumplir la relación $c_1+c_2+c_3=1$. El coeficiente Y se interpreta como la luminancia de la imagen a nivel de píxel, mientras que T_1 y T_2 son variables cromáticas que se pueden considerar independientes de la iluminación. A continuación, el ejemplo 3.1., explica el resultado de separar los tres canales del espacio de color RGB.

Ejemplo 3.1. *Imagen en color y obtención de su descomposición en el espacio RGB*

Solución: A continuación se muestra una imagen en color y las tres imágenes que descomponen el color en tres canales: rojo, verde y azul, figura 3.2. Como se puede observar las tres imágenes pertenecientes a cada plano de color están en escala de grises, donde cada píxel posee un valor en el rango [0, 255], en el caso de que la representación de cada píxel de la imagen tome valores enteros de un byte.

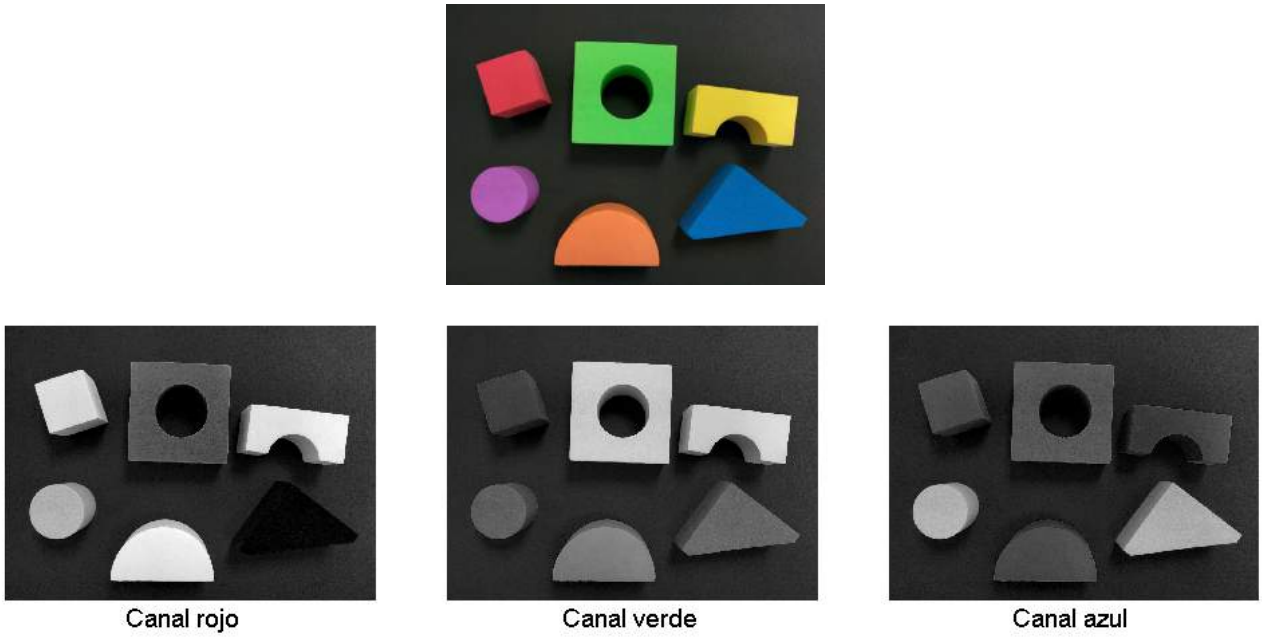


Figura 3.2. Espacio de color RGB: descomposición en sus tres canales

3.2.2. El espacio HSI.

El espacio de color HSI se basa en el modo de percibir los colores que tenemos los humanos. Dicho sistema caracteriza el color en términos de tono o tinte (hue), saturación (saturation) y brillo (intensity); componentes que se muestran favorables de cara a realizar segmentaciones de la imagen en atención al tono o tinte del color, con mayor invariancia a la iluminación. Por ejemplo, si se tienen dos

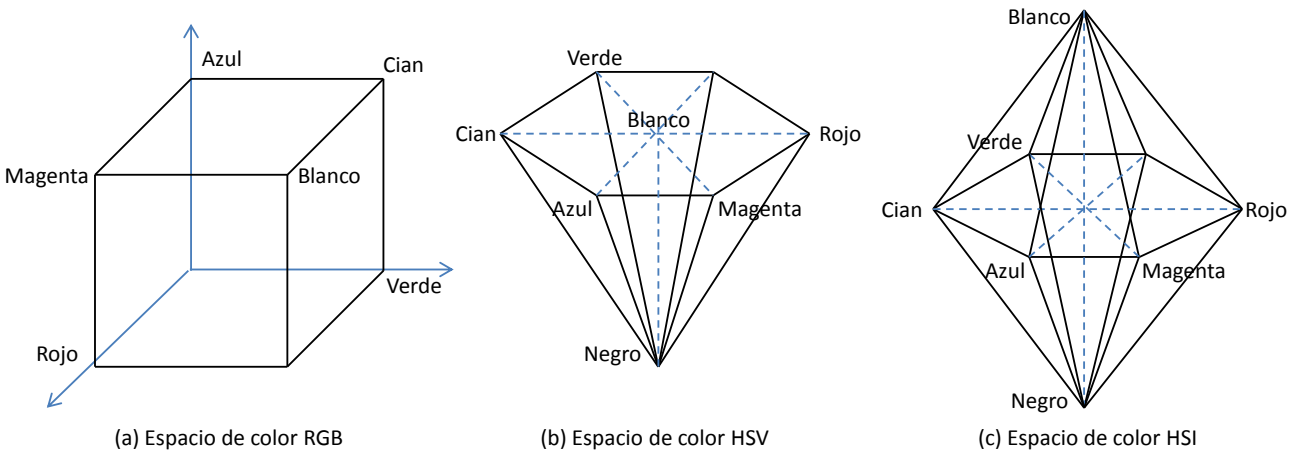
fuentes de luz con el mismo espectro, mediante el espacio RGB, aquella que presente mayor intensidad será la que aparecerá con un color más brillante (Yoshida, 2010). Sin embargo mediante el espacio HSI, podemos segmentar colores basándonos en el tono, una luz verde y otra roja se distinguen por su distinto tono (hue) que representa la longitud de onda de ese color, sin necesidad de acudir a la intensidad lumínica. Además, algunos colores que se encuentran en la naturaleza (el púrpura por ejemplo) no se puede obtener a partir de la descomposición de la luz blanca, sino que se obtiene de la mezcla de dos colores (azul y rojo), por lo tanto no puede representarse con el modelo RGB, aunque en este caso se usan aproximaciones. La componente de saturación permite distinguir la blancura de un color (el color rojo presenta máxima saturación y el color rosa mínima).

La transformación matemática que permite el paso del espacio de color RGB al HSI es realizada mediante hardware específico para tal propósito, debido a su elevado coste computacional. La figura 3.3 muestra las tres estructuras tridimensionales que representan los espacios de color RGB y HSI. Las ecuaciones de transformación entre el espacio RGB y el HSI son las propo

$$I = \frac{R+G+B}{3}$$

$$H = \arctan\left(\frac{\sqrt{3}(G-B)}{(R-G)+(R-B)}\right) \qquad (3.5)$$

$$S = 1 - \frac{min(R,G,B)}{I}$$



(a) Espacio de color RGB (b) Espacio de color HSV (c) Espacio de color HSI

Figura 3.3. (*a*) Espacio de color RGB representado en forma de cubo (*b*) Espacio de color HSV representado en forma de cono, blanco en el centro del cono y negro en el vértice inferior (*c*) Espacio de color HSI representado en forma de bipirámide hexagonal, donde el vértice inferior corresponde al color negro y el vértice superior al color blanco.

En la expresión del tono $H=arctan(y/x)$, se emplea el signo de x e y para establecer el cuadrante al que pertenece el ángulo resultante. Se puede considerar que el tono indica el ángulo formado entre el eje de referencia (el correspondiente al color rojo) y el punto que ocuparía el color tratado en el espacio RGB.

A partir de la figura anterior se pueden establecer las componentes *x* e *y* de un píxel arbitrario mediante las siguientes relaciones trigonométricas:

$$x = R - \frac{G+B}{2} = \frac{1}{2}\left[(R-G)+(R-B)\right]$$
$$y = \frac{\sqrt{3}}{2}(G-B) \tag{3.6}$$

Así como el espacio RGB se representa por un cubo, el HSI lo forman dos hexaedros unidos por su base, figura 3.3(c). Dependiendo de la intensidad se tiene un corte a los hexaedros. Dentro del hexágono obtenido el color viene definido por el tono, representado en grados (el rojo es 0º, el verde 120º y azul 240º) y la saturación indica la distancia al centro del hexágono. Si bien esta misma representación puede realizarse en forma de cono, figura 3.4.

Experimentalmente se puede comprobar que para colores saturados la desviación estándar del tono es del orden de 2,5 a 6 grados, mientras que para colores no saturados el intervalo anterior aumenta considerablemente, no siendo recomendado el uso de este espacio de color.

Variaciones de este espacio de color son las siguientes:

- HSL (Hue, Saturation and Lightness)
- HSV (Hue, Saturation and Value) (figura 3.3(b) y 3.4)
- HCI (Hue, chroma / colourfulness and Intensity)
- HVC (Hue, Value and Chroma)
- TSD (Hue, Saturation and Darkness)

Las expresiones más comúnmente utilizadas en la transformación del espacio RGB al HSI vienen definidas por las ecuaciones (3.7) y (3.8):

$$I = \frac{R+G+B}{3} \tag{3.7}$$
$$H = \frac{\alpha - arctan\left(\frac{\sqrt{3}(R-I)}{G-B}\right)}{2\pi}$$

con:

$$\alpha = \frac{\pi}{2} \text{ si G>B} \qquad \alpha = \frac{3\pi}{2} \text{ si G>B}$$

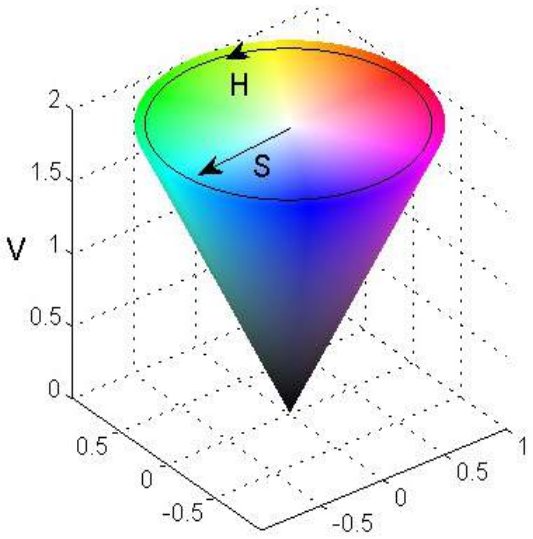


Figura 3.4. Espacio de color HSV representado en forma de cono invertido. Para elegir un color, primero se selecciona el tono (H) en la región circular, valores de 0º a 360°, donde cada valor corresponde a un color, por ejemplo, 0º (rojo), 60º (amarillo) y 120º (verde), y posteriormente se selecciona la saturación del color en el eje horizontal del cono (S) y el valor del color en el eje vertical (V).

En el caso de tener un color con igual componente verde y azul (G=B), el tono será igual a la unidad (H=1).

$$S = \sqrt{R^2 + G^2 + B^2 - 2RG - RB - BG} \tag{3.8}$$

La conversión inversa, es decir, aquella que permite recuperar de nuevo la información de la imagen en el espacio RGB a partir del HSI (Gonzalez y Woods, 2007), viene dada por las siguientes relaciones en función de los valores tomados por el tono (3.9), (3.10), (3.11):

Si $0^{\circ} < H \leq 120º$

$$\begin{aligned} B &= \frac{1}{3}(1-S) \\ R &= \frac{1}{3}\left(1+\frac{S\cos H}{\cos(60^o - H)}\right) \\ G &= 1-(B+R) \end{aligned} \tag{3.9}$$

$120º < H \leq 240º$

$$\begin{aligned} H &= H - 120º \\ R &= \frac{1}{3}(1-S) \\ G &= \frac{1}{3}\left(1+\frac{S\cos H}{\cos(60^o - H)}\right) \\ B &= 1-(R+G) \end{aligned} \tag{3.10}$$

$240^{\circ} < H \leq 360^{\circ}$

$$
\begin{aligned}
H &= H - 240^{o} \\
G &= \frac{1}{3}(1-S) \\
B &= \frac{1}{3}\left(1+\frac{S\cos H}{\cos(60^{o}-H)}\right) \\
R &= 1-(G+B)
\end{aligned}
\tag{3.11}
$$

Los inconvenientes que presenta el uso del espacio de color HSI son:

- No linealidad de las transformaciones realizadas,
- La singularidad existente en el entorno del eje definido por R=G=B (saturación = 0), que provoca que los valores obtenidos para el tono sean inestables.

A continuación, se muestra el Ejemplo 3.2 donde se observa la visualización de una imagen en el espacio de color HSV (tres canales), y su descomposición en sus tres canales del espacio HSV.

3.2.3. Los espacios XYZ, Luv, Lab.

Con las representaciones anteriores del espacio del color aparecen coeficientes negativos para colores de determinadas longitudes de onda, lo cual supone una fuente de error importante en los cálculos colorimétricos (comparación de un color monocromático con el blanco de referencia). La CIE (Commission Internationale de l´Eclairage) estableció un nuevo espacio de color (XYZ) con la finalidad de evitar los coeficientes negativos. En dicho espacio la ordenada Y (luminosidad) es perpendicular al plano definido por XZ (coloridad) de luminancia nula. Este espacio de color se caracteriza por ser independiente del dispositivo que se esté usando (Gevers y col., 2012).

En la mayoría de los casos se suele trabajar con valores normalizados entre cero y uno. Así, se tiene la expresión (3.12):

$$
x = \frac{X}{X+Y+Z} \qquad y = \frac{Y}{X+Y+Z} \qquad z = \frac{Z}{X+Y+Z} \tag{3.12}
$$

y para una determinada luminosidad Y, la expresión (3.13):

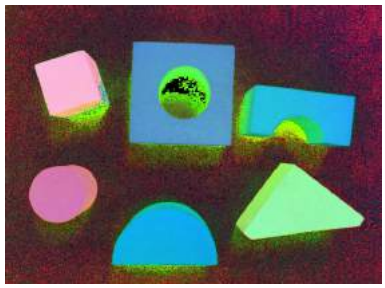
$$
X = \frac{xY}{y} \qquad Z = \frac{zY}{y} \tag{3.13}
$$

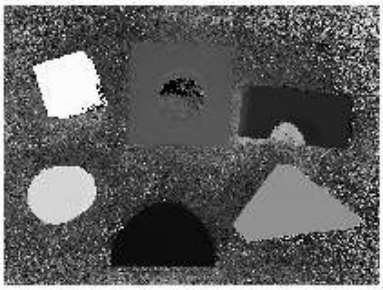
La relación matemática que permite el cambio de ejes entre el espacio de color RGB y XYZ se basa en la medición de las componentes RGB a través de un colorímetro, ecuación (3.14).

$$
\begin{cases}
X = \alpha R + \beta G + \gamma B \\
Y = \alpha' R + \beta' G + \gamma' B \\
Z = \alpha'' R + \beta'' G + \gamma'' B
\end{cases}
\tag{3.14}
$$

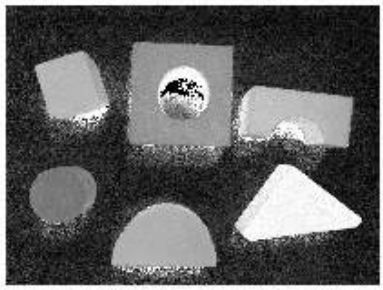
Ejemplo 3.2. *Visualización de una imagen en el espacio de color HSV y obtención de la descomposición en sus tres canales.*

Solución: Visualización de una imagen en el espacio de color HSV y su descomposición en los tres canales: tono, saturación y valor. El color viene representado por el primer canal (tono), que es el color inicial, la saturación es la concentración que tiene el color seleccionado, y el valor indica la tonalidad más o menos oscura. Así, la imagen del canal H muestra la tonalidad, la segunda imagen (canal S) muestra la saturación, y la tercera imagen el valor. El canal más importante es la saturación, debido a que indica la pureza de la tonalidad con respecto al blanco. Por ejemplo, el color verde sin blanco está saturado, si a continuación añadimos blanco, el color verde se transforma en un verde más pastel, es decir, la tonalidad continua siendo verde pero el color cambia a un verde más claro.

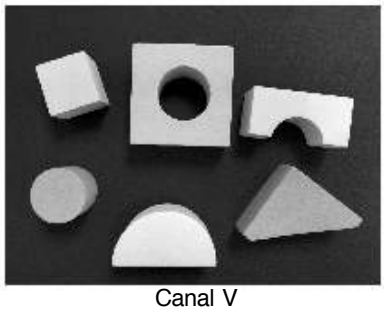




Canal H



Canal S



Canal V

Figura 3.5. Espacio de color HSV, y sus tres canales, tono (H), saturación (S) y valor (V)

Un colorímetro comprende dos campos de observación y diafragmas que dosifican las cantidades necesarias RGB para igualar visualmente el color (X) mediante tres primarios (R), (G), (B) bien elegidos.

Los coeficientes anteriores más comúnmente empleados son los propuestos por la ITU (International Telecommunications Union), que establecen la siguiente relación entre ambos espacios de color, ecuación (3.15):

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.431 & 0.342 & 0.178 \\ 0.222 & 0.707 & 0.071 \\ 0.020 & 0.130 & 0.939 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3.063 & -1.393 & -0.476 \\ -0.969 & 1.876 & 0.042 \\ 0.068 & -0.229 & 1.069 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3.15}$$

El espacio de color XYZ obtenido, constituye el denominado *"Diagrama Internacional xy, CIE"* o carta cromática *xy*. Una vez obtenidas las tres componentes primitivas del espacio de color XYZ, se pueden construir diferentes espacios CIES, entre ellos destacan el CIE (L*u*v*) y CIE (L*a*b*), en ambos la información representada es el tono o tinte, cromatismo e intensidad del color.

Considerando que X_n Y_n Z_n son los valores triestímulo CIE XYZ del punto de blanco de referencia, El espacio CIE (L*u*v*) se define a partir de las siguientes relaciones, donde de nuevo L es la información de la luminosidad y *uv* del color, ecuación (3.16):

$$\begin{aligned} L^* &= 116\left(\frac{Y}{Yn}\right)^{\frac{1}{3}} \qquad si \quad \frac{Y}{Yn} > 0.008856 \\ L^* &= 903.3\frac{Y}{Yn} \qquad si \quad \frac{Y}{Yn} < 0.008856 \\ u^* &= 13L^*(u' - u'_n) \\ v^* &= 13L^*(v' - v'_n) \end{aligned} \tag{3.16}$$

siendo:

$$u' = \frac{4X}{(X + 15Y + 3Z)}$$

$$v' = \frac{9Y}{(X + 15Y + 3Z)}$$

los coeficientes u´$_n$, v´$_n$ presentan las mismas ecuaciones que u', v', representando los valores para la referencia del blanco.

En el espacio de color CIE (L*a*b*), *L* es la luminosidad, *a* constituye la referencia respecto a la relación rojo/azul y *b* respecto a la relación amarillo/azul. Las ecuaciones que lo definen dependen del valor de la relación Y/Y$_n$ son:

Para valores $\frac{Y}{Yn} > 0.008856$, se emplean las expresiones (3.17),

$$\begin{aligned} L^* &= 116\left(\frac{Y}{Yn}\right)^{\frac{1}{3}} - 16 \\ a^* &= 500\left(\left(\frac{X}{Xn}\right)^{\frac{1}{3}} - \left(\frac{Y}{Yn}\right)^{\frac{1}{3}}\right) \\ b^* &= 200\left(\left(\frac{X}{Xn}\right)^{\frac{1}{3}} - \left(\frac{Z}{Zn}\right)^{\frac{1}{3}}\right) \end{aligned} \tag{3.17}$$

Mientras que si $\frac{Y}{Yn} \le 0.008856$, se tienen las expresiones (3.18);

$$L^* = 903.3\frac{Y}{Yn}$$
$$a^* = 7.87\left(\frac{X}{Xn} - \frac{Y}{Yn}\right) \tag{3.18}$$
$$b^* = 7.87\left(\frac{Y}{Yn} - \frac{Z}{Zn}\right)$$

El cromatismo en este espacio queda definido por la ecuación (3.19):

$$croma = \left[(a^*)^2 + (b^*)^2\right]^{1/2} \tag{3.19}$$

mientras que el tono o tinte se define a través de (3.20):

$$tono = arctan\left(\frac{b^*}{a^*}\right) \tag{3.20}$$

Ambos espacios CIES requieren una transformación intermedia al espacio de color XYZ a partir de la información RGB, realizando posteriormente una normalización o transformación asociada a raíces cúbicas, con la finalidad de obtener información similar a la captada por el ojo humano desde el punto de vista de su uniformidad.

A continuación, se muestra el Ejemplo 3.3 de conversión entre el espacio de color RGB y el XYZ.

3.2.4. Espacios YIQ, YUV, YCbCr, YCC.

Estos espacios se basan en la obtención de la luminancia (luminosidad) y la crominancia (color). Están muy relacionados con los sistemas de color para la televisión, YIQ para el NTSC, YUV para el PAL y YCbCr digital. Estos espacios de color son los sistemas utilizados en algunas cámaras de color.

A partir de la elección de los colores primarios y del nivel blanco de referencia, se puede obtener la expresión fundamental de la luminancia monocroma según la expresión (3.21):

$$Y = 0.30R + 0.59G + 0.11B \tag{3.21}$$

siendo R, G, B las señales de salida o componentes espectrales de la cámara de vídeo tricolor.

En cuanto a la crominancia, se han experimentado diversas formas de señal; inicialmente se trabajó con las señales R y B de forma aislada, pero al no satisfacer el criterio de que la información de color debe desaparecer cuando el color es acromático en la escena, y cuando se transmite en monocromo, se buscaron otras formas de señal, que son las llamadas señales de diferencia de color.

Ejemplo 3.3. *Visualización de imagen en espacio de color XYZ y obtención de su descomposición en los canales del espacio X, Y, Z*

Solución: A continuación se muestra una imagen, visualizada en espacio de color XYZ, y las tres imágenes que muestran su descomposición en tres canales: canal X, canal Y y canal Z, figura X.6. Las imágenes de los tres canales de color aparecen más oscuras que en los planos de color del espacio RGB, esto se debe a que cada píxel es representado en 16 bits en el canal X, Y, Z, y cada valor de píxel puede tomar un valor en el rango [0 - 65535].

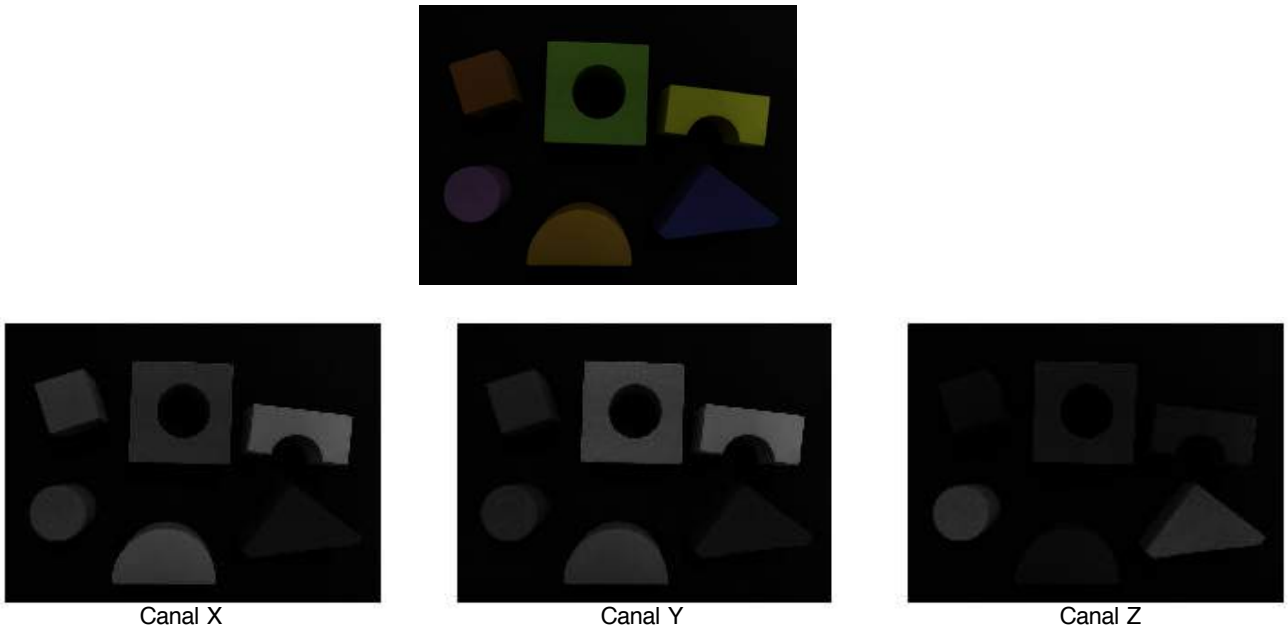


Figura 3.6. Visualización de imagen en espacio de color XYZ y descomposición en sus tres canales

Las señales simples de crominancia se obtienen al restar a las señales primarias proporcionadas por la cámara de color, la señal de luminancia Y. En total para transmitir la información de luminancia y crominancia se tienen cuatro señales, como cada color está definido por el espacio RGB, bastará con tres señales. La señal que se elimina es G-Y, considerando que se trata de una señal de baja relación, debido a sus menores coeficientes respecto a las señales R-Y y B-Y.

Las ecuaciones son las que aparecen en la ecuación (3.22):

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.22}$$

$$U=0.493\ (B - Y)$$
$$V= 0.877(R - Y)$$

La reducción en amplitud mostrada en las expresiones anteriores se debe a que si se modula una portadora directamente con las señales U=B-Y y V=R-Y, se produce una sobremodulación que provoca distorsión en los colores. La compresión anterior permite que la relación señal ruido sea la mejor posible, (un poco de la información de cromaticidad puede eliminarse sin una pérdida

significativa de la calidad de la imagen, ya que el ojo humando es menos sensitivo a los cambios de cromaticidad que a los de luminancia).

La transformación inversa viene dada por la ecuación (3.23):

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.000 & 1.140 \\ 1.000 & -0.396 & -0.581 \\ 1.000 & 2.029 & 0.000 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix} \tag{3.23}$$

Los sistemas europeos de TV (PAL y SECAM) usan el espacio YUV. La componente Y tiene un ancho de banda de 5 MHz y las componentes U y V de 2.5 MHz.

Experimentalmente se han encontrado otras dos señales de crominancia I, Q, que satisfacen mejor la reproducción de los colores. El sistema de referencia IQ forma un ángulo de 33º respecto al UV. Ambos sistemas se eligen ortogonales para facilitar su realización por medios electrónicos.

Las componentes YIQ definen el sistema de TV. La señal Y tiene un ancho de banda de 4.2 MHz, I 0.5 MHz y Q 1.5 MHz. Las ecuaciones que definen la transformación son las dadas en (3.24):

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.569 & -0.274 & -0.322 \\ 0.211 & -0.523 & -0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.24}$$

La trasformación inversa es la mostrada en la ecuación (3.25):

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.956 & 0.621 \\ 1.000 & -0.272 & -0.647 \\ 1.000 & -1.106 & -1.703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix} \tag{3.25}$$

Si se aplican las fórmulas clásicas de transformación de ejes de coordenadas, resultan las siguientes relaciones (3.26):

$$\begin{aligned} I&=-0.27(B - Y) + 0.74(R - Y) \\ Q&=0.41(B - Y) + 0.48(R - Y) \end{aligned} \tag{3.26}$$

Y por lo tanto la expresión (3.27):

$$\begin{aligned} I&=0.6\ R - 0.28\ G - 0.32\ B \\ Q&= 0.21\ R - 0.52\ G + 0.31\ B \end{aligned} \tag{3.27}$$

Recíprocamente se tiene la ecuación (3.28):

$$R\text{-}Y = 0.96\ I + 0.62\ Q \tag{3.28}$$

$$B\text{-}Y = -1.10\ I + 1.70\ Q$$

A continuación, se muestra un ejemplo del espacio de color YCbCr, ejemplo 3.4, donde el canal Y (representación en 8 bits [0 - 255]) indica la luminancia y los canales Cb y Cr indican el tono del color. De esta forma, Cb representa el color en una escala entre el azul y el amarillo, y Cr representa el color entre el rojo y el verde.

Ejemplo 3.4. *Visualización de una imagen en el espacio de color YCbCr y obtención de su descomposición en los canales Y Cb Cr*

Solución: La visualización se muestra en la siguiente Figura 3.7, donde aparece la imagen en color y las tres imágenes que muestran su descomposición en sus tres canales: canal Y, canal Cb y canal Cr.

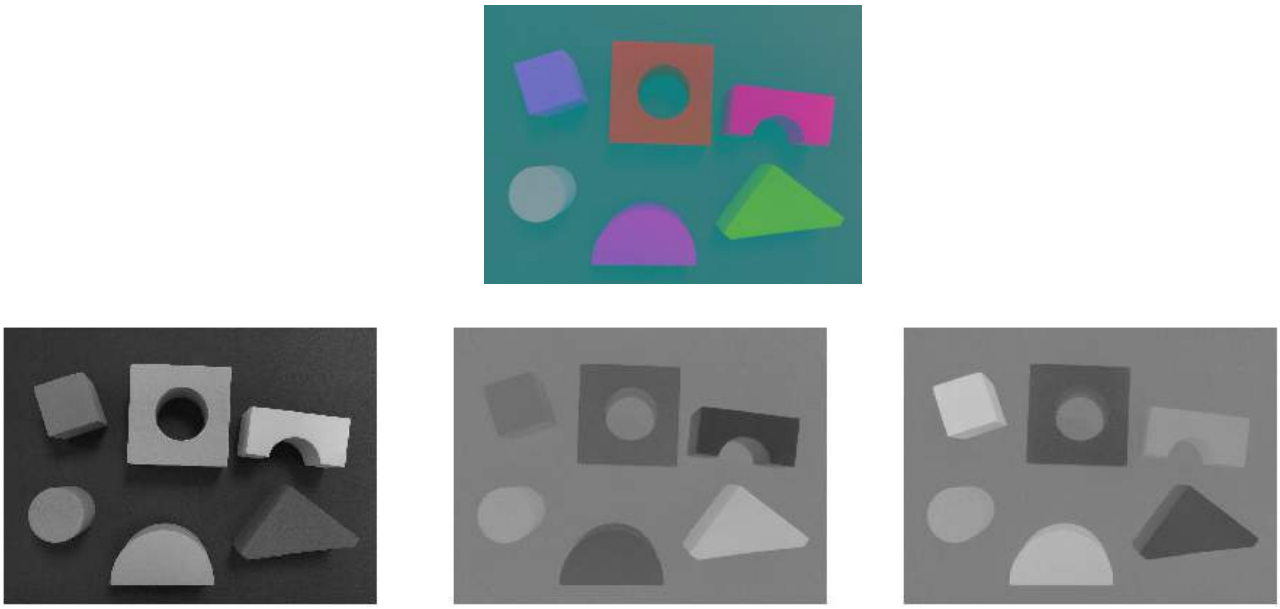


Figura 3.7. Espacio de color YCbCr y su descomposición en tres canales Y, Cb, Cr (8 bits)

3.3. Bibliografía

Forsyth, D (2012) *Computer vision: a modern approach,* Pearson, 2º Edición.Gevers, T.; Gijsenij, A.; van de Weijer, J.; Geusebroek, J.C. (2012) *Color in Computer Vision: Fundamentals and Applications*, John Wiley & Sons.

Szeliski, R. (2010) *Computer vision: algorithms and applications,* Springer.

Yoshida, S. R. (2010) *Computer Vision,* Elsevier.

Gonzalez R. C., Woods R. E. (2007) *Digital image processing,* Prentice Hall.

CAPÍTULO 4

OPERACIONES DE TRANSFORMACIÓN DE IMÁGENES

Pedro J. HERRERA[1], María GUIJARRO[2], José M. GUERRERO[3]

[1] Escuela Politécnica Superior, Universidad Francisco de Vitoria, Madrid, España
[2] Dpto. Arquitectura de Computadores y Automática, Facultad de Informática, Universidad Complutense, Madrid, España
[3] Dpto. Ingeniería del Software e Inteligencia Artificial, Facultad de Informática, Universidad Complutense, Madrid, España

La transformación de una imagen consiste en modificar el contenido de la misma con un objetivo concreto, como puede ser el de prepararla para un posterior análisis. Se pueden distinguir dos tipos de transformaciones: basadas en los niveles de intensidad de la imagen, o como consecuencia de la aplicación de una operación geométrica. Dentro del primer grupo se consideran por un lado las transformaciones que son consecuencia de la aplicación de una función sobre el valor de intensidad de cada píxel individualmente, y las transformaciones en las que los píxeles vecinos intervienen en la misma con distintas finalidades, tales como eliminar el ruido presente en la imagen para así suavizarla o con el fin de extraer bordes. Las transformaciones lógicas se aplican considerando que los valores numéricos de las imágenes se pueden representar a nivel de bits. Dentro del segundo grupo se incluyen las transformaciones geométricas, las cuales modifican las coordenadas espaciales de la imagen, ofreciendo aspectos de la misma bajo diferentes resoluciones, siendo en ocasiones necesario modificar también los valores de intensidad de los píxeles de la imagen original.

4.1. Transformaciones basadas en los niveles de intensidad

Las transformaciones en las que la generación de un nuevo píxel se obtiene en función del valor de intensidad, pueden clasificarse en dos tipos:

1. Operaciones basadas en un píxel individual de la imagen.

2. Operaciones involucrando píxeles vecinos.

Por tanto, la generación de un nuevo píxel dependerá bien del valor concreto de cada píxel, o del valor de los píxeles próximos a él, lo que se conoce como vecindad del píxel o vecindario, como muestra la figura 4.1.

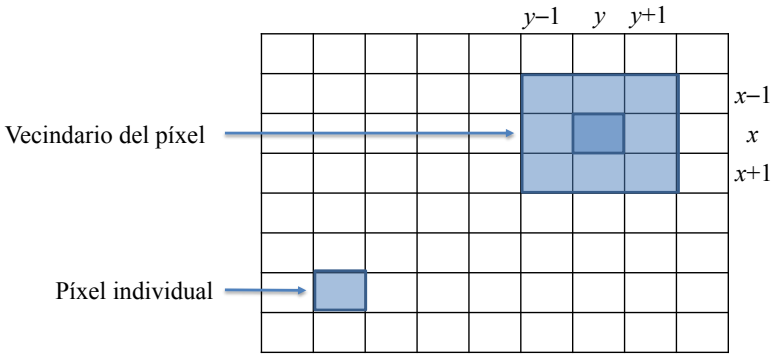


Figura 4.1. Las transformaciones básicas constan de operaciones centradas en un píxel concreto de la imagen y operaciones en las que interviene el entorno del píxel (vecindario).

4.1.1. Operaciones centradas en un píxel individual

Estas operaciones tienen la particularidad de transformar la imagen mediante la modificación uno a uno de los píxeles de la imagen; es decir, el valor de intensidad del píxel $S(i,j)$ de la imagen de salida es el resultado de aplicar una determinada transformación sobre el valor de intensidad del píxel $E(i,j)$ de la imagen original, tal y como se indica en la Figura 4.2. La imagen obtenida tendrá el mismo tamaño que la original.

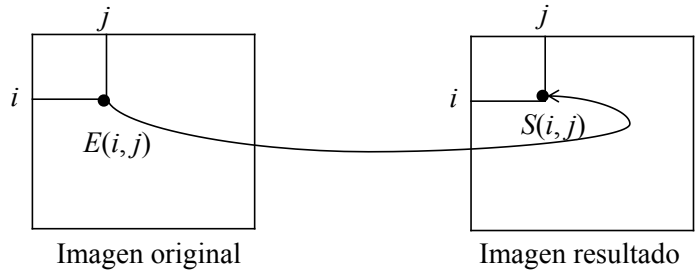


Figura 4.2. Transformación individual

La transformación se obtiene a partir de la ecuación (4.1), donde la operación *f* puede aplicarse sobre un píxel concreto de la imagen e incluso sobre una determinada región de la misma siempre considerando la intensidad o intensidades de los píxeles involucrados. Incidir en que, a diferencia de cómo se verá más adelante, la salida no depende de los píxeles adyacentes al píxel original.

$$S(i,j) = f(E(i,j)) \tag{4.1}$$

La función *f* aplicada variará dependiendo del operador escogido, y puede ser lineal o no lineal. A continuación se enumeran los principales operadores, definiendo las funciones de transformación que los caracterizan (Gonzalez y Woods, 2002; Pajares y Cruz, 2007),

1. *Operador identidad*. La aplicación de este operador proporciona una imagen equivalente a la original, tal y como muestra la figura 4.3. La función de transformación es la siguiente,

$$S(i,j) = E(i,j) \tag{4.2}$$

2. *Operador inverso*. Este operador calcula la inversa de la imagen original. La función de transformación para una imagen expresada en escala de grises (figura 4.3), esto es con valores de intensidad en el rango [0, 255], es tal y como muestra la figura 4.4(*a*) en base a la ecuación siguiente,

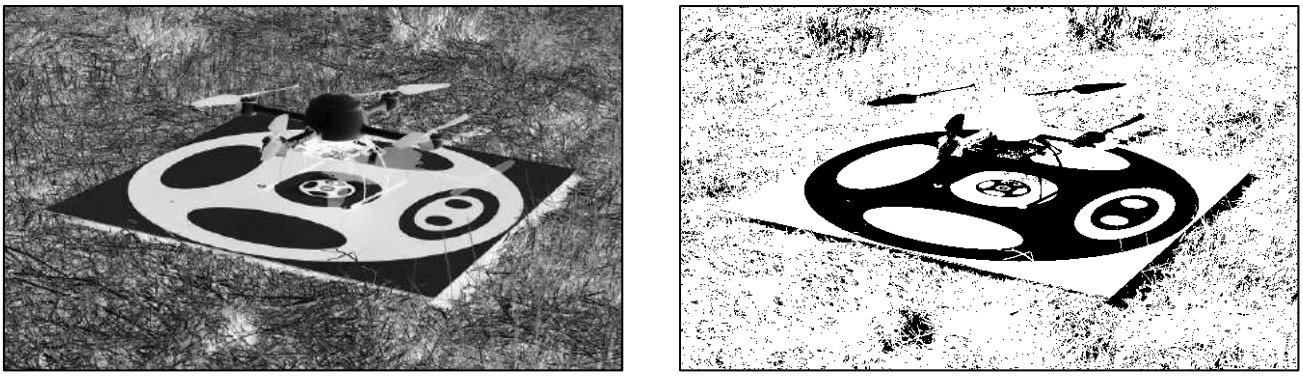
$$S(i,j) = 255 - E(i,j) \tag{4.3}$$

En la ecuación anterior, el valor 255 representa el máximo valor posible, que es el correspondiente a imágenes con representación de ocho bits en sus niveles de intensidad.

Figura 4.3. Imagen original. El operador identidad proporciona una imagen equivalente a la original.

3. *Operador umbral*. El operador umbral aplicado sobre la imagen original píxel a píxel proporciona como resultado una imagen binaria, es decir, una imagen con dos niveles de intensidad, figura 4.4(*b*),donde el valor de la variable *p*, conocido como umbral, hace las veces de discriminador que decide a cuál de los dos niveles corresponde cada valor de intensidad del píxel de la imagen original. Al igual que en el caso anterior, se asume que los valores de intensidad de la imagen original se encuentran en el intervalo [0, 255].

$$S(i,j) = \begin{cases} 0 & \text{si } E(i,j) \le p \\ 255 & \text{si } E(i,j) > p \end{cases} \tag{4.4}$$



(*a*) (*b*)

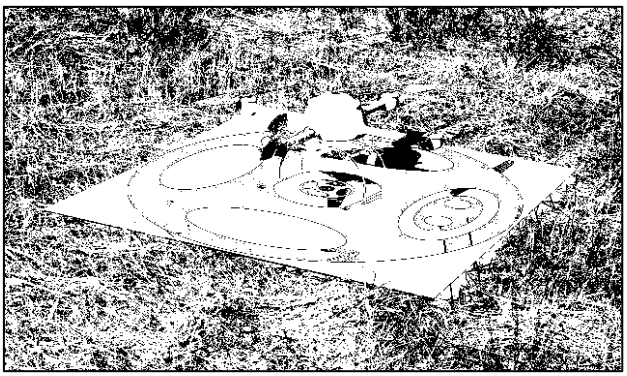
Figura 4.4.(*a*) Imagen inversa de la original generada a partir del operador inverso. (*b*) Imagen generada al aplicar el operador umbral con *p* = 90, sobre la imagen original.

4. *Operador intervalo de umbral binario*. La aplicación de este operador proporciona también una imagen binaria, figura 4.5(*a*), si bien la diferencia con el operador umbral estriba en que aquí se definen dos umbrales p_1 y p_2 de manera que si el valor de intensidad de un píxel de la imagen original se encuentra dentro del intervalo delimitado por p_1 y p_2, se le asigna el valor 0 en la imagen de salida, y 255 en caso contrario. La ecuación (4.5) define esta transformación.

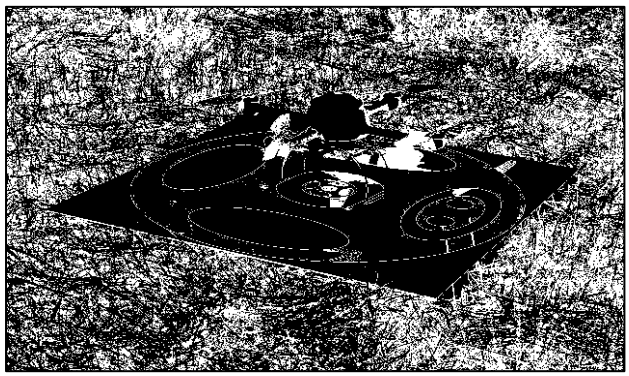
$$S(i,j)=\begin{cases} 0 & \text{si } p_1 < E(i,j) < p_2 \\ 255 & \text{si } E(i,j) \le p_1 \text{ o } E(i,j) \ge p_2 \end{cases} \tag{4.5}$$

Existe una variante de este operador conocida como *intervalo de umbral binario invertido*, en la que si el valor de intensidad de un píxel de la imagen de partida se encuentra dentro del intervalo delimitado por p_1 y p_2, se le asigna el valor 255 en la imagen de salida, y 0 en caso contrario. La figura 4.5(*b*) muestra esta situación tras aplicar la ecuación (4.6) sobre los píxeles de la imagen original.

$$S(i,j)=\begin{cases} 255 & \text{si } p_1 < E(i,j) < p_2 \\ 0 & \text{si } E(i,j) \le p_1 \text{ o } E(i,j) \ge p_2 \end{cases} \tag{4.6}$$



(*a*)



(*b*)

Figura 4.5.(*a*) Imagen generada tras aplicar el operador intervalo de umbral binario sobre la imagen original, con p_1 = 50 y p_2 = 150. (*b*) Imagen generada tras aplicar el operador intervalo de umbral binario invertido sobre la imagen original, con los mismos valores de p_1 y p_2 que en el caso anterior.

5. *Operador umbral de la escala de grises*. La aplicación de este operador sobre la imagen original genera como resultado una imagen en la que únicamente se preservan los valores de intensidad de aquellos píxeles comprendidos dentro del intervalo definido por p_1 y p_2. En caso contrario, el valor de salida será 255. La figura 4.6(*a*) muestra un ejemplo ilustrativo.

$$S(i,j)=\begin{cases} E(i,j) & \text{si } p_1 < E(i,j) < p_2 \\ 255 & \text{si } E(i,j) \le p_1 \text{ o } E(i,j) \ge p_2 \end{cases} \tag{4.7}$$

Al igual que ocurría con el operador intervalo de umbral binario, existe una variante de este operador conocida como *umbral de la escala de grises invertido*, que transforma la imagen de

partida de manera que los píxeles con intensidades entre p_1 y p_2 son invertidos, y el resto toma el valor 255. La figura 4.6(*b*) muestra esta variante.

$$S(i,j)=\begin{cases}255-E(i,j) & \text{si } \; p_1 < E(i,j) < p_2 \\ 255 & \text{si } \; E(i,j) \le p_1 \text{ o } E(i,j) \ge p_2\end{cases} \tag{4.8}$$

(*a*) (*b*)

Figura 4.6.(*a*) Imagen generada tras aplicar el operador umbral de la escala de grises sobre la imagen original, con p_1 = 1 y p_2 = 50. (*b*) Imagen generada tras aplicar el operador umbral de la escala de grises invertido sobre la imagen original, con los mismos valores de p_1 y p_2 que en el caso anterior.

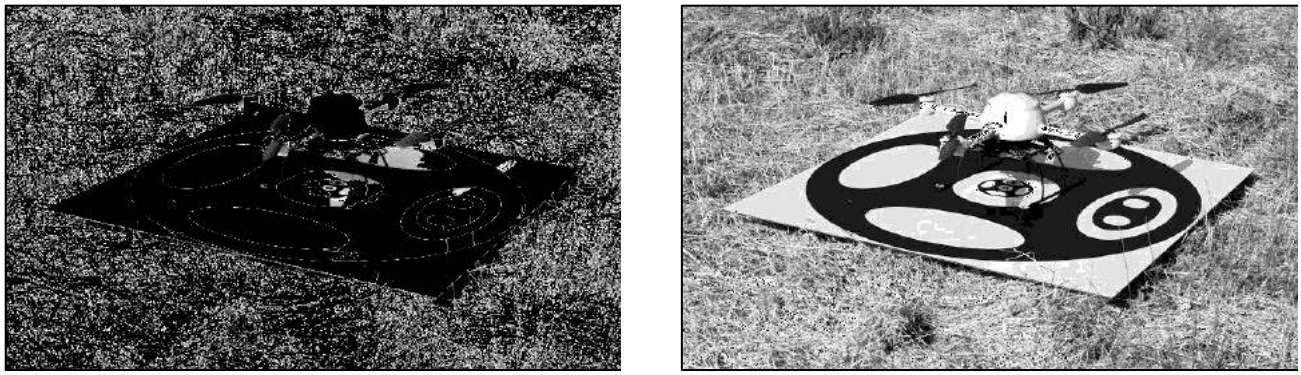
6. *Operador de extensión*. Proporciona una imagen con la escala de grises completa para los píxeles cuyo nivel de intensidad se encuentre comprendido dentro del intervalo abierto (p_1, p_2). La figura 4.7(*a*) muestra un ejemplo ilustrativo tras aplicar sobre la imagen original la ecuación siguiente,

$$S(i,j)=\begin{cases}(E(i,j)-p_1)\dfrac{255}{p_2-p_1} & \text{si } \; p_1 < E(i,j) < p_2 \\ 0 & \text{si } \; E(i,j) \le p_1 \text{ o } E(i,j) \ge p_2\end{cases} \tag{4.9}$$

7. *Operador reducción del nivel de gris*. Como su propio nombre indica, este operador reduce el número de niveles de intensidad de la imagen original, en base a la siguiente ecuación,

$$S(i,j)=\begin{cases}0 & \text{si } \; E(i,j) \le p_1 \\ S_1(i,j) & \text{si } \; p_1 < E(i,j) \le p_2 \\ S_2(i,j) & \text{si } \; p_2 < E(i,j) \le p_3 \\ \ldots & \\ S_n(i,j) & \text{si } \; p_{n-1} < E(i,j) \le 255\end{cases} \tag{4.10}$$

La figura 4.7(*b*) ilustra a modo de ejemplo, la aplicación del operador reducción del nivel de intensidad sobre la imagen original mostrada en la figura 4.3. En este ejemplo se ha reducido el número de niveles de intensidad de 256 a 10.



(*a*) (*b*)

Figura 4.7.(*a*) Imagen generada tras aplicar el operador de extensión sobre la imagen original, con p_1 = 50 y p_2 = 150. (*b*) Imagen generada tras aplicar el operador reducción con diez niveles de gris y p_1 = 25, p_2 = 50, p_3 = 75, p_4 = 100, p_5 = 125, p_6 = 150, p_7 =175, p_8 = 200, p_9 = 225 y p_{10} = 255.

Como se ha descrito hasta aquí, los anteriores operadores ofrecen la posibilidad de transformar una imagen de partida tras la aplicación de una transformación basada en una función que puede ser bien lineal o no lineal. Cabe la posibilidad también de generar una nueva imagen como resultado de aplicar una transformación sobre dos imágenes, considerando dos píxeles uno de cada imagen y ubicados en la misma posición espacial en sendas imágenes, tal y como refleja el esquema de la figura 4.8. El tamaño de las dos imágenes ha de ser el mismo y la función utilizada, sea lineal o no, se aplicará a todos los pares de píxeles de las dos imágenes de partida.

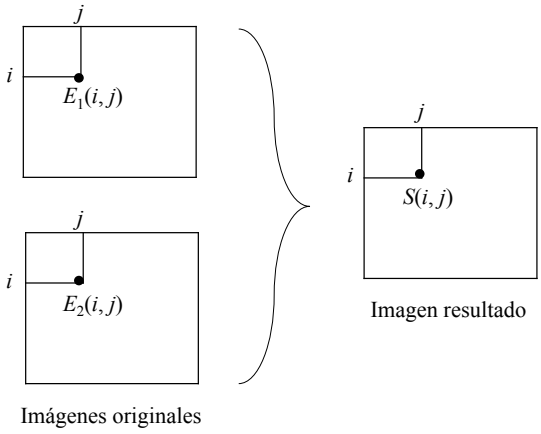


Figura 4.8. Transformación píxel a píxel de dos imágenes en una.

Se pueden aplicar diferentes funciones tales como la *adición* y *substracción*. La operación de adición queda definida por la ecuación (4.11), donde *k* toma el valor 2 para el caso de partir de dos imágenes de entrada. Si la suma implicase más de dos imágenes, *k* tomaría el valor del número de imágenes utilizadas. Esta operación es útil para reducir los efectos provocados en la imagen por el ruido, ya que el resultado resulta ser el valor medio de los píxeles de entrada.

$$S(i,j) = \frac{(E_1(i,j) + E_2(i,j))}{k} \tag{4.11}$$

La operación de substracción, definida por la ecuación (4.12) depende también de la variable *k* que cumple la misma función que en el caso anterior. Esta operación es utilizada con el propósito de detectar cambios producidos en las imágenes como consecuencia de haber sido tomadas en instantes de tiempo diferentes. El movimiento es uno de tales cambios. El símbolo $|\cdot|$ indica que tras aplicar la operación de substracción sobre los valores de intensidad de los píxeles ubicados en las mismas posiciones (*i*, *j*) en las imágenes de partida, se calcula el valor absoluto para no obtener valores negativos en el resultado con vistas a su visualización, dado que el rango de valores intensidad está comprendido en el intervalo [0, 255].

$$S(i,j) = k\left|E_1(i,j) - E_2(i,j)\right| \tag{4.12}$$

4.1.2. Transformaciones de vecindad

La principal diferencia entre las transformaciones de intensidad estudiadas en el apartado anterior, y las transformaciones de vecindad es que en éstas el valor del píxel de salida se obtiene tras realizar sobre el píxel de origen una combinación de los valores de los píxeles vecinos. Por tanto, la transformación de la imagen se produce por la combinación de píxeles, en lugar de realizar una transformación píxel a píxel.

Con carácter general, el vecindario de un píxel lo componen ocho valores, correspondientes a las posiciones alrededor del píxel, tal y como refleja la figura 4.1 y la siguiente ecuación,

$$Vecinos_{E(x,y)} = \begin{Bmatrix} E(x-1,y-1), E(x-1,y), E(x-1,y+1), E(x,y+1), \\ E(x+1,y+1), E(x+1,y), E(x+1,y-1), E(x,y-1) \end{Bmatrix} \tag{4.13}$$

De manera que el valor de intensidad del píxel de salida *S*(*x*,*y*) es la suma promediada de los valores de intensidad de los ocho vecinos alrededor del píxel de entrada *E*(*x*, *y*). La transformación de una imagen puede variar dependiendo de la influencia que cada uno de los vecinos ejerza en el promedio sobre el resultado. Esto se consigue mediante una *máscara* que permite escoger de manera selectiva los vecinos que intervienen en la transformación, y en qué medida contribuyen a la modificación del píxel central.

$$M1_{3x3} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}; \quad M2_{3x3} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}; \quad M3_{3x3} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}; \tag{4.14}$$

Si se aplica la máscara *M*1 sobre un píxel concreto de la imagen, todos los vecinos influirían en el resultado en la misma medida, tal y como indica la siguiente ecuación,

$$\begin{aligned} S(x,y)_{M1} = {} & E(x-1,y-1) + E(x-1,y) + E(x-1,y+1) \\ & + E(x,y-1) + E(x,y) + E(x,y+1) \\ & + E(x+1,y-1) + E(x+1,y) + E(x+1,y+1) \end{aligned} \tag{4.15}$$

Sin embargo, si se aplica la máscara *M*2, se observa que unos píxeles vecinos influyen en la transformación frente a otros que no lo hacen, e incluso ejerciendo distintas influencias.

$$\begin{aligned} S(x,y)_{M2} = & 1\cdot E(x-1,y-1)+2\cdot E(x-1,y)+1\cdot E(x-1,y+1) \\ & +0\cdot E(x,y-1)+0\cdot E(x,y)+0\cdot E(x,y+1) \\ & -1\cdot E(x+1,y-1)-2\cdot E(x+1,y)-1\cdot E(x+1,y+1) \end{aligned} \tag{4.16}$$

4.2. Transformaciones lógicas

Resulta ya suficientemente conocido el hecho de que una imagen digital en gris no es ni más ni menos que una matriz de números (Gonzalez y Woods, 2002; Pajares y col., 2003; Pajares y Cruz, 2007). Bajo esta consideración se pueden aplicar las mismas operaciones aritméticas que sobre matrices numéricas, con las limitaciones derivadas del hecho de que los valores de intensidad sólo toman valores en el rango establecido, generalmente en el intervalo [0 255],en este caso por su representación de ocho bits.

Dado que la representación interna de los datos en un ordenador se realiza finalmente mediante una representación binaria, es posible aplicar operaciones lógicas binarias sobre estos datos. Las operaciones lógicas habituales son: *and*, *or*, *xor*, *not* y derivadas.

En la sección 4.1 se estudiaron diversas técnicas de transformación de imágenes píxel a píxel. Una de ellas era el resultado de aplicar el operador umbral, donde una imagen en escala de grises se transformaba en una imagen binaria según un determinado valor umbral: los píxeles cuyo valor de intensidad no superaba el valor umbral se transformaban en el valor 0, y en caso contrario tomaban el valor 255. Por tanto, se obtenía una imagen con dos únicos valores posibles (0 y 255).

Trasladándose al dominio de la lógica binaria matemática, pueden verse las imágenes binarias anteriores formadas únicamente por valores lógicos "0" y "1", donde el "0" lógico equivaldría al nivel de intensidad 0, y el "1" lógico al nivel de intensidad 255. Por tanto, es posible aplicar operaciones lógicas sobre la imagen o imágenes derivadas de la representación binaria de los datos que contienen. Un sencillo ejemplo ilustrará esta idea: considerando dos imágenes binarias de tamaño 5*x*5 representadas en formato matricial, tal y como muestra la figura 4.9(*a*) y (*b*), la figura 4.9(*c*) muestra el resultado de aplicar la operación lógica *and* píxel a píxel.

0	1	1	1	0
0	1	0	1	1
1	0	1	0	1
1	0	0	1	0
0	1	1	1	0

(*a*)

0	1	0	0	1
0	0	1	1	1
1	0	1	0	1
1	1	1	1	0
0	1	1	0	1

(*b*)

0	1	0	0	0
0	0	0	1	1
1	0	1	0	1
1	0	0	1	0
0	1	1	0	0

(*c*)

Figura 4.9.(*a*) y (*b*) Imágenes binarias representadas en formato matricial.(*c*) Imagen binaria resultado de aplicar la operación lógica *and* píxel a píxel sobre las dos matrices anteriores.

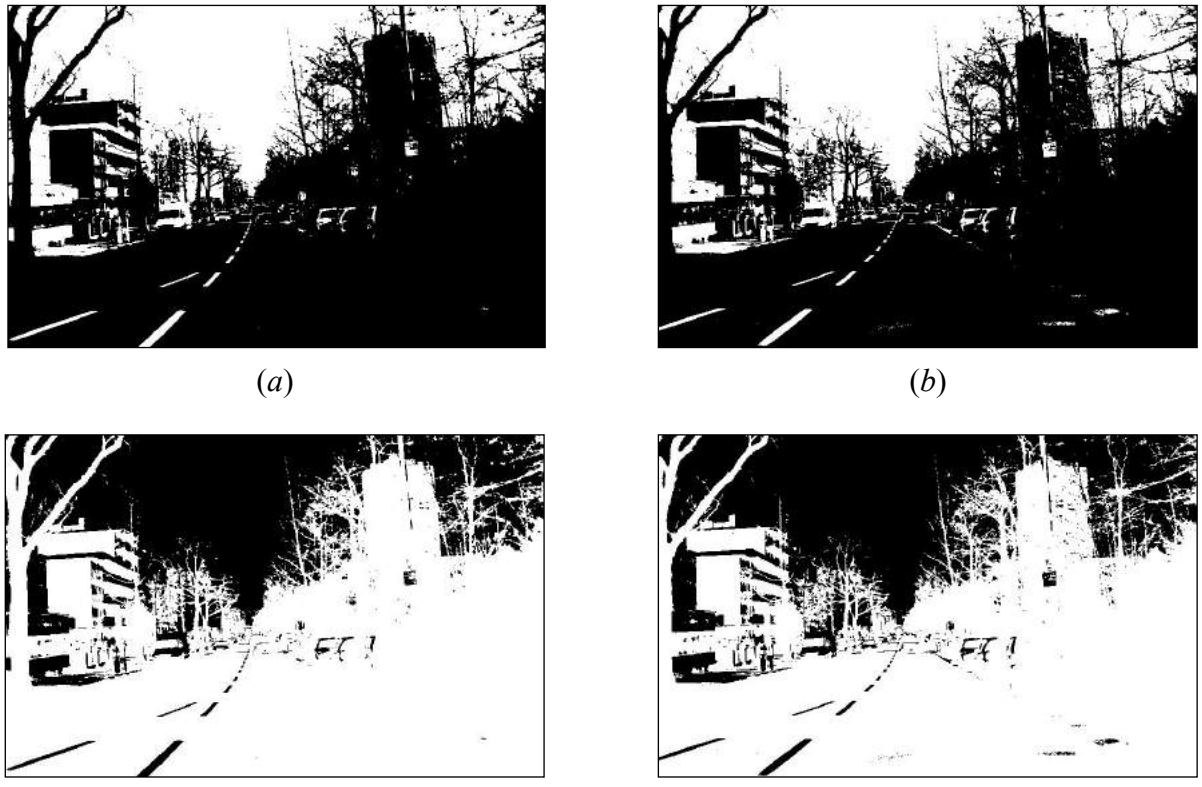
Tomando como referencia las dos imágenes estereoscópicas mostradas en las figuras 4.10(*a*) y (*b*), donde la misma escena ha sido capturada con dos cámaras ligeramente desplazadas horizontalmente

una respecto de la otra, a continuación se muestra el resultado de aplicar las operaciones lógicas mencionadas anteriormente, ilustrando así el efecto que dichos operadores generan.

(*a*) (*b*)

Figura 4.10.(*a*) Imagen estereoscópica izquierda. (*b*)Imagen estereoscópica derecha. Ambas imágenes son similares salvo por el ligero desplazamiento horizontal que se aprecia entre las dos, debido al desplazamiento de las cámaras con las que han sido capturadas.

La figura 4.11 muestra el resultado de aplicar las operaciones lógicas *not*, *and*, *or* y *xor*. Para ello, previamente se ha realizado un binarizado de las imágenes originales, tal y como se ha explicado anteriormente, aplicando la ecuación (4.4), con $p = 100$.



(*a*) (*b*)

(*c*) (*d*)

(*e*)

(*f*)

(*g*)

(*h*)

Figura 4.11.(*a*) y (*b*) Imágenes binarias con *p* = 100.(*c*) y (*d*) Resultado de aplicar la negación lógica a (*a*) y (*b*) respectivamente. (*e*), (*f*) y (*g*) Resultado de aplicar las operaciones lógicas *or*, *and* y *xor* respectivamente sobre (*c*) y (*d*). (*h*) Resultado de aplicar la operación $\leq$ sobre las imágenes originales en la figura 4.10(*a*) y (*b*), respectivamente.

Dadas dos imágenes, es posible aplicar otro tipo de operaciones relacionales tales como $<, >, \leq$ y $\geq$. Por ejemplo, supónganse las imágenesI_1 e I_2, una imagen $R = I_1 \leq I_2$ se obtendrá por comparación píxel a píxel de los valores de las imágenes de entrada, de manera que un píxel de la imagen resultado R valdrá 1 si cumple la relación y 0 si no la cumple. La figura 4.11(*h*) muestra el resultado de aplicar la relación $\leq$ sobre las dos imágenes originales en la figura 4.10.

4.3. Transformaciones geométricas

Hasta el momento, se ha estudiado la transformación del contenido de las imágenes mediante la modificación de los valores de intensidad de los píxeles. En este apartado se presenta un enfoque diferente, tratando la manipulación de la imagen como un cambio en la posición de los píxeles en lugar de la modificación de su valor. Este tipo de transformaciones supone un cambio en la distribución de los píxeles respecto de un sistema de coordenadas, de manera que el resultado es una transformación geométrica de la imagen original (De la Escalera, 2001; Pajares y col., 2003; Pajares y Cruz, 2007).Aunque también es cierto que algunas de estas transformaciones implican necesariamente la modificación de los valores de intensidad como consecuencia de dicha transformación.

Con frecuencia, para el análisis de una imagen es preciso considerar más concretamente un área dentro de la propia imagen que recibe el nombre de *Región de Interés* (en terminología inglesa *Region Of Interest* - ROI). Para ello, es necesario disponer de operaciones geométricas que modifiquen las

coordenadas espaciales de la imagen. Una operación geométrica tiene como objetivo principal transformar los valores de una imagen tal y como podrían observarse desde otro punto de vista. De esta manera, operaciones como magnificar o reducir una imagen no son más que aproximar o alejar el punto de vista, realizar una rotación equivale a girar el punto de vista, y una traslación equivale a hacer lo propio con dicho punto.

El paso previo a toda operación geométrica consiste en observar la distribución espacial de los píxeles en la imagen original y en la transformada. En cualquier caso, tanto la imagen original como la transformada asumen una estructura matricial, tal y como ilustra la figura 4.12.

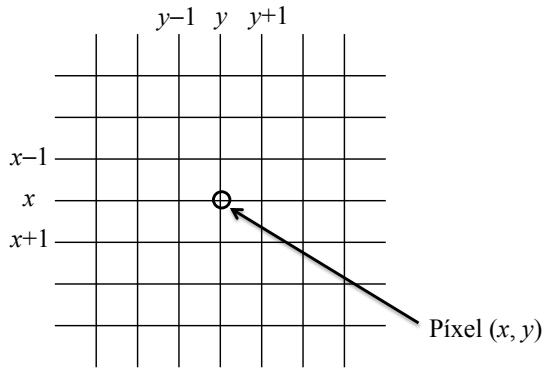


Figura 4.12. Distribución de los píxeles de una imagen en formato matricial o rejilla.

En una imagen digital no existen valores de intensidad entre los valores de las coordenadas *x* e *y*, que coinciden con las intersecciones de los valores en la horizontal y la vertical. Esto se debe a que la imagen digital es de naturaleza discreta. Al producirse una transformación en la rejilla, ya sea por un desplazamiento, un giro o una aproximación, los nuevos píxeles ya no tienen por qué coincidir con intersecciones y por lo general quedarán situados sobre puntos intermedios, tal y como refleja la figura 4.13, donde la rejilla continua muestra la disposición convencional de los píxeles de la imagen original, mientras que la rejilla discontinua refleja cómo quedaría la rejilla original después de girarla un ángulo determinado. Dado que los píxeles deben proyectarse sobre los de la imagen final, con una estructura similar a la de la imagen original, es necesario convertir estas coordenadas, y además obtener los valores de los píxeles finales en función de los transformados.

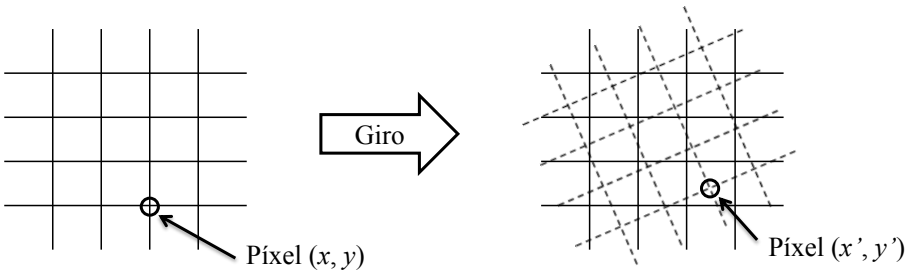


Figura 4.13. Una transformación geométrica suele provocar que los píxeles de la rejilla transformada no coincidan con los píxeles de la rejilla final.

Para ello, por un lado es preciso determinar las coordenadas de cada píxel (*x*,*y*) en la rejilla transformada (en líneas discontinuas), tal y como refleja la figura 4.13. En general, los píxeles (*x'*, *y'*) obtenidos tras la transformación no serán valores enteros y por tanto, no coincidirán con píxeles de la

rejilla destino. Por otro lado, es necesario calcular los valores de los píxeles (x',y') en la rejilla destino a partir de los valores conocidos de píxeles (x,y) entre ellos.

El primer paso depende de la transformación a realizar, mientras que el segundo corresponde a una operación de interpolación. Una vez se ha aplicado una transformación geométrica, es preciso obtener los valores de intensidad asociados con la transformación realizada, de forma que la imagen original aparezca transformada geométricamente pero con los valores de intensidad obtenidos a partir de los correspondientes en la imagen original. Es en este paso donde se genera la modificación de los valores de intensidad de la imagen original.

En esta sección se estudiará en primer lugar el concepto de interpolación y la manera de calcular los valores de intensidad asociados a una transformación geométrica, para después ver diferentes tipos de funciones de transformación.

4.3.1. Interpolación

La interpolación puede verse como el cálculo del valor de intensidad de un píxel, en una posición cualquiera, como una función de los píxeles que le rodean (ocupando las posiciones enteras de la rejilla destino). Si por ejemplo, se desea calcular el valor del píxel de coordenadas (x, y) mostrado en la figura 4.14, a partir de los valores de los píxeles de la rejilla (i, j), entonces la formalización matemática de la interpolación puede expresarse de la siguiente forma.

$$p(x,y) = \sum_{i=-n}^{n} \sum_{j=-m}^{m} p(i,j) h(x-i, y-j) \tag{4.17}$$

donde $p(i,j)$ es el valor de un píxel de la imagen original en la posición espacial (i,j), $p(x,y)$ es el valor del píxel en la imagen resultante de la interpolación a partir de $p(i,j)$, y $h(x,y)$ es el núcleo de interpolación. Existen diferentes núcleos de interpolación, de entre los cuales los más utilizados en el tratamiento geométrico de imágenes son: *vecino más próximo*, *bilineal* y *bicúbico*.

4.3.1.1. Interpolación por vecino más próximo

Consiste en suponer que el píxel a ser interpolado, toma el mismo valor que el del píxel más cercano de entre los cuatro que le rodean. Para decidir cuál es el píxel más cercano, se suele utilizar la distancia euclídea. En la figura 4.14 tomaría el valor del píxel $p(i,j)$. Otra opción consiste en asignarle la intensidad media asociada a los dos píxeles más cercanos, uno en la dirección horizontal x y el otro en la dirección vertical y. En este caso, en la figura 4.14tomaría el valor medio entre $p(i,j)$ y $p(i+1,j)$.

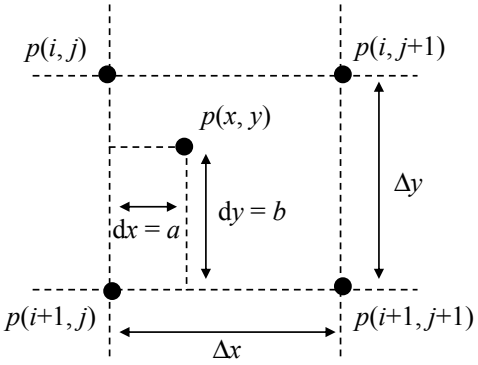


Figura 4.14. Esquema de la interpolación.

Por otra parte, el núcleo de interpolación $h(x,y)$ se define como sigue:

$$h(x)=\begin{cases}1 & \text{si } 0<|x|<0.5\\ 0 & \text{de otro modo}\end{cases} \tag{4.18}$$

4.3.1.2. Interpolación bilineal

Otra forma de realizar la interpolación con mejores resultados que los obtenidos con la interpolación por vecino más próximo, si bien a costa de una mayor carga computacional es la interpolación bilineal, la cual asigna a cada píxel objeto de la transformación un valor medio ponderado de las intensidades de los cuatro píxeles que le rodean.

El valor de intensidad del píxel interpolado, en función de los cuatro valores de intensidad de los píxeles de su entorno, se calcula como sigue.

$$p(x,y)=a_1p(i,j)+a_2p(i,j+1)+a_3p(i+1,j)+a_4p(i+1,j+1) \tag{4.19}$$

donde los factores de ponderación (a_1 , a_2 , a_3 y a_4) vienen dados por la distancia entre cada píxel y los de su entorno. Los factores de ponderación se calculan de la siguiente manera en base a la figura 4.14.

$$a_1=(1-dx)(1-dy);\quad a_2=dx(1-dy);\quad a_3=(1-dx)dy;\quad a_4=dxdy \tag{4.20}$$

En la interpolación bilineal, el núcleo de la interpolación $h(x, y)$ se define como sigue,

$$h(x)=\begin{cases}1-|x| & \text{si } 0<|x|<1\\ 0 & \text{de otro modo}\end{cases} \tag{4.21}$$

4.3.1.3. Interpolación bicúbica

Una tercera forma realizar la interpolación con mejores resultados que la interpolación bilineal es la interpolación bicúbica. En este caso, intervienen 16 puntos vecinos del píxel que se está interpolando. La función de interpolación en este caso viene definida por la siguiente expresión,

$$h(x)=\begin{cases}1-2|x|^2+|x|^3 & \text{si } 0<|x|<1\\ 4-8|x|+5|x|^2-|x|^3 & \text{si } 1<|x|<2\\ 0 & \text{de otro modo}\end{cases} \tag{4.22}$$

4.3.2. Transformaciones elementales

En esta sección se estudian tres tipos de transformaciones elementales: *traslación*, *rotación* y *escalado*, desde el punto de vista de su aplicación al tratamiento de imágenes, y por tanto, en base al sistema de coordenadas bidimensional asociado a la imagen, como es habitual. Independientemente de la transformación elemental aplicada, será necesario realizar una interpolación a la imagen resultante. Para mayor claridad, se supone que la imagen original tiene coordenadas (i, j) mientras que la imagen resultado posee coordenadas (x, y).

4.3.2.1. Traslación

La traslación o desplazamiento consiste en trasladar el píxel (i, j) en la imagen original hasta que alcance la posición $(i+i_d, j+j_d)$ y sustituir el valor de la intensidad en $(i+i_d, j+j_d)$ por el correspondiente a la posición (i, j) de la imagen original, obteniendo finalmente el valor del píxel en las coordenadas (x, y) para la imagen transformada.

En ocasiones será necesario realizar algún tipo de interpolación ya que los desplazamientos pueden no ser enteros. La traslación de la imagen original viene dada por la siguiente transformación,

$$\begin{aligned} x &= i + i_d \\ y &= j + j_d \end{aligned} \tag{4.23}$$

4.3.2.2. Rotación

La rotación difiere de la traslación en que la transformación es un giro. La rotación de un ángulo θ con respecto al origen de coordenadas de la imagen viene dada por la siguiente transformación,

$$\begin{aligned} x &= \cos\theta i - sen\theta j \\ y &= sen\theta i + \cos\theta j \end{aligned} \tag{4.24}$$

La aplicación más importante de la rotación en una imagen es para simular determinadas situaciones de giro de la cámara de captura de la escena, o el giro del propio objeto en situaciones de movimiento. Otro uso habitual de esta transformación es para lograr efectos estéticos.

Los parámetros necesarios para llevar a cabo una rotación son el ángulo de giro y las coordenadas del centro de rotación. La figura 4.15 muestra un ejemplo donde la imagen original mostrada en la figura 4.3 aparece ahora rotada con un ángulo $\theta = -15^{\circ}$ alrededor de su centro,

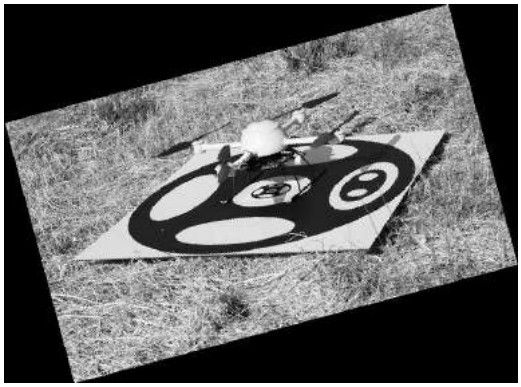


Figura 4.15. Imagen rotada -15^{o} (345^{o}) sobre su centro.

4.3.2.3. Escalado

La transformación de escalado consiste en variar el tamaño de la imagen original. Dicha variación puede realizarse a lo largo de cualquiera de los ejes de coordenadas *x* e *y*. De esta forma, se representa mediante sendos factores de escala S_x y S_y en las direcciones *x* e *y*, respectivamente. Cuando el factor toma valores entre 0 y 1, se produce una reducción de la propia imagen mientras que cuando son mayores que la unidad, el resultado es un aumento. Esta transformación viene dada por las siguientes expresiones,

$$x = S_x i, \quad y = S_y j \tag{4.25}$$

Uno de los efectos más relevantes del escalado es el *zoom* cuyo proceso se realiza como sigue: seleccionar una zona de la imagen (subimagen), separarla del resto de la imagen original y realizar una expansión sobre la zona seleccionada. El proceso de expansión puede realizarse de distintas maneras, siendo los más sencillos a la par que intuitivos los siguientes:

1. Repetir los valores de los píxeles previos, para crear un efecto de bloques.
2. Utilizar interpolación lineal: calcular el valor medio entre dos píxeles y usar dicho valor como el valor del píxel entre los dos.

El segundo proceso se puede realizar en dos pasos, primero por filas, y después por columnas, tal y como refleja la figura4.16,

10	6	10
6	10	6
10	4	10

(*a*)

10	8	6	8	10
6	8	10	8	6
10	7	4	7	10

(*b*)

10	8	6	8	10
8	8	8	8	8
6	8	10	8	6
8	7,5	7	7,5	8
10	7	4	7	10

(*c*)

Figura 4.16.(*a*) Imagen original. (*b*)Imagen con las filas expandidas. (*c*)Imagen con filas y columnas expandidas.

Este método permite ampliar una imagen de dimensión *NxN* a otra de dimensión $(2N-1)$ *x* $(2N-1)$ pudiéndose repetir tantas veces como se desee.

4.4. Bibliografía

De la Escalera, A. (2001) *Visión por computador: fundamentos y métodos*; Prentice Hall, Madrid, España.

Gonzalez, R.C., Woods, R.E. (2002) *Digital Image Processing*, 2nd ed.; Prentice Hall, Upper SaddleRiver, New Jersey, USA.

Pajares, G., de la Cruz, J.M. (2007) *Visión por Computador: imágenes digitales y aplicaciones*, 2ª ed.;RA-MA, Madrid, España.

Pajares, G., de la Cruz, J.M., Molina, J.M., Cuadrado, J., López, A. (2003) *Imágenes Digitales: procesamiento práctico con JAVA*; RA-MA, Madrid, España.

CAPÍTULO 5

PROCESAMIENTO MORFOLÓGICO

José A. CANCELAS[1], Rafael C. GONZÁLEZ[1], Ignacio ÁLVAREZ[1] , José M. ENGUITA[1]

[1] Profesores Titulares de Universidad: Departamento de Ingeniería Eléctrica, Electrónica de Computadores y de Sistemas, Universidad de Oviedo, Gijón, España

El procesamiento morfológico es una técnica de la visión por computador que analiza las imágenes basándose en propiedades de la forma, empleando álgebra de conjuntos. Estos métodos pueden ser aplicados tanto a imágenes binarias, como a imágenes en niveles de gris y de color.

La utilidad de estos métodos es muy variada, pudiendo emplearse tanto para el preprocesamiento, con operaciones para la supresión de ruido o el realce de objetos, como para el análisis de imágenes, extrayendo características tales como el esqueleto o definición del contorno. También se emplean en la búsqueda de patrones dentro de una imagen. Los temas que se tratarán son:

- Operaciones básicas con conjuntos.
- Definición de elemento estructurante.
- Principales operaciones con imágenes binarias: erosión, dilatación, apertura y cierre.
- Transformaciones Acierta-Falla (Hit or Miss)
- Esqueletonización, rellenado y reconstrucción de objetos.
- Introducción a la morfología con imágenes de niveles de gris.
- Operaciones Top-Hat y Bottom-Hat
- Ejemplos y ejercicios propuestos.

5.1. Introducción

La morfología matemática (MM) es una técnica para el procesamiento de imágenes que puede ser utilizada tanto para etapas de preprocesamiento y mejora de las imágenes, como en la fase de segmentación o también en la de extracción de características y que fue propuesta y desarrollada por Matheron y Serra 2002) en 1964. A lo largo de este capítulo se presentarán los conceptos que permiten entender el funcionamiento de las operaciones morfológicas para luego pasar a describir las principales operaciones. El procesamiento morfológico emplea operaciones que pretenden simplificar las imágenes preservando su forma.

El capítulo comenzará con la explicación de la morfología aplicada a imágenes binarias, para luego dar una breve explicación al empleo de la morfología aplicada a imágenes con niveles de gris.

Con cada operación se mostrarán ejemplos para ilustrar su funcionamiento y también se expondrán casos de utilidad práctica en lo que podría ser una aplicación de visión por computador.

5.2. Operaciones básicas con conjuntos

El lenguaje de la MM está basado en la teoría de conjuntos y en la topología. Hay también un enfoque empleando conjuntos ordenados y álgebra de *Lattice*, que trabaja con conjuntos ordenados. En morfología los conjuntos representan las formas de los objetos, ya sea en imágenes binarias o de niveles de gris. Así, el conjunto de todos los píxeles puestos a 1 en una imagen binaria constituye una descripción completa de la misma. Para el empleo de operaciones morfológicas con imágenes binarias se considerarán conjuntos de puntos 2D, representando a los objetos de la imagen, asumiendo, por ejemplo, que el valor 1 se otorga al objeto y 0 al fondo.

Antes de adentrarnos propiamente en las operaciones morfológicas se definirán algunas propiedades de conjuntos que resultarán útiles (Pajares y De la Cruz 2001).

Inclusión: Un conjunto X está incluido en otro Y, si todo elemento de X también pertenece a Y.

$$Inclusión: X \subset Y: \{\forall x \ \in X \ entonces \ x \in Y\} \tag{5.1}$$

A cada transformación Ψ(X), se le puede asociar una **transformación dual** Ψ*(X), de modo que si dos operaciones son duales, por ejemplo la intersección y la unión, se llegue al mismo resultando sustituyendo una por la otra y los conjuntos por sus complementarios. Esto también es aplicable al álgebra de Boole con las operaciones AND y OR reemplazando los 0 por 1. Más adelante aparecerán nuevas operaciones duales.

$$Complemento: X^c = \{x | x \notin X\} \tag{5.2}$$

$$Dualidad: \Psi^*(X) = (\Psi(X^c)^c \tag{5.3}$$

Unión de conjuntos: Un conjunto es unión de los conjuntos X e Y si está constituido por todos los elementos de los conjuntos X e Y, es decir, sus elementos o bien pertenecen a X, a Y, o a ambos. La unión se programa fácilmente entre imágenes binarias con la operación booleana OR.

$$Unión: X \cup Y = \{x | x \in X \ \vee x \in Y\} \tag{5.4}$$

Intersección de conjuntos: Es el conjunto formado por los elementos comunes de X e Y. La intersección es fácil de realizar entre imágenes binarias con la operación booleana AND.

$$Intersección: X \cap Y = \{x | x \in X \ \wedge x \in Y\} \ de \ forma \ dual \ X \cap Y = (X^c \cup Y^c)^c \tag{5.5}$$

Al igual que la unión, la intersección de conjuntos es conmutativa, asociativa e idempotente:

$$Conmutatividad: X \cap Y = Y \cap X \; ; \; X \cup Y = Y \cup X \; ; \tag{5.6}$$

$$Asociatividad: (X \cap Y) \cap Z = X \cap (Y \cap Z); \; (X \cup Y) \cup Z = X \cup (Y \cup Z) \tag{5.7}$$

Idempotencia: Esta propiedad, que se cumple con algunas operaciones, implica que no se logra nada al repetir la operación sobre un conjunto si también se repite el mismo operador.

$$Idempotencia: (X \cap Y) \cap Y = (X \cap Y); \; (X \cup Y) \cup Y = (X \cup Y) \tag{5.8}$$

Diferencia entre conjuntos: Es el conjunto donde sus elementos son pertenecientes a un conjunto pero no al otro.

$$Diferencia: X - Y = \{x | x \in X \wedge x \notin Y\} = X \cap Y^c \tag{5.9}$$

Traslación por un vector: Un conjunto X es trasladado por un vector v de coordenadas (v1,v2) cuando cada uno de los elementos de x de coordenadas (a,b) sufre esa traslación.

$$Traslación: X_v = \{y | y = x + v \; \forall x \in X\} \tag{5.10}$$

Esta propiedad de la traslación será de gran utilidad para la implementación de las operaciones morfológicas.

5.3 Principios y transformaciones morfológicas básicas

Las transformaciones de la MM consisten en el empleo de las operaciones de conjuntos antes enumeradas. Se empleará, por una parte, un conjunto X constituido por un elemento de la imagen identificado como objeto, y otro conjunto B, habitualmente de pequeño tamaño, llamado **elemento estructurante** (EE o también SE *Structuring Element*). Según la forma y tamaño del EE, así como del tipo de funciones que se combinen, podrán realizarse multitud de transformaciones de la imagen con diversos ámbitos de aplicación.

Las coordenadas de X se representan como suele ser habitual, considerando el origen en la primera fila y columna de la imagen. Para el elemento estructurante se define un origen local o punto representativo, no necesariamente el central o el superior izquierdo.

Una forma de concebir las operaciones morfológicas es trasladar el EE a lo largo de la imagen, de manera semejante a las convoluciones en el filtrado lineal, en cada nueva posición se realizarán operaciones de conjuntos entre el conjunto que forma el objeto de la imagen y los puntos con valor 1 del EE. El problema de exceder contorno de la imagen se puede resolver con cualquiera de los métodos que también se emplean en la convolución lineal: añadir un píxel adicional al contorno igual al vecino del lateral, no procesar los límites de la imagen etc.

5.3.1 Dilatación (*dilation*)

Es la operación que da como resultado un conjunto de puntos, formado por todas las posibles sumas de pares de puntos, en las que uno pertenece al conjunto X y otro al elemento estructurante desplazado. Esta operación es también conocida como adición de Minkowski, es una operación de crecimiento progresivo: sobre cada uno de los puntos de la imagen se traslada el origen del EE, si la intersección del conjunto X con los píxeles a 1 del EE desplazado sobre el punto en estudio es no vacía, entonces el punto en cuestión pertenece a la dilatación de X. De una manera formal:

$$Dilatación: X \oplus B = \{c | c = x + b, \forall\, x \in X\ \forall\, b \in B\} \tag{5.11}$$

También puede expresarse a partir del reflejo $\check{B}$ del elemento estructurante, obtenido por simetría respecto al punto considerado origen. Un posible algoritmo sería el siguiente:

- Se calcula el EE reflejado, $\check{B}$, tomando como punto pivote el elemento central del EE.
- Se pone el centro del EE en la coordenada origen de la imagen.
- Se recorren todas las filas y columnas.
- En cada punto donde la intersección del conjunto imagen y $\check{B}$, sea no vacía, nos encontraremos en un punto de la dilatación.

$$Dilatación: X \oplus B = \{x | \check{B} \cap X \neq \emptyset\} \tag{5.12}$$

Dependiendo de la forma y tamaño del EE el crecimiento se puede orientar en uno u otro sentido, lo más habitual es emplear elementos isótropos, que no privilegien ninguna dirección.

Al dilatar la imagen aumenta su tamaño, pero pueden perderse detalles de la forma. Se rellenan intersticios de la imagen y también, como efecto muchas veces pernicioso, se pueden fusionar objetos diferentes que se encuentren próximos entre si. Cuando el elemento estructurante es grande el efecto es más acusado. En la figura 5.1 se ve el efecto de la dilatación con 2 EE diferentes.

Con EE (b) : X={(0,1),(1,1),(1,2),(2,1); B={(0,0),(0,1)}
Dilatación={(0,1),(1,1),(1,2),(2,1),(0,2),(1,2),(1,3),(2,2)}
Con EE (d) : X={(0,1),(1,1),(1,2),(2,1); D={(0,-1),(0,0)}
Dilatación={(0,1),(1,1),(1,2),(2,1),(0,0),(1,0),(1,1),(2,0)}

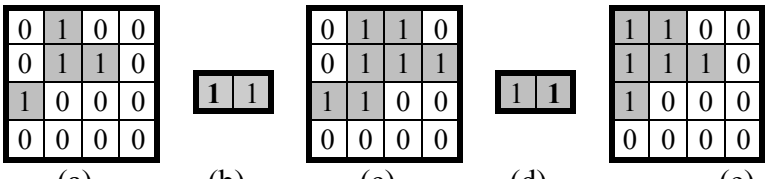


Figura 5.1 (a) Imagen original (b) EE centrado con origen en el píxel izquierdo; (c) Imagen dilatada con (b); (d) EE centrado con origen en el píxel derecho (d); (e) Imagen dilatada con (d):

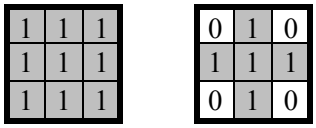
Las propiedades de la dilatación, (Haralick y col. 1987), son: conmutatividad, asociatividad y las siguientes:

$$Invarianza\ a\ la\ traslación: X_v \oplus B = (X \oplus B)_v \quad (5.13)$$

$$Creciente: SI\ X \subset Y \Rightarrow X \oplus B \subset Y \oplus B \quad (5.14)$$

$$Distributiva\ respecto\ a\ la\ unión: B \oplus (X \cup Y) = (B \oplus X) \cup (B \oplus Y) \quad (5.15)$$

Los EE más comunes cuadrados para la dilatación, que pretenden preservar la isotropía, son los mostrados en la figura 5.2:



(a) (b)

Figura 5.2 (a) EE para nueve vecinos; (b) EE para 4 vecinos

Las operaciones de dilatación no suelen emplearse por si solas, ya que tienden a deformar la imagen y a perder detalles e información. Su uso es más habitual combinado con otras operaciones. Se debe tener presente que esta operación no es reversible, es decir, no se puede recuperar la imagen original.

5.3.2 Erosión (*erosion*)

Es una operación con el efecto contrario a la dilatación, por lo que reduce el tamaño de los objetos. La mecánica es semejante, desplazando el elemento estructurante a lo largo de la imagen, pero ahora el resultado será 1 para el punto en estudio si todos los puntos del EE están contenidos dentro del conjunto del objeto. Esto puede expresarse en función del complementario de la imagen de la siguiente manera: un punto pertenecerá a la imagen erosionada si el elemento estructurante desplazado para cada x tiene una intersección no vacía con el conjunto complementario del objeto.

$$Erosión: X \ominus B = \{x | B_x \subseteq X\} = \{x | B_x \cap X^c \neq \emptyset\} \quad (5.16)$$

La erosión reducirá el tamaño de los objetos dependiendo de la dimensión y forma del elemento estructurante: los intersticios se agrandan, la separación entre objetos crece, elementos pequeños de la imagen pueden desaparecer por lo que es muy útil para eliminar ruido impulsional o de pequeño tamaño. Aunque el efecto es contrario al de la dilatación esto no debe llevar al error de considerar que erosionando una imagen dilatada seremos capaces de recuperar la imagen original, esto nunca es así.

Para el cálculo de la erosión el algoritmo sería el siguiente:

- Se pone el centro del EE en la coordenada origen de la imagen.

- Se recorren todas las filas y columnas, situando el origen del EE sobre los píxeles del conjunto X.
- Se erosionará, eliminará, cada punto del conjunto donde el EE no esté contenido por completo en el conjunto X, verificable haciendo operaciones AND entre los elementos de X y el EE.

En la figura 5.3 se puede ver el efecto de esta operación empleando dos EE iguales en tamaño pero con origen diferente.

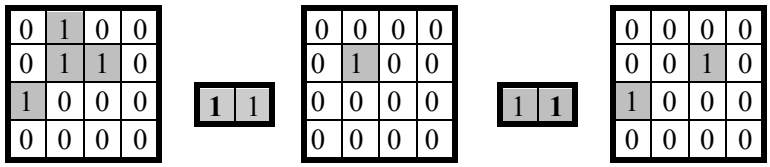


Figura 5.3 (a) Imagen original; (b) EE centrado con origen en el píxel izquierdo; (c) Imagen erosionada con (b); (d) EE centrado con origen en el píxel derecho (d); (e) Imagen erosionada con (d).

En cuanto a sus propiedades, que podemos ver en Haralick y col. (1987), señalar que NO es conmutativa, si es invariante a la traslación y también es creciente, además de esto:

$$(X \cup Y) \ominus B \supseteq (X \ominus B) \cup (Y \ominus B) \tag{5.17}$$

$$B \ominus (X \cup Y) = (X \ominus B) \cup (Y \ominus B) \tag{5.18}$$

$$Si\ D \subseteq B\ entonces\ X \ominus B \subseteq\ X \ominus D \tag{5.19}$$

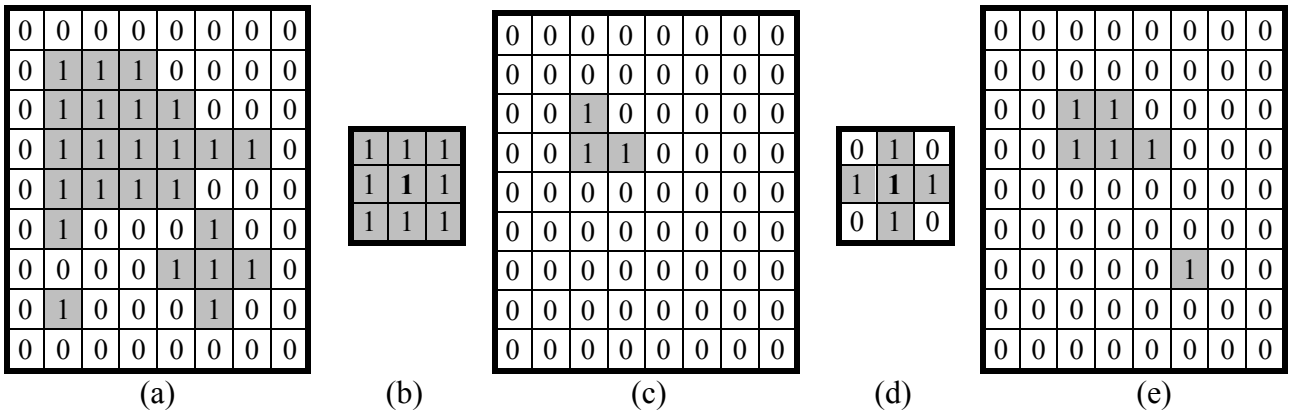


Figura 5.4 (a) Imagen original; (b) EE vecinos a 8 ;(c) Erosión de la imagen con (b); (d) EE vecinos a 4 ;(e) Erosión de la imagen con (d).

En las imágenes (c) y (e) de la figura 5.4 se puede apreciar como ha desaparecido el píxel aislado, que podemos identificar como ruido, pero también se ha reducido el tamaño de la figura y en este caso, por ser una figura pequeña, ha perdido elementos del contorno que podrían ser definitorios.

La erosión puede emplearse para suprimir de una imagen los elementos más pequeños dejando solo los grandes para, por ejemplo, aplicaciones de conteo y etiquetado. También permite separar objetos rompiendo contactos débiles entre ellos.

5.3.3 Apertura (*opening*)

Esta operación se realiza mediante una erosión seguida de una dilatación empleando el mismo EE. Al erosionar se pierden detalles pequeños, desaparece el ruido, mengua la imagen. Haciendo seguidamente la dilatación la imagen crece, y los detalles que no llegaron a perderse se resaltan.

$$Apertura: X \circ B = (X \ominus B) \oplus B \tag{5.20}$$

También se puede formular como la unión del EE B desplazado de tal manera que está siempre contenido por completo dentro del conjunto imagen.

$$Apertura: X \circ B = \cup\{B_v \mid B_v \subseteq X\} \tag{5.21}$$

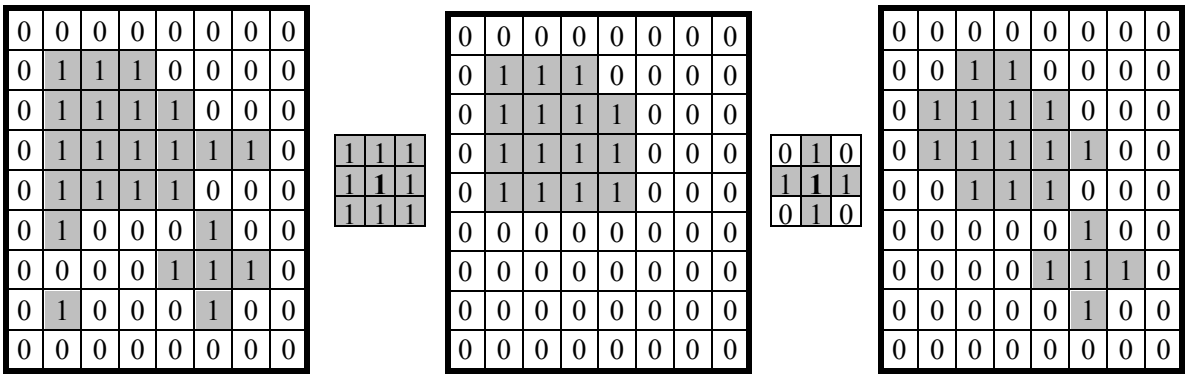


Figura 5.5 (a) Imagen original; (b) EE vecinos a 8; (c) Apertura de la imagen (a) con b; (d) EE vecinos a 4; (e) Apertura de la imagen (a) con (d)

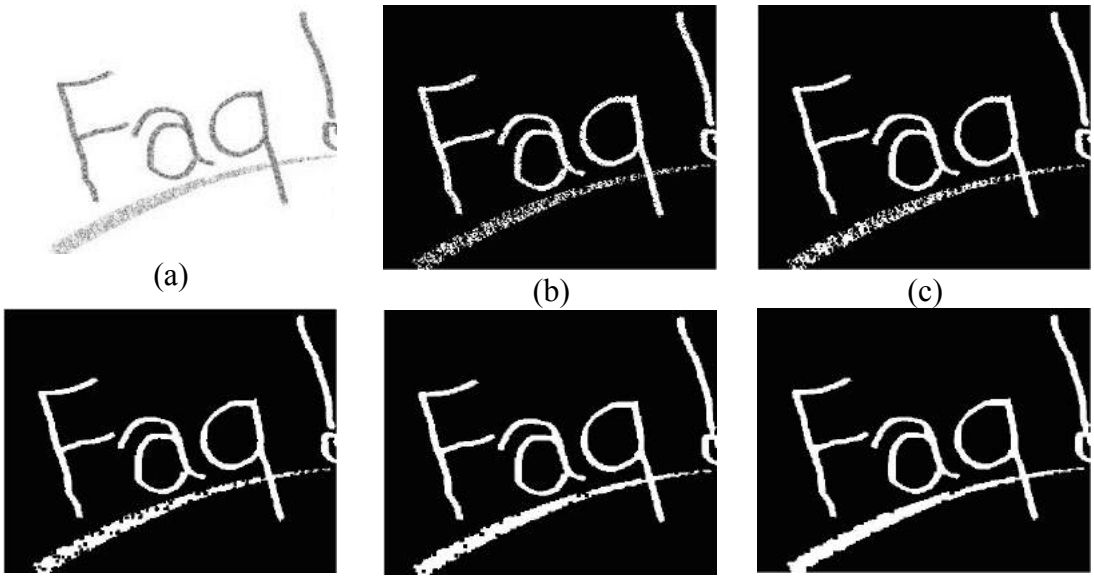
Los resultados de la erosión se pueden ver en la figura 5.4, aplicando a ellos una dilatación con el mismo EE se obtienen los resultados de la figura 5.5. Se puede comprobar que la dilatación no es la inversa de la erosión, ya que no se recupera la imagen original.

5.3.4 Cierre (*closing*)

En este caso en primer lugar se hará una operación de dilatación, para seguidamente, manteniendo el mismo EE, hacer una erosión. Al dilatar se rellenan intersticios, la imagen crece, luego cuando se erosiona la imagen vuelve a un tamaño semejante al original.

$$Cierre: X \bullet B = (X \oplus B) \ominus B \tag{5.22}$$

$$Cierre: X \bullet B = \cup\{B_v \mid B_v \cap X \neq \emptyset\} \tag{5.23}$$



(a) (b) (c)

(d) (e) (f)

Figura 5.6 Desde la fila superior y de izquierda a derecha imagen en gris, binarizada y cierre de la misma con EE cuadrados de 3,5,7 y 9 elementos de lado.

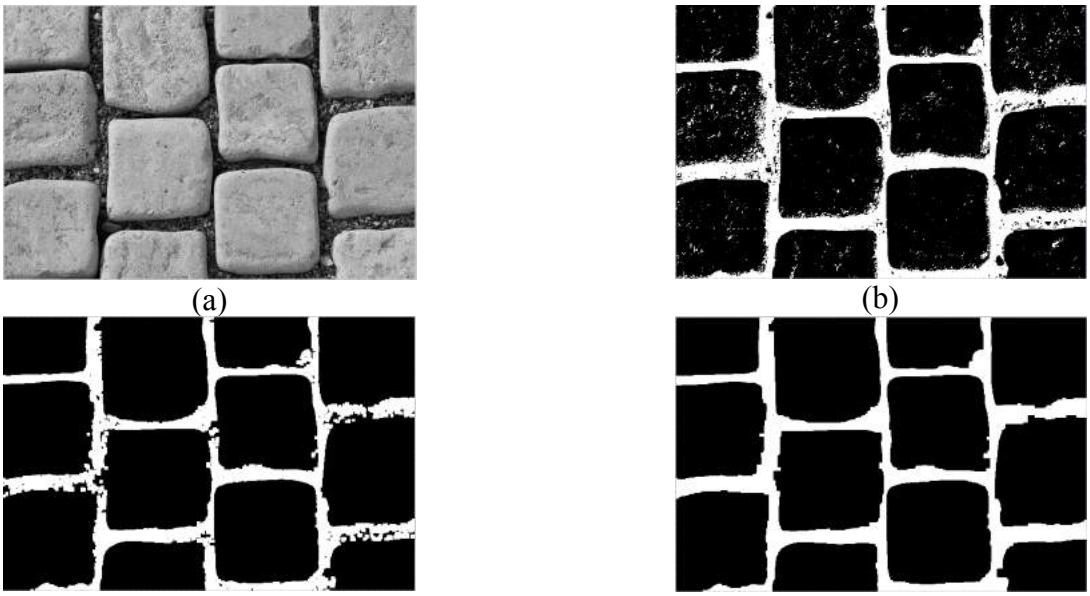
Veamos un ejemplo sencillo del cierre para rellenar intersticios dentro de una imagen, en cada caso se emplea un EE cuadrado como el (b) de la figura 5.5 aunque de diferentes tamaños En esta serie de imágenes se puede ver que el trazado, que en la imagen binarizada aparece discontinuo, se va rellenando a medida que se emplean elementos estructurantes más grandes. Se aprecia también que en lo sustancial la forma no ha variado, si acaso ha perdido irregularidades, pudiendo ser más fácil de tratar para, por ejemplo, algoritmos de determinación de contornos.

(b) (a)

Figura 5.7 Operación de apertura: (a) imagen binaria, (b) apertura con un EE cuadrado tamaño 5. En el caso de la imagen del serrucho se ha realizado una operación de apertura para suprimir los dientes de sierra, preservando el orificio circular de la punta del serrucho, que se perdería con operaciones de cierre.

Ejemplo 5.1. *De la imagen de la figura 5.8 (a) se desea obtener perfectamente separados los adoquines del empedrado para por ejemplo determinar su tamaño.*

Solución: Tras la umbralización y obtención de una imagen binaria figura 5.8 (b) se realiza una apertura, esto suprime el ruido aunque abre las regiones separadoras, figura 5.8 (c). Posteriormente un cierre restablece las uniones figura 5.8 (d).



(a) (b)

(c) (d)

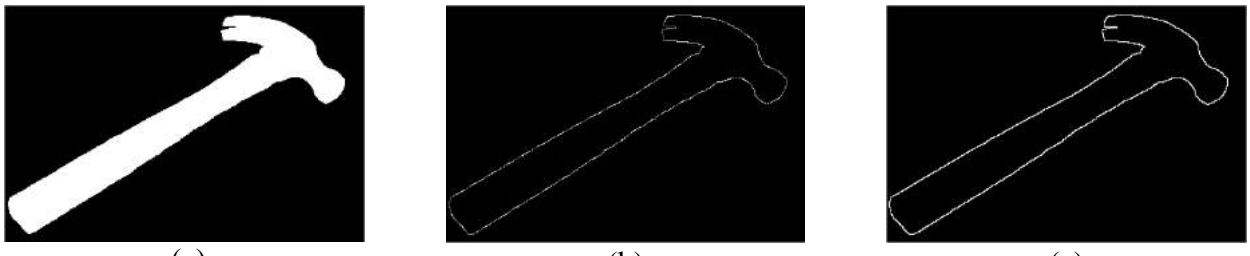
Figura 5.8 (a) Imagen original (b) imagen binarizada, (c) tras una apertura con EE cuadrado de 5 elementos, (d) Cierre con un EE cuadrado de 9 elementos

5.3.5 Obtención de contornos

Con las operaciones de erosión y dilatación es fácil obtener un contorno, de la parte exterior o interior al perímetro de una figura. Para ello tomaremos la imagen dilatada y le restaremos la original, en el caso del contorno exterior. Para la obtención del contorno interior restaremos de la imagen original la imagen erosionada. El contorno puede ser tanto o más marcado empleando EE de mayor tamaño tal como se ve contorno exterior del perímetro en la figura 5.9.

$$Contorno_{exterior} = (X \oplus B) - X \tag{5.24}$$

$$Contorno_{Interior} = X - (X \ominus B) \tag{5.25}$$



(a) (b) (c)

Figura 5.9 (a) Imagen binaria (b) contorno exterior obtenido tras una dilatación con EE cuadrado de 3 elementos y posterior sustracción de la imagen binaria (c) Ídem que (b) pero con EE de tamaño 5.

5.3.6 Rellenado de regiones *(región filling*)

Anteriormente se mencionó que mediante el cierre pueden rellenarse intersticios y huecos interiores. Esto puede apreciarse en la figura 5.6 donde el éxito en el relleno depende del tamaño del EE y queda limitado a pequeños huecos. Hay otro método, (Raid y col. 2014), basado en un algoritmo iterativo, que permite rellenar regiones sin tener que usar EE grandes.

$$Relleno\ de\ regiones: X_k = (X_{k-1} \oplus B) \cap A^c \tag{5.26}$$

Se irá construyendo un conjunto de puntos Xk, partiendo de una imagen en la que solo contiene un punto del interior, hasta que se encuentre que Xk y Xk-1 son iguales. En cada iteración el conjunto Xk es el resultado de la dilatación de Xk-1 con el EE y la posterior intersección con el complementario del conjunto de la imagen original tal como se ve en la figura 5.10.

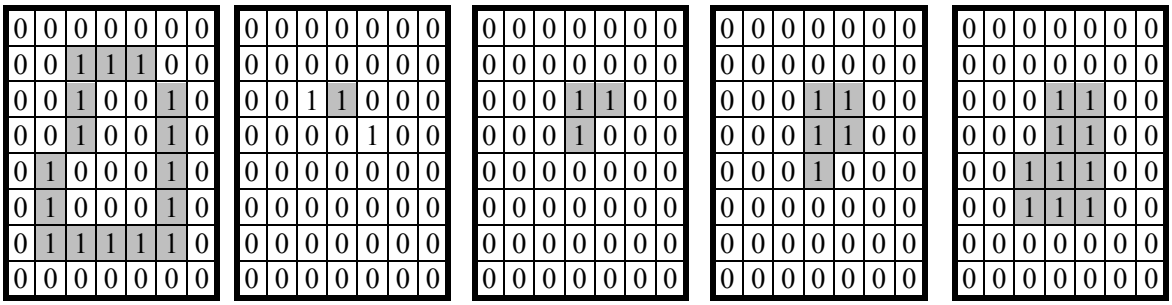
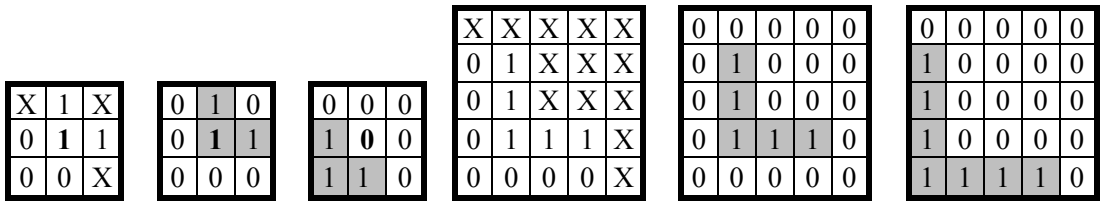


Figura 5.10 (a) Imagen de contorno (b) píxel interior de arranque (c) a (d) resultado de las 2 primeras iteraciones (e) resultado final. El EE es de tamaño 3x3 para 4 vecinos.

5.3.7 Transformación acierta o falla (*Hit-or-Miss*)

Esta transformación buscará un patrón sobre la imagen. El EE se establece del tamaño y forma concordante con el patrón. En nuestro EE habrá tres categorías de píxeles, los elementos del patrón que tienen valor 1, los que tienen valor 0 y los que es indiferente su valor. Definiremos entonces 2 subconjuntos disjuntos, el primero B1 que contiene los píxel a 1 del conjunto B, y otro B2 poniendo a uno los píxel a 0 del B. Dentro del conjunto B puede haber también ciertos píxeles cuyo valor sea indiferente que esté a 1 o a 0. Los elementos del patrón cuyo valor es irrelevante, que reflejamos como X, se ponen a 0 en los elementos estructurantes B1 y B2.

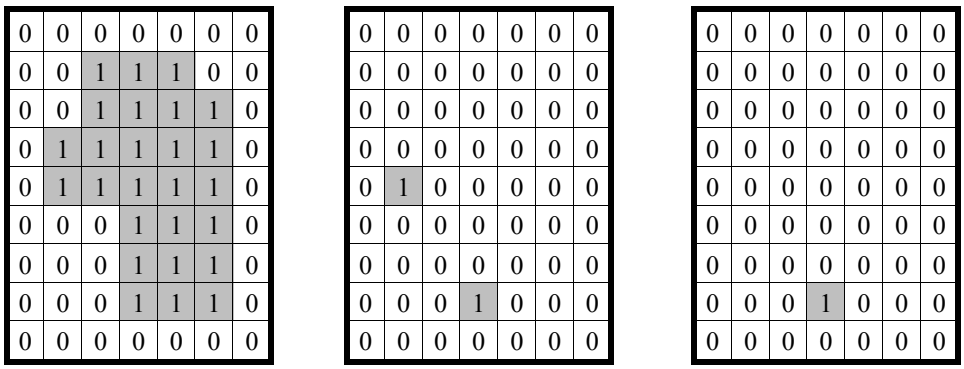


(a) (b) (c) (d) (e) (f)

Figura 5.11 (a) Patrón detección esquina inferior izquierda; (b) elemento B1 con los 1 del patrón; (c) elemento B2 con los 0 del patrón puestos a 1. (d), (e), (f) es el detector de esquinas aún mayores

$$Acierta-falla:\ X \circledast B = \{x|B1 \subset X \wedge B2 \subset X^c\} = (X \ominus B1) \cap (X^c \ominus B0) \qquad (5.27)$$

La parte B1 del EE compuesto que tiene su punto representativo en el punto marcado en negrita de la figura 5.11 estaría contenida en X, y simultáneamente, la parte B2 del EE estaría contenida en el conjunto complementario de 5. En la figura 5.12 se puede ver la aplicación de los detectores de esquinas de la figura 5.11.



(a) (b) (c)

Figura 5.12 (a) Imagen original; (b) esquinas detectadas con (a) de la figura 5.10; (c) esquinas detectadas con (d) de la figura 5.11.

5.3.8 Adelgazamiento (*thinning*) y engrosamiento (*thickening*)

Estas transformaciones pueden obtenerse mediante aplicaciones de la operación acierta–falla. Empezando por la operación de adelgazamiento, se puede definir un EE para una operación acierta-falla, lo que implica que tendrá dos componentes. La imagen adelgazada es el resultado de sustraer, según 5.9, el resultado de la operación acierta-falla de la imagen original, la engrosada se realiza con la suma. Adelgazar una imagen es equivalente a engrosar el fondo, son por lo tanto operaciones duales

$$Adelgazamiento: (X \rhd B) = X - (X \circledast B) \qquad (5.28)$$

$$Engrosamiento\ (X \lhd B) = X \cup (X \circledast B) \qquad (5.29)$$

El método será trasladar el origen del EE por cada píxel de la imagen y estudiar el resultado de la operación acierta-falla. En el caso del adelgazamiento si ese píxel acierta, píxel a 1 y a 0 del EE concuerdan con el entorno del píxel estudiado, entonces ese píxel es marcado como elemento de fondo,

se sustrae de la imagen. En caso de fallo el píxel queda inalterado. Para la operación de engrosamiento en caso de acierto el píxel se une al objeto y si falla la imagen queda inalterada.
Estas operaciones suelen realizarse de manera iterativa, mostraremos ejemplos para cada técnica.

5.3.9 Esqueletonización (*skeleton*)

Este es un procedimiento para obtener un descriptor de una región mediante un grafo, la idea es que dos objetos diferentes tendrán distinto grafo, que ocupan mucho menos espacio de almacenamiento que la región y que son también más fáciles de procesar y comparar.
El esqueleto de una región sería el grafo de los puntos interiores cuya distancia mínima a algún punto del borde se repite dos o más veces. Suele ilustrarse el efecto de determinación del esqueleto como el último frente de fuego que quedaría si se pudiera encender simultáneamente todo el perímetro de la región y asumimos que este fuego avanza uniformemente. Obtener el esqueleto mediante el cálculo de distancias sería muy costoso computacionalmente por lo que se sustituye por otro más rápido basado en la utilización de filtros morfológicos, que pueden orientarse a trabajar con conectividades a 4 o a 8. Un conjunto de estos elementos se puede ver en la figura 5.13. A base de sucesivas iteraciones se iría adelgazando la imagen, hasta obtener el esqueleto cuando no haya diferencia entre dos iteraciones sucesivas.
En cada iteración deben respetarse 2 normas:

- No suprimir puntos extremos
- No debe romperse la conectividad

X 0 X X 1 X 1 1 1	X 0 0 1 1 0 1 1 X	X X 1 0 1 1 X X 1	0 0 X 0 1 1 X 1 1	1 1 1 X 1 X X 0 X	X 1 1 0 1 1 0 0 X	1 X X 1 1 0 1 X X	1 1 X 1 1 0 X 0 0
(a)	(b)	(c)	(e)	(f)	(g)	(h)	(i)

Figura 5.13 Elementos acierta-falla para obtener el esqueleto empleando conectividad 4, en (Escalera 2001, pp 228) se pueden ver los elementos para una conectividad a 9.

Este método es también computacionalmente costoso, en cada iteración se empleas 8*2=16 máscaras, 12*2 en el caso de la conectividad a 8, y debe iterarse hasta reducir la figura a un grafo. Hay otro método diferente, propuesto por Zhang y Suen (1984) y descrito en Escalera (2001) que puede ofrecer mejores resultados.

Se definen 8 vecinos de un píxel y, de forma iterativa, se aplican dos conjuntos de condiciones. Aquellos píxeles que cumplan en una etapa todas las condiciones de un conjunto se eliminan.

P8	P1	P2
P7		P3
P6	P5	P4

Figura 5.14 Ordenación de los 8 vecino del píxel.

Conjunto 1

1. Para preservar el mantenimiento de los puntos finales se examina si el número de vecinos distintos de 0 es ≥2 y ≤6.
2. Que solamente se pasa de 0 a 1 una vez si se recorre el borde del entorno de P1 a P8.
3. Que alguno de los vecinos P1, P3 y P5 es cero.
4. Que alguno de los P3, P5 o P7 es cero

Conjunto 2

1. Igual a la del conjunto 1
2. Igual a la del conjunto 2
3. Que alguno de los P1, P3 y P7 es cero
4. Que alguno de los P1, P5 o P7 es cero

Una forma más rápida de realizar este algoritmo es mediante una tabla LUT (*Look Up Table*), en la que se programa la convolución lineal de la imagen con el núcleo de la figura 5.15, formado por potencias de 2. El valor resultante para cada píxel estará entre 0 y 255.

128	1	2
64		4
32	16	8

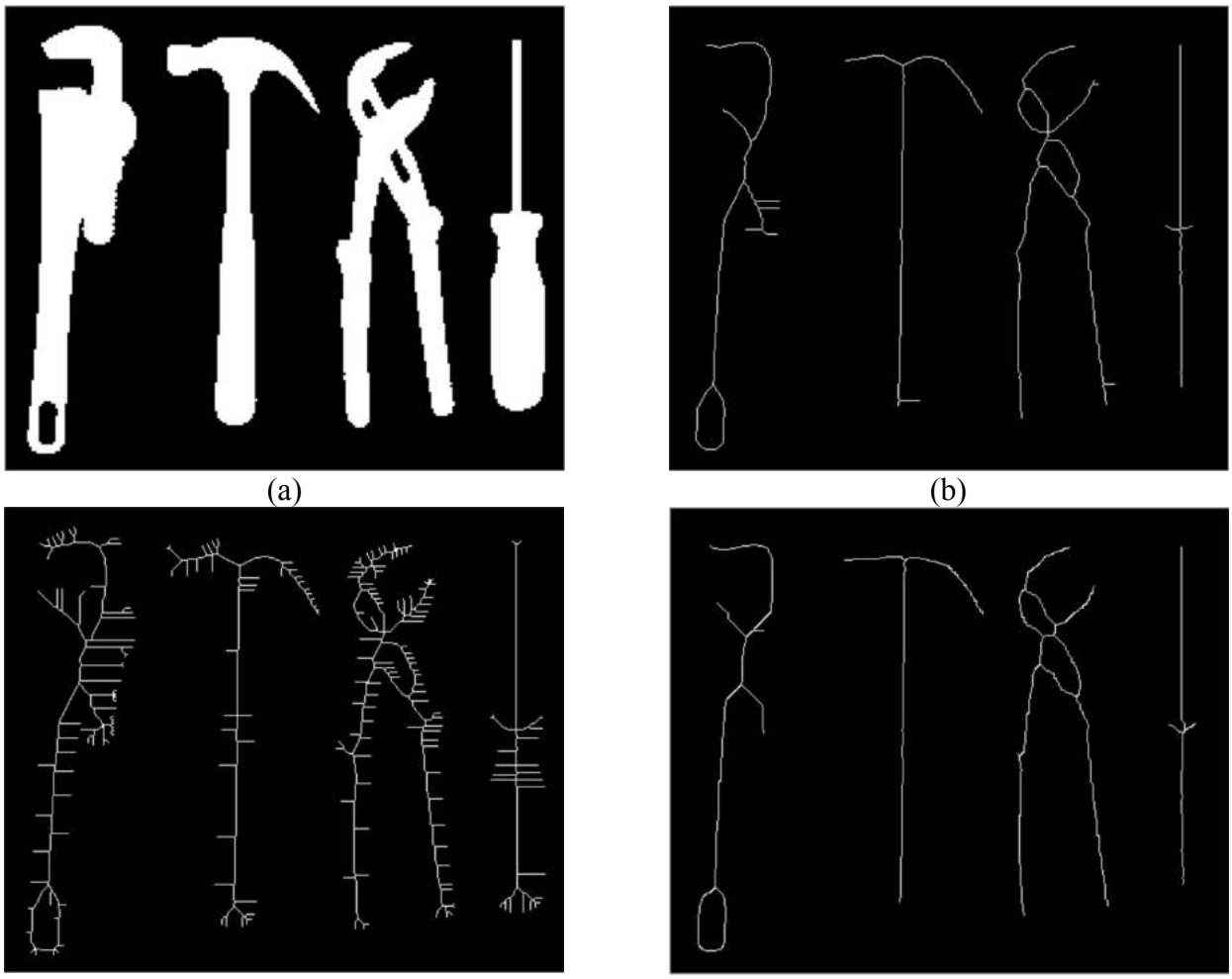
Figura 5.15 Pesos para la implementación del esqueleto.

Calculando por anticipado el valor numérico que correspondería a la verificación de las 4 condiciones de cada conjunto encontramos ciertos valores, 2 subconjuntos, que indican que el píxel debe ser eliminado si el resultado de la convolución se encuentra dentro de uno de los subconjuntos.

Eliminar_S1={3, 6, 7, 12, 14, 15, 24, 28, 30, 48, 56, 60, 62, 96, 112, 120, 129, 131, 135, 143, 192, 193, 195, 199, 207, 224, 225, 227, 231, 240, 241, 243, 248, 249}

Eliminar_S2={3, 6, 7, 12, 14, 15, 24, 28, 30, 31, 48, 56, 60, 62, 63, 96, 112, 120, 124, 126, 129, 131, 135, 143, 159, 192, 193, 195, 224, 225, 227, 240, 248, 252}

Por ejemplo, si los vecinos P1, P2 y P8 de la figura 14 están a 1 y los restantes son 0 el valor obtenido sería 131, pertenece al conjunto Eliminar_S1 por lo que se suprime y no pasa a la siguiente iteración. Se ve que efectivamente se cumplen las 4 condiciones del conjunto 1



(a) (b)

(c) (d)

Figura 5.16 (a) Original; (b) adelgazada hasta 1 píxel; (c) esqueleto; (d) esqueleto usando Zhang-

En la figura anterior 5.16 (c) se ve cómo el esqueleto contiene muchas ramificaciones, debidas a irregularidades en los contornos de las figuras. Esas ramificaciones confunden respecto a la forma básica del objeto, puede ser suprimidas realizando operaciones acierta-falla de eliminación de ramas (*pruning*).

La correcta supresión de las ramas está tratada en Xiang y col. (2007), véase la figura 5.17.

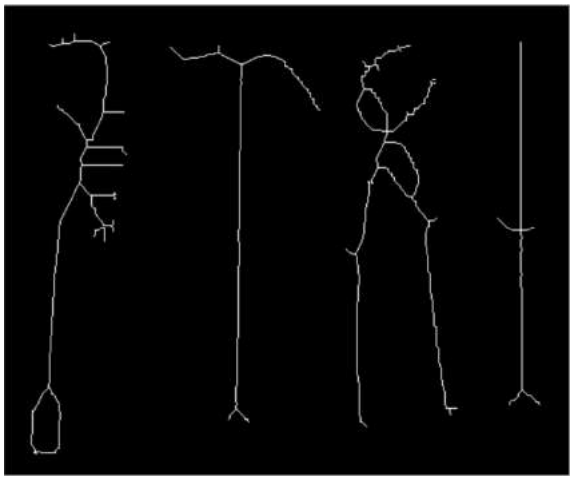


Figura 5.17 Esqueleto podado suprimiendo elementos entre bifurcaciones y puntos finales

5.3.10 Cerco convexo (*convex hull*)

Este es un descriptor de regiones útil, que puede calcularse con varias técnicas, pero también aplicando la MM. Se trata de determinar el contorno más pequeño de una región de tal manera que sea posible unir con un segmento recto dos puntos cualesquiera de la región estando ese segmento contenido por completo en ella. La forma más sencilla de visualizar lo que buscamos es suponer al objeto envuelto con una banda elástica y la forma que adoptase la banda sería nuestra envoltura o contorno convexo.

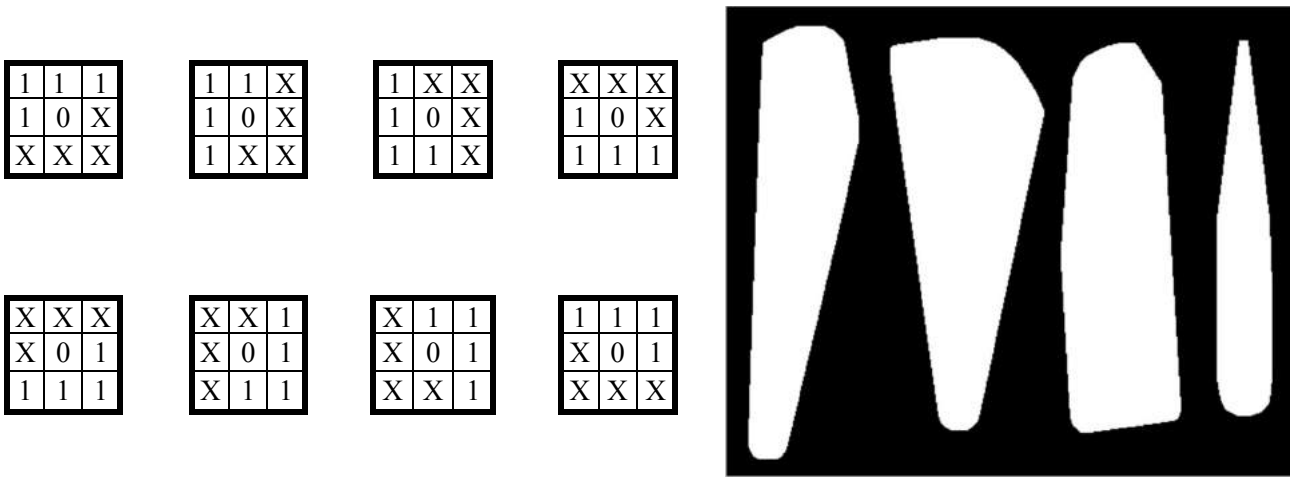


Figura 5.18 Máscaras para la determinación del cerco convexo y cerco calculado para las herramientas de la figura 5.15 (a).

5.4 Morfología con niveles de gris.

En el apartado 5.3 los conjuntos estaban formados por imágenes binarias, con lo cual la determinación de pertenencia a objeto o fondo es inmediata. Ahora dentro de los píxel que definen un objeto puede haber múltiples valores. Para aplicar MM debemos redefinir las operaciones, empleando el álgebra de retículos (*Lattice*), que se basarán como veremos en el ordenamiento de los valores de los píxeles de la imagen. Veamos en primer lugar algunas definiciones formales (Serra y Vincent 1992).

- Un elemento $x \in X$, es cota inferior de S si $x \leq y$, para todo $y \in S$.
- Un elemento $y \in Y$, es cota superior de S si $x \leq y$, para todo $x \in S$.
- Un elemento $x \in X$, es extremo inferior o ínfimo de S si y sólo si es cota inferior de S y para toda cota inferior i de S se verifica que $i \leq 5$. (Es la mayor de la cotas inferiores). Si este elemento existe es único y se denota por $\wedge$.

- Un elemento y ∈ X, es extremo superior o supremo de S si y sólo si es cota superior de S y para toda cota superior de S se verifica que y≤ s.(es la menor de las cotas superiores). Si este elemento existe es único y se denota por ∨.
- Un conjunto ordenado (X, ≤) es un retículo completo (*lattice*) si todos los subconjuntos de X poseen un ínfimo y un supremo.

Trabajar con escalas de grises, manejando los operadores de mínimo y máximo, supone pasar de operar con conjuntos a hacerlo con funciones, donde el valor de gris es representativo del valor de la función en cada punto: E → T, donde E es el espacio de puntos y T el de niveles de grises, que será siempre un subconjunto de los números reales (González y Woods 2009). Los niveles de gris estarán numéricamente ordenados y las operaciones morfológicas computarán en cada punto de E en una combinación de supremos e ínfimos de valores de gris.

5.4.1 Erosión y dilatación

Redefiniremos en primer lugar la erosión y la dilatación. En 'teoría de retículos', una erosión es una operación que conmuta con el ínfimo y por su parte la dilatación conmuta por el supremo. Cuando se trata de imágenes en escalas de gris, donde los elementos del conjunto solo toman valores enteros, el máximo y el supremo coinciden, de la misma manera que lo hacen el ínfimo y el mínimo.

$$Dilatación: f \oplus b = \max\{f(x-i, y-j) + b(i,j) \mid (x-i, y-j) \in Df; (i,j) \in Db\} \quad (5.30)$$

$$Erosión: f \ominus b = \min\{f(x-i, y-j) - b(i,j) \mid (x-i, y-j) \in Df; (i,j) \in Db\} \quad (5.31)$$

En ambos casos Df y Db son el dominio de la imagen y el del EE respectivamente; (x,y) definen las coordenadas en el dominio de la imagen e (i,j) lo hacen en el del EE. Dado que, en el cálculo del resultado, los valores (i,j) de las coordenadas del EE aparecen restando a las (x,y) de la imagen, a la hora de operar puede emplearse el reflejo del EE. Para cada evaluación de un entorno el resultado será el valor máximo o mínimo del resultado de sumar en una vecindad los valores de los puntos de la imagen y los del EE reflejado. Es frecuente utilizar EE planos, con todos sus valores a cero.

Como se desprende de las fórmulas, asumiendo que el EE está compuesto de valores enteros positivos o cero, la imagen resultante de la dilatación es más clara que la original, mientras que la erosionada es más oscura. Un posible algoritmo sería, continuando con la similitud con las convoluciones:

- Situar el centro del EE reflejado sobre cada punto de la imagen recorriendo filas y columnas.
- Sumar/restar cada valor del EE con el píxel de la imagen que tiene debajo.
- Encontrar el máximo/mínimo y otorgar ese valor ya se trate de la dilatación o de la erosión.

3	4	2	3	4	5	4
5	2	3	4	4	1	2
3	4	5	6	6	6	3
3	2	4	4	3	4	2
5	2	4	4	4	1	2
3	3	4	4	3	2	2
2	1	1	2	2	1	1

(a) Original

1	1	1
1	**1**	1
1	1	1

(b)

6	6	5	5	6	6	6
6	6	7	7	7	7	7
6	6	7	7	7	7	7
6	6	7	7	7	7	7
6	6	5	5	5	5	5
6	6	5	5	5	5	5
4	5	5	5	5	4	3

(c) Dilatada

1	1	1	1	0	0	0
1	1	1	1	0	0	0
1	1	1	2	0	0	0
1	1	1	2	0	0	0
1	1	1	2	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

(d) Erosionada

Figura 5.19 Operaciones morfológicas en escala de gris

En la figura 5.19 podemos ver un ejemplo de aplicación con una imagen y un elemento estructurante no plano, en la 5.20 la aplicación de este mismo EE sobre una imagen de gris.

(a) (b) (c)

Figura 5.20 (a) Imagen en gris de Lena; Imagen erosionada con un EE plano, todos los elementos a cero, de tamaño 3x3; Imagen dilatada con un EE plano 3x3.

Al dilatar no solo la imagen resultante es más clara, también los objetos brillantes ganan extensión y aumenta el valor medio del brillo, por el contrario los oscuros pierden contraste y pueden desaparecer. Los efectos de la erosión son justamente los opuestos.

5.4.2 Apertura y cierre.

Las operaciones de apertura y cierre se definen de la misma manera que en la MM binaria, como sucesión de operaciones de erosión-dilatación y dilatación-erosión respectivamente. Ambas operaciones, como en el caso de la MM binaria, son duales. Combinando un cierre tras una apertura se obtiene un efecto de suavizado, tal como se ve en la figura 5.21.

$$Suavizamiento(f) = (f \circ b) \bullet b \tag{5.32}$$

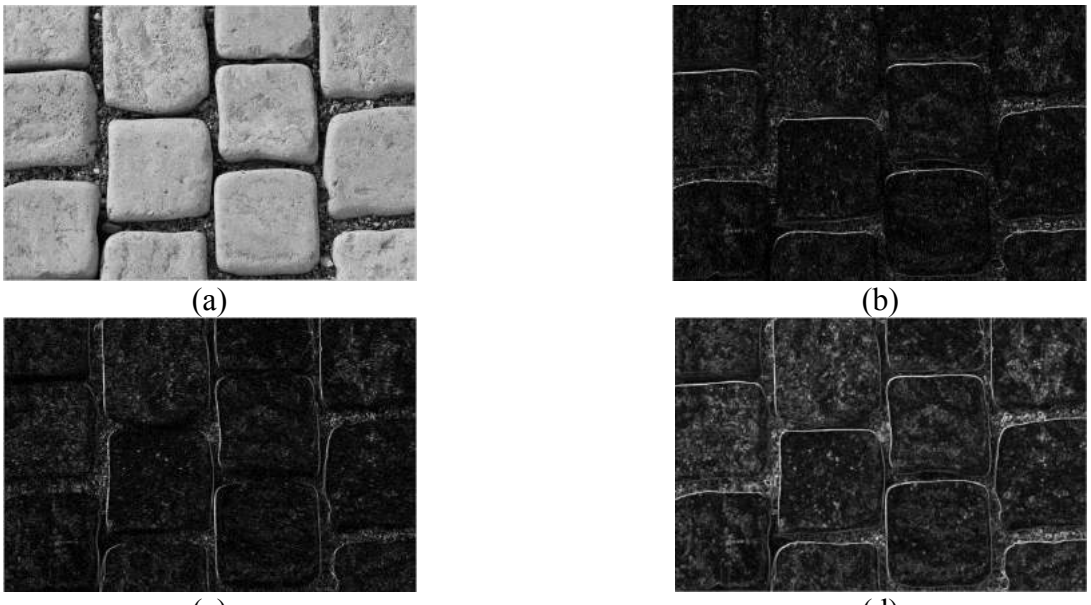
(a) (b) (c)

Figura 5.21 (a) Apertura de la imagen 5.11 (a); (b) Cierre de la imagen 5.11 (a); (c) Cierre de la imagen 5.20 (a)

5.4.3 Gradiente Morfológico

El efecto ya conocido del operador gradiente, basado en el cálculo discreto de las derivadas parciales, es resaltar los elementos de alta frecuencia de la imagen, y de forma particular los contornos de la misma. Este efecto se puede conseguir sustrayendo de la dilatación el resultado de la erosión.

$$Gradiente(f) = f \oplus b - f \ominus b \tag{5.33}$$



(a) (b)

(c) (d)

Figura 5.22 (a) Imagen original (b) gradiente horizontal (c) tras una apertura con EE cuadrado de 5 elementos, (d) Cierre con un EE cuadrado de 9 elementos

Los elementos del contorno pueden destacarse más empleando EE mayores o con valores distinto de cero. Aumentar el tamaño y valor del EE conlleva obtener bordes más gruesos, lo que en principio no es deseable para un detector de contornos.

5.4.4 Transformación Sombrero de Copa (*Top-Hat)*

Las transformaciones anteriores tienden a perder detalles pequeños de la imagen, veremos ahora dos transformaciones para resaltarlos. Esta transformación es muy útil cuando las variaciones de intensidad del fondo de la imagen no permiten destacar, por ejemplo con una umbralización, a objetos de pequeño tamaño o intensidades semejantes al fondo. Hay dos tipos de transformaciones: una llamada *top-hat* de blancos, o simplemente *top-hat*, que destaca los elementos claros. La otra es la llamada *top-hat* de negros o *bottom-hat*, cuya misión es destacar los elementos oscuros.

$$T_{hat}(f) = f - (f \circ b) \tag{5.34}$$

$$B_{hat}(f) = (f \bullet b) - f \tag{5.35}$$

La transformación *top-hat* devuelve aquellos elementos que:

- Son menores que el elemento estructurante.
- Son más brillantes que sus alrededores

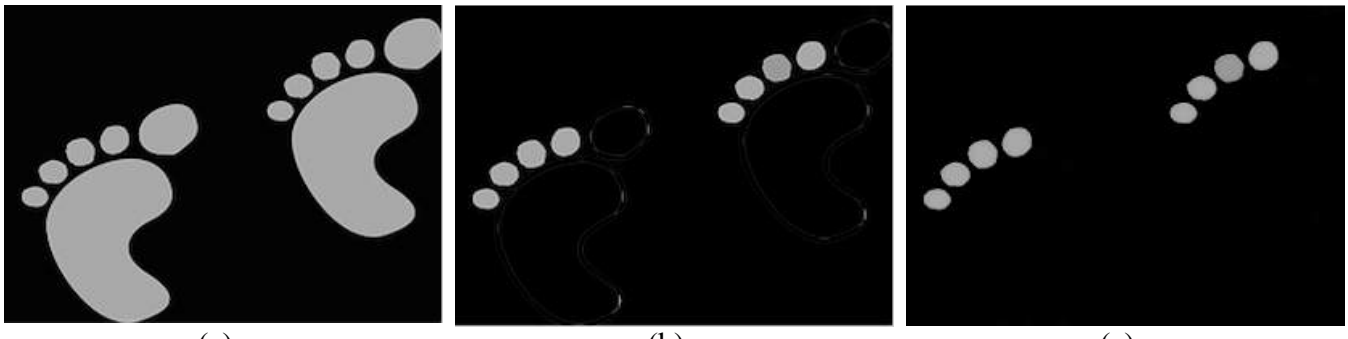
La transformación *bottom-hat* devuelve aquellos elementos que:

- Son menores que el elemento estructurante.
- Son más oscuros que sus alrededores.

Estas transformaciones sirven también para corregir el efecto de una iluminación no uniforme.

Ejemplo 5.2. *de la imagen de la figura con 2 huellas de pie realizar un filtrado para quedarse solamente con los dedos excluyendo los pulgares.*

Una posible solución pasa por emplear una transformación top-hat con un EE en forma de disco, en este caso el tamaño 9 es apropiado. Como se ve en la figura 5.23 aparecen también contornos del pie y del pulgar. Para suprimirlos se puede hacer una apertura, por ejemplo con un disco de tamaño 2.



(a) (b) (c)

Figura 5.23 (a) Imagen original; (b) resultado top-hat disco plano tamaño 9; (c) filtrado de (b) con una apertura para eliminar restos de contornos.

5.5 Conclusiones

Se han expuesto las bases de la morfología matemática, tanto con imágenes binarias como con niveles de gris. Se podría seguir avanzando mostrando más conceptos como la distancia geodésica o la aplicación de la morfología para imágenes, no abordadas aquí por razones de espacio. En la bibliografía aparecen dos referencias: (Vincent 1993) y (Comer y Delp 1999) para quienes estén interesados.

Para obtener un verdadero conocimiento resulta imprescindible experimentar con imágenes y operaciones. Para ello pueden emplearse las funciones de OpenCV, con la ventaja de poder incorporarlas fácilmente a programas escritos en diversos lenguajes. Otra opción, más cómoda aún, es el empleo de Matlab ® y su *Image Processing Toolbox*. En este último caso se cuenta con abundante información del propio fabricante, con programas demo y explicaciones, hay libros de texto de procesamiento de imágenes con Matlab (González y col. 2009) y recomendamos, por su facilidad e inmediatez, el uso de Image Morphology de Brett Shoelson. Este es un entorno interactivo desarrollado en Matlab ® con el que probar el funcionamiento de las principales operaciones morfológicas.

5.6. Bibliografía

Comer M.L, Delp E. (1999) Morphological Operations for Color Image Processing Journal of Electronic Imaging 03/1999; pp 279-289.

De la Escalera A. (2001). Visión Por Computador. Fundamentos y Métodos. 1ª Ed Ed Prentica Hall. Madrid, España pp 221-238.

González R.C; Woods R.E. Eddins S.L. (2009) Digital Image Processing Using Matlab (2nd Edition). Gatesmark Publishing. pp 486-534.

Haralick R; Sternberg S.; Zhuang 5.(1987). Image analysis using mathematical morphology, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 9, no. 4, July 1987. pp. 532-550.

Matheron G; Serra J (2002) Mathematical Morphology. Proceedings of the VI International Symposium ISMM 2002. Ed Hugues Talbot and Richard Beare.

Pajares, G.; De la Cruz ,J. M. (2001) Visón por Computador Imágenes Digitales y Aplicaciones, 1ª ed; Editorial Ra-Ma: Madrid, España, pp. 267-294.

Raid A.M.; Khedr W.M.; El-dosuky M.A.; Mona A.(2014) Image Restoration Based On Morphological Operations. International Journal of Computer Science, Engineering and Information Technology (IJCSEIT), Vol. 4, No.3, June 2014 .

Serra J; Vincent L. (1992) An Overview of Morphological Filtering . Circuits Systems and Signal Processing. vol 11, No 1 pp 47-108 January 1992.

Shoelson B. Image Morphology (2009) http://www.mathworks.com/matlabcentral/fileexchange/23697-image-morphology (18-5-2015).

Vincent L. (1993). Morphological Grayscale Reconstruction in Image Analysis: Applications and Efficient Algorithms. IEEE Transactions on Image Processing vol 2 Nº 2 April 1993. pp 176-201.

Xiang B; Latecki, L.J. ; Wen-yu L. (2007) Skeleton Pruning by Contour Partitioning with Discrete Curve Evolution. Pattern Analysis and Machine Intelligence vol29 march 2007 ieeexplore.ieee.org, pp 449-462.
Zhang T. ; Suen C. (1984). A Fast Parallel Algorithm for Thinning Digital Patterns. Communications of the ACM vol 27 Number 3 March 1984 pp 236-239

CAPÍTULO 6

SEGMENTACIÓN DE REGIONES

María GUIJARRO[1], Pedro J. HERRERA[2], Martín MONTALVO[1]

[1] Dpto. Arquitectura de Computadores y Automática, Facultad de Informática, Universidad Complutense, Madrid, España
[2] Escuela Politécnica Superior, Universidad Francisco de Vitoria, Madrid, España

En este capítulo se tratan diferentes técnicas de segmentación de regiones con las que extraer información de una imagen. Tanto si nos enfrentamos a una imagen a color, como si está en escala de grises, los píxeles y sobre todo las agrupaciones de píxeles que la forman nos darán información acerca de las regiones que podemos considerar de interés. Es por ello que las técnicas que se estudian en este capítulo, incluyendo principalmente las de binarización, tienen una importancia relevante, ya que a través de ellas se obtienen, esto es se segmentan, los diferentes objetos de interés en la escena.

6.1. Introducción

Cuando trabajamos con imágenes se hace imprescindible la extracción de información. Considerando que una imagen es un conjunto de píxeles, siendo éstos las unidades que la definen, podemos, a través de ellos extraer información por su posición y su nivel de intensidad.

En este capítulo se estudian diferentes técnicas para la extracción de información de una imagen considerando las regiones de interés, para las cuales hemos tomado como referencia métodos propuestos en Pajares y Cruz (2007).

Si partimos de una imagen como la que se aprecia en la figura 6.1(*a*), vemos que nos interesaría binarizar la imagen extrayendo los diferentes objetos, en este caso rotuladores, que aparecen en ella sobre un fondo homogéneo, resultado que se observa en la figura 6.1(*b*).

(*a*) (*b*)

Figura 6.1.(*a*) Imagen original. (*b*) Imagen resultante binarizada.

Con este fin se proponen las diferentes técnicas que se describen a lo largo del presente capítulo. En la sección 6.2 se toman como referencia los píxeles que se encuentran en una determinada región, que presentan una distribución de intensidad similar, por lo que el histograma de niveles de gris nos dirá qué zonas pertenecen a qué regiones. Si trabajásemos con imágenes a color veríamos que el método sigue siendo aplicable ya que se pueden identificar las regiones de interés mediante los histogramas de los tres canales espectrales. En la sección 6.3, bajo la hipótesis de que una región está limitada por sus bordes, se incluye un algoritmo de etiquetado de componentes conexas que asigna el mismo valor numérico (etiqueta) a cada área delimitada por un borde cerrado como una región de interés. En la sección 6.4 se describe la técnica de crecimiento y división de forma que estableciendo una determinada propiedad, que cumplan las regiones de interés, determinar cuáles son y dónde se sitúan dichas regiones. Por último, la sección 6.5 muestra cómo a partir del color es posible identificar las regiones existentes en la imagen, en función de las propiedades definidas al efecto de las componentes espectrales.

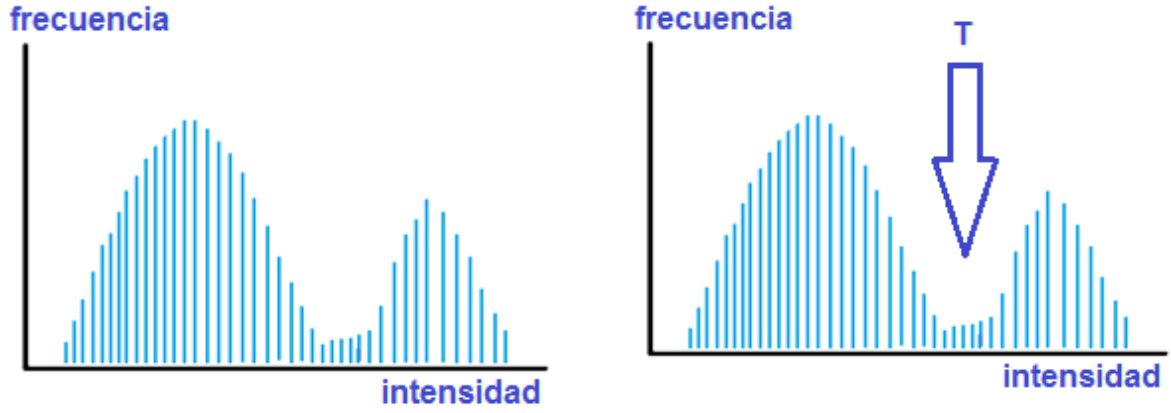
6.2. Binarización por umbral

En esta sección se describe la utilización de umbrales para poder obtener una binarización de las imágenes que extraiga la mayor información posible (Fu y col., 1988; Gonzalez y Woods, 1993, Escalera, 2001).

Desde el punto de vista del tratamiento de imágenes, un umbral se identifica como un valor de intensidad a partir del cual un grupo determinado de píxeles serán considerados como pertenecientes a un subconjunto determinado y catalogados como *blancos*, mientras que el resto se asignan a un segundo subconjunto, siendo en este caso etiquetados como *negros* (Ajenjo, 1993).

El histograma de la imagen constituye el punto de partida para la detección de un umbral, ya que es bien sabido que un histograma se define como la frecuencia relativa de los niveles de intensidad de una imagen. Bajo esta definición podemos ver que el histograma de intensidad de la figura 6.2 correspondiente a una imagen dada *f(x,y)*, está compuesto por dos lóbulos bien diferenciados en

niveles de intensidad marcando dos subconjuntos, que se corresponden en última instancia con dos tipos de zonas u objetos en la imagen. En este ejemplo resulta muy clara la identificación de un valor del umbral, siendo exactamente el valor de intensidad que separa los dos lóbulos del histograma.



(*a*) (*b*)

Figura 6.2.(*a*) Histogramas de intensidad. (*b*) Identificación del valor umbral *T* en el histograma de intensidad.

El valor *T*, que representa en nuestro ejemplo el umbral, es el valor de intensidad que separa los dos modos dominantes permitiendo así la identificación de objetos en la imagen. Suponiendo que los objetos poseen valores de intensidad mayores que los del entorno, correspondiéndose con el histograma mostrado, la binarización se realiza atendiendo a si un píxel (x,y) cumple que el valor de intensidad de ese píxel en la imagen f es mayor que el umbral, $f(x,y) > T$, entonces dicho píxel será considerado como del objeto; en caso contrario, dicho punto será considerado del entorno.

De manera formal el valor de un umbral T se fija a partir de una operación de la forma:

$$T = T[x, y, p(x,y), f(x,y)] \tag{6.1}$$

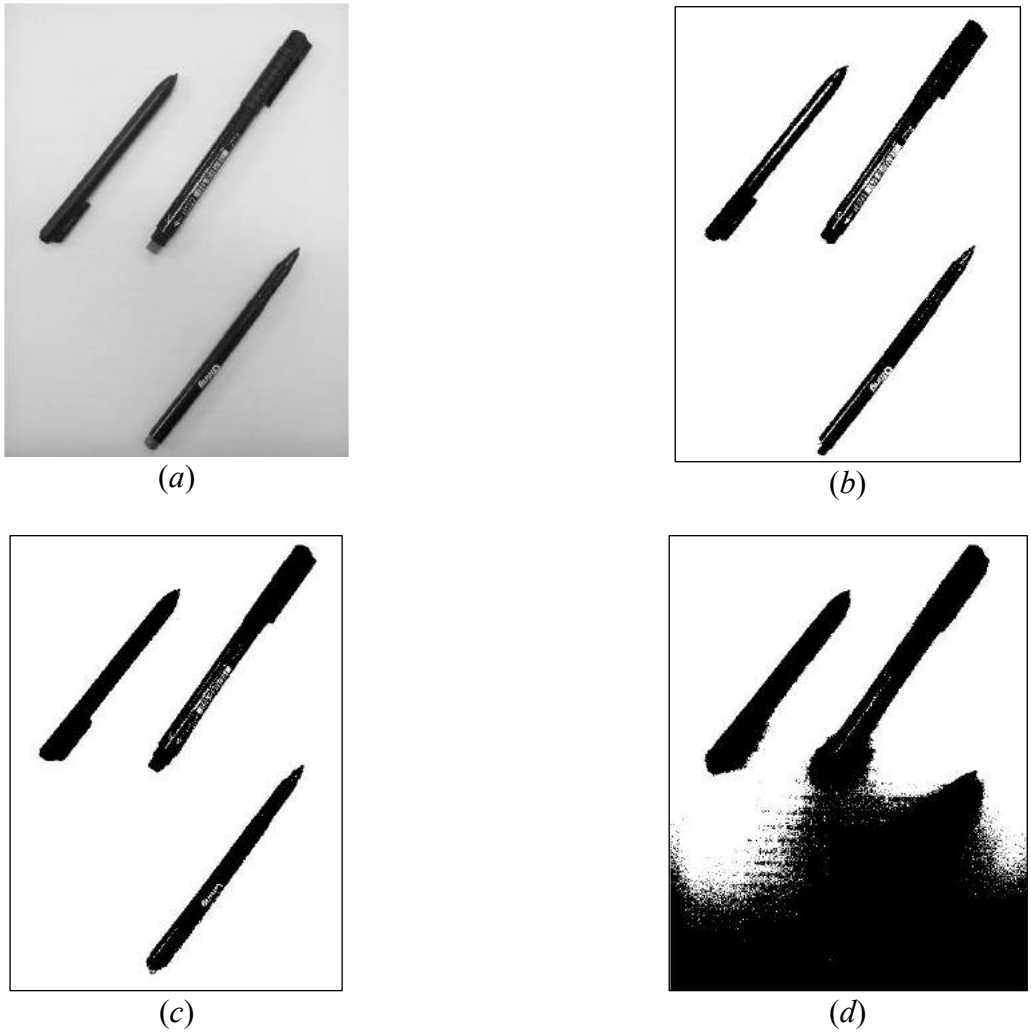
donde x e y son las coordenadas del píxel en la imagen f, $f(x,y)$ es la intensidad del píxel en dichas coordenadas, $p(x,y)$ representa alguna propiedad local del punto que vamos a tener en cuenta para la discriminación, por ejemplo la diferencia de intensidad del píxel central frente a los que le rodean. A partir de este umbral obtenido mediante la ecuación (6.1), se obtiene una imagen binaria $g(x,y)$ definiendo,

$$g(x,y) = \begin{cases} 0 & si \quad f(x,y) > T \\ 1 & si \quad f(x,y) \le T \end{cases} \tag{6.2}$$

Examinando $g(x,y)$ se observa que los píxeles a los que se les asigna un valor 0 corresponden a los objetos, mientras que los que corresponden al entorno tienen valor 1.

De esta manera si consideramos la imagen de partida mostrada en la figura 6.1(*a*), el primer paso será obtener su imagen en escala de grises, tal y como refleja la figura 6.3(*a*). En las figuras 6.3(*b*), (*c*) y (*d*) podemos ver el resultado de segmentar la imagen mostrada en la figura 6.3(*a*) con diferentes

umbrales. De estas imágenes se puede concluir que no todos los umbrales son óptimos y que dependiendo del tipo de información que queramos obtener será mejor un umbral que otro. En efecto, en este ejemplo resulta claro que el resultado mostrado en (*d*) es con diferencia el peor.



(*a*) (*b*) (*c*) (*d*)

Figura 6.3.(*a*) Imagen en escala de grises correspondiente a la figura 6.1(*a*). (*b*) Imagen binarizada con un umbral $T = 0.3$. (*c*) Imagen binarizada con un umbral $T = 0.6$. (*d*) Imagen binarizada con un umbral $T = 0.8$. Estos valores son en la escala de valores de [0,1].

6.2.1. Selección del umbral óptimo

En las imágenes reales no es común encontrarnos con histogramas con sus lóbulos tan diferenciados como el mostrado en la figura 6.2. La propiedad local del punto a tener en cuenta para discriminar no suele aparecer de forma tan nítida, por lo que se hace necesario el estudio de técnicas que nos permitan la obtención de un umbral óptimo. En esta sección vamos a tratar este problema y propondremos una técnica que nos permita su localización.

Los histogramas pueden considerarse formados por la suma de funciones de densidad de probabilidad. Observando la imagen mostrada en la figura6.2 los dos lóbulos de los histogramas señalados se pueden definir como funciones de densidad, que permiten determinar la probabilidad de que un píxel pertenezca o no a un lóbulo, figura 6.4. En el caso del histograma bimodal (aquellos que poseen dos lóbulos), la función global que aproxima el histograma viene dada por,

$$p(z) = P_1 p_1(z) + P_2 p_2(z) \tag{6.3}$$

donde z es una variable aleatoria que representa la intensidad, $p_1(z)$ y $p_2(z)$ son las funciones de densidad de probabilidad y P_1 y P_2 son las *probabilidades a priori*. Estas dos últimas son las que nos proporcionan la probabilidad de ocurrencia de los dos tipos de niveles de intensidad.

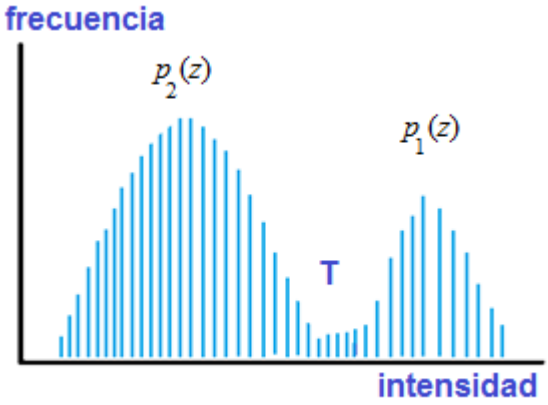


Figura 6.4.Histograma de intensidad como suma de dos funciones de densidad de probabilidad.

Suponiendo las dos partes de la ecuación (6.3) como funciones de z,

$$d_1(z) = P_1 p_1(z) \quad \text{y} \quad d_2(z) = P_2 p_2(z) \tag{6.4}$$

tomaremos la decisión de clasificar el píxel como un objeto si$d_1(z) > d_2(z)$ o como píxel de fondo en caso contrario. El valor óptimo que nos permite determinar a partir de qué valor es un objeto o fondo vendrá dado entonces por el valor de z para el cual $d_1(z) = d_2(z)$. Esto es,

$$P_1 p_1(z) = P_2 p_2(z) \tag{6.5}$$

Si suponemos que $p_1(z)$ y $p_2(z)$ son funciones de densidad de probabilidad gaussianas, esto es,

$$p_1(z) = \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left[-\frac{(z-m_1)^2}{2\sigma_1^2}\right]; \quad p_2(z) = \frac{1}{\sqrt{2\pi}\sigma_2} \exp\left[-\frac{(z-m_2)^2}{2\sigma_2^2}\right] \tag{6.6}$$

haciendo $z = T$ en estas expresiones, sustituyendo en (6.5) y simplificando se obtiene la ecuación de segundo grado en T:

$$AT^2 + BT + C = 0 \tag{6.7}$$

donde

$$\begin{aligned} A &= \sigma_1^2 - \sigma_2^2 \\ B &= 2(m_1\sigma_2^2 - m_2\sigma_1^2) \\ C &= \sigma_1^2 m_2^2 - \sigma_2^2 m_1^2 + 2\sigma_1^2\sigma_2^2 \ln\frac{\sigma_2 P_1}{\sigma_2 P_2} \end{aligned} \tag{6.8}$$

Cuando resolvemos la ecuación de segundo grado (6.7) se pueden obtener dos posibles soluciones, lo cual nos indicaría que se pueden requerir dos valores de umbral para obtener una solución óptima. En cambio si las desviaciones estándar son iguales $\sigma_1 = \sigma_2 = \sigma$, será suficiente un umbral único.

$$T = \frac{m_1 + m_2}{2} + \frac{\sigma^2}{m_1 - m_2} \ln \frac{P_2}{P_1} \tag{6.9}$$

Analizando la ecuación (6.9) vemos que si $\sigma = 0$ o $P_1 = P_2$, el nivel óptimo es el valor medio de las respectivas mediasm1y m2 La primera condición nos indica que tanto el objeto como el fondo tienen intensidades constantes en toda la imagen. Por otro lado, la condición de que $P_1 = P_2$significa que los píxeles del objeto y del entorno tienen igual probabilidad de ocurrir, lo cual se da siempre que el número de píxeles del objeto sea igual al número de píxeles del entorno en una imagen.

El procedimiento desarrollado anteriormente es aplicable a la selección de múltiples niveles, si bien aplicando el procedimiento en sucesivas etapas una vez que se han separado los dos lóbulos principales se procede a la separación en cada uno de ellos, y así sucesivamente hasta el nivel que se desee.

6.2.2. Selección de umbral basada en características de la frontera

Con lo visto hasta ahora, si nos enfrentamos a un histograma con sus picos bien diferenciados, el umbral que los separe será considerado el mejor (Tou y González, 1974; Fu y Mui, 1981). Si bien este no es el caso real en la mayoría de las imágenes.

Siguiendo con la imagen de la figura 6.1(a), vemos que su histograma de intensidad es el representado en la figura 6.5.

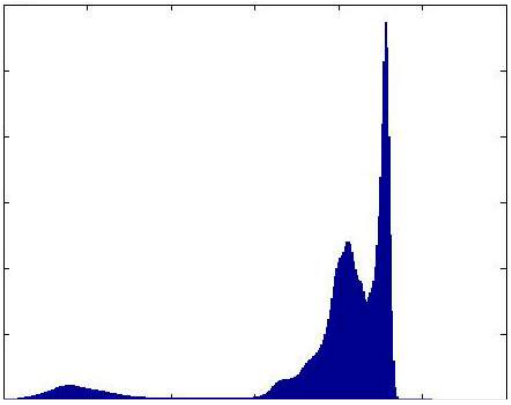


Figura 6.5. Histograma de intensidad de la figura 6.1(*a*).

Este histograma no tiene los picos claramente separados, lo cual hace difícil la obtención de un umbral de manera clara. Por esto vamos a intentar mejorar el histograma teniendo en cuenta sólo aquellos píxeles que están cerca o pertenecen a la frontera de los objetos y el fondo, consiguiendo así hacer al histograma menos dependiente del tamaño relativo. En el histograma de la figura 6.5 podemos ver cómo el pico donde mayor cantidad de píxeles hay se corresponden con aquellas intensidades relacionadas con el fondo de la imagen, ya que constituyen el grueso de la misma. En este histograma no somos capaces de determinar con claridad dónde estarían situados los píxeles de los objetos en

cuestión. Si en el histograma tomásemos sólo los píxeles considerados frontera entre los objetos y el entorno, o aquellos cercanos a la frontera, quedarían resaltados los picos al tener los pesos más equilibrados.

Para conseguir resaltar aquellos píxeles considerados frontera, o que estén cerca de ella vamos a utilizar el gradiente ya que nos permite determinar si un píxel es o no de borde. El gradiente de una imagen f(x,y) en un punto (x,y) se define como un vector bidimensional dado por la ecuación (6.10), siendo un vector perpendicular al borde.

$$\mathbf{G}[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} f(x,y) \\ \frac{\partial}{\partial y} f(x,y) \end{bmatrix} = \begin{bmatrix} \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} \\ \frac{f(y+\Delta y) - f(y-\Delta y)}{2\Delta y} \end{bmatrix} \tag{6.10}$$

Además el uso de la Laplaciana proporciona información sobre si un píxel dado se encuentra en el lado oscuro o claro de la frontera en base al signo que genera su aplicación. Definimos la Laplaciana de una función 2-D f(x,y) como un operador segunda derivada definido como,

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{6.11}$$

Debido a que la Laplaciana es cero en las regiones de intensidad constante, se puede esperar que los valles de histogramas formados con píxeles seleccionados por un criterio de gradiente/Laplaciana estén escasamente poblados. Esta propiedad produce los valles profundos deseados que se han mencionado anteriormente.

Bajo la consideración de estos conceptos se propone la siguiente función con el fin de generar una imagen con los tres niveles de intensidad,

$$s(x,y) = \begin{cases} 0 & \text{si } G[f(x,y)] < T \\ + & \text{si } G[f(x,y)] \geq T \text{ y } L[f(x,y)] \geq 0 \\ - & \text{si } G[f(x,y)] \geq T \text{ y } L[f(x,y)] < 0 \end{cases} \tag{6.12}$$

donde $G[f(x,y)]$ es el gradiente, $L[f(x,y)]$ la Laplaciana, los símbolos 0, + y − representan tres niveles distintos cualesquiera de gris y T es un umbral. Suponiendo que tenemos los objetos oscuros sobre un fondo claro, como en la figura 6.1(a), el uso de la ecuación (6.12) produce una imagen s(x,y) en la cual todos los píxeles que no están en un borde se etiquetan “0”, todos los píxeles del lado oscuro de un borde se etiquetan “+” y todos los píxeles del lado claro de un borde se etiquetan “− “. Los símbolos “+” y “−” en la ecuación anterior se intercambian de posición para un objeto claro sobre un fondo oscuro.

La información que se obtiene usando el procedimiento anterior se puede emplear para crear una imagen binaria segmentada, en la cual los objetos de interés se representan con ceros (negro) y el fondo con valores del nivel de gris 255 (blanco).

Con el fin de obtener esta imagen binaria vamos a estudiar los cambios que se producen:

"–" "+" en s(x,y) cuando pasamos de fondo claro a un objeto oscuro.

"+" o "0" en el interior de un objeto.

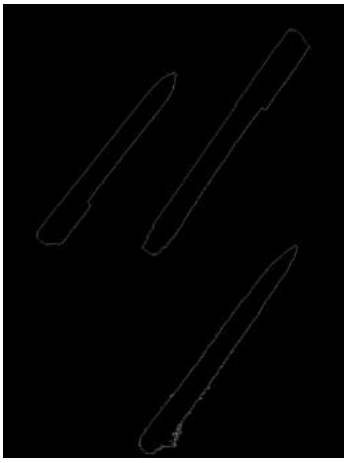
"+" "–" en s(x,y) cuando pasamos de un objeto al fondo.

Así sabemos que una línea de exploración que contiene una sección de un objeto tiene la siguiente estructura:

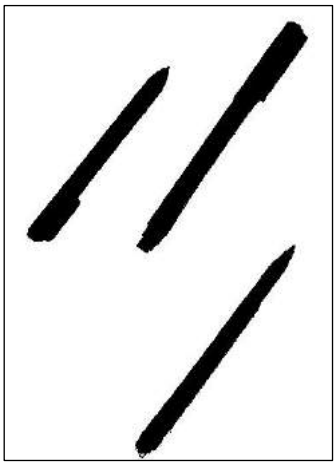
$$(...)(-,+)(0\text{ o }+)(+,-)(...) \tag{6.13}$$

donde (…) representa cualquier combinación de +, − o 0. El paréntesis central contiene puntos de objetos y se marca como 255. Los demás píxeles que se encuentran en la misma línea de exploración se etiquetan 0, con la excepción de cualquier secuencia de (0 o +) limitado por (–,+) y (+, –).

La aplicación de este algoritmo sobre la imagen de la figura 6.1(*a*) se puede ver en las figuras6.6(*a*) y (*b*).



(*a*)



(*b*)

Figura 6.6.(*a*) Imagen obtenida por aplicación de la ecuación (6.12) sobre la imagen de la figura 6.1(*a*). (*b*) Imagen binaria obtenida aplicando el criterio sintetizado en (6.13); los objetos en negro y el fondo blanco.

6.2.3. Método de Otsu

Otsu (1979) estudió que, al igual que en el caso de las imágenes con los histogramas bimodales, éstos están formados por la suma de dos funciones de densidad de probabilidad gaussianas, ecuaciones (6.3) y (6.6), donde cada gaussiana se aproxima a uno de los lóbulos. Dedujo a raíz de esto, que cuando las gaussianas se asemejan al histograma real, las desviaciones estándar disminuyen haciendo que el mejor umbral a seleccionar sea aquel que minimice la suma de las varianzas de los dos lóbulos del histograma. Expresado de forma matemática, dada una imagen con L niveles de intensidad y asumiendo que el umbral buscado es T, las probabilidades acumuladas hasta T y desde T hasta L resultan ser,

$$w_1(t) = \sum_{z=1}^{T} P(z) \quad \text{y} \quad w_2(t) = \sum_{z=T+1}^{L} P(z) \tag{6.14}$$

Obtenemos las medias y varianzas asociadas,

$$\mu_1(t) = \sum_{z=1}^{T} zP(z) \quad \text{y} \quad \mu_2(t) = \sum_{z=T+1}^{L} zP(z) \tag{6.15}$$

$$\sigma_1^2(t) = \sum_{z=1}^{T} (z - \mu_1(t))^2 \frac{P(z)}{w_1(t)} \quad \text{y} \quad \sigma_2^2(t) = \sum_{z=T+1}^{L} (z - \mu_2(t))^2 \frac{P(z)}{w_2(t)} \tag{6.16}$$

Finalmente se obtiene la varianza ponderada,

$$\sigma_w^2(t) = w_1(t)\sigma_1^2(t) + w_2(t)\sigma_2^2(t) \tag{6.17}$$

Se elige el umbral T correspondiente al nivel de intensidad que proporcione la mínima varianza ponderada definida en (6.17).En la figura 6.7se muestra el resultado de la binarización sobre la imagen de la figura 6.1(*a*) utilizando el método de Otsu, obteniéndose en este caso un valor de umbral T = 0.505, que transformado a la escala [0, 255] resulta ser 129.

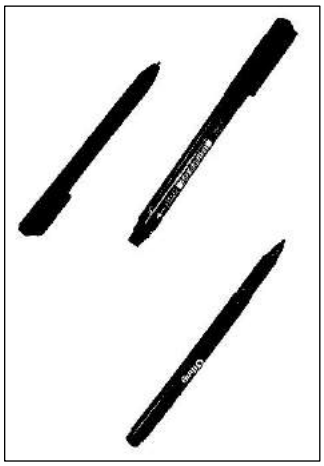


Figura 6.7. Binarización de la imagen de la figura 6.1(*a*) mediante el método de Otsu.

6.2.4. Método de Ridler-Calvard

Como en el caso anterior, Ridler y Calvard (1978) definieron un algoritmo iterativo con el fin de encontrar el mejor umbral. Los pasos de que consta son:

1) Iteración k = 0, calcular el valor medio de la imagen $T(k) = m$. Este valor será considerado como la inicialización del umbral. Este valor medio determina dos clases w_1 y w_2 formadas por los píxeles cuya intensidad es menor y mayor, respectivamente.
2) Iteración k = 1, para las dos nuevas clases determinar los valores medios de cada clase m_1 y m_2, obteniendo $T(k) = (m_1 + m_2)/2$.
3) El nuevo umbral determina la existencia de otras dos nuevas clases como en el paso 2); para cada iteración, incluidas la 0 y 1; mientras $|T(k+1) - T(k)| \geq \varepsilon$repetir las acciones del paso 2), consideramos ε como la distancia mínima entre los dos umbrales obtenidos, entre la iteración actual y la anterior, con la que se puede considerar que son prácticamente iguales.

La figura 6.8muestra el resultado de aplicar el método de Ridler-Calvard sobre la imagen de la figura 6.1(*a*), obteniéndose un valor umbral T = 0.512 o equivalentemente de 131 en la escala [0,255].

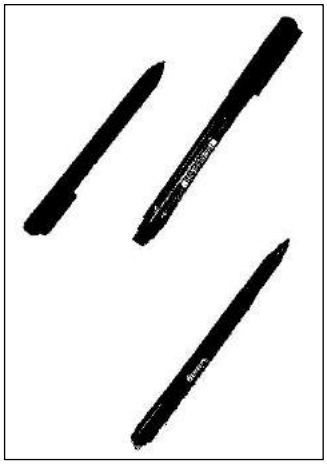


Figura6.8. Binarización de la imagen de la figura 6.1(*a*) mediante el método de Ridler-Calvard.

6.3. Etiquetado de componentes conexas

Consideramos componentes conexas aquellas partes de la imagen entre cuyos píxeles siempre existe un camino de forma que se pueda ir de uno a otro sin abandonar la región que los contiene. Si la imagen es binaria se puede determinar la homogeneidad por el valor de sus píxeles. Así, si partimos de una imagen binarizada, en blanco o negro (ceros o unos), las componentes conexas serán aquellas regiones o áreas cuyos píxeles posean por ejemplo un valor de "1" y estén conectados entre sí por un camino o conjunto de píxeles del mismo valor. A estos píxeles se les asignarán la misma etiqueta identificativa, que debe ser única, constituyendo así su identificador.

A continuación se describe un algoritmo iterativo clásico de etiquetado de componentes conexas (Haralick y Shapiro, 1993).

El algoritmo iterativo consta de un paso de inicialización. A partir de ahí comienza una secuencia de propagación de etiquetas de arriba-abajo seguida por una propagación de etiquetas de abajo-arriba de forma iterativa, hasta que no haya más cambios de etiquetas.

Dicho algoritmo se expresa en forma de pseudo-código por su mayor claridad y reutilización. Se dispone de una imagen binaria I. La función *etiqueta* (*línea,píxel*) será aquella que marque un píxel con una determinada etiqueta. La función *nueva_etiqueta* () genera una nueva etiqueta con valor entero cada vez que es llamada. La función *vecinos* () devuelve el conjunto de los vecinos ya etiquetados de un determinado píxel en su misma línea o en la línea previa. La función *etiquetas* (), cuando se le proporciona el conjunto de píxeles ya etiquetados, devuelve el conjunto de sus etiquetas. Finalmente, la función *min* () cuando se le proporciona un conjunto de etiquetas devuelve la mínima etiqueta (Pajares y Cruz, 2007).

```
Procedimiento iterar
    //inicialización de cada píxel de valor 1 a una única etiqueta//
    para cada línea lhacer
    para cada píxel phacer
    si I(l,p) == 1 entonces
        etiqueta (l,p) = nueva_etiqueta ()
```

sino
etiqueta (l,p) = 0
fin si
fin para
fin para
//Iteración de arriba-abajo seguida por pasos de abajo-arriba//
repetir hasta que la variable CAMBIO sea FALSO
//"Paso de arriba-abajo"//
cambio= falso
para cada línea l *hacer*
para cada píxel p *hacer*
si etiqueta (l,p) $\neq 0$ *entonces*
m = min (etiquetas (vecinos ((l,p)) $\cup$ (l,p)));
si m $\neq$ etiqueta(l,p) *entonces*
cambio= verdadero
etiqueta(l,p) = m
fin si
fin si
fin para
fin para
//"Paso de abajo-arriba"//
para cada línea l *hacer*
para cada píxel p *hacer*
si etiqueta (l,p) $\neq 0$ *entonces*
m = min(etiquetas (vecinos ((l,p)) $\cup$ (l,p)));
si m $\neq$ etiqueta (l,p) entonces
cambio= verdadero
etiqueta (l,p) = m
fin si
fin si
fin para
fin para
Fin Iterar

Para entender mejor el algoritmo consideremos un sencillo ejemplo. En la figura 6.9 podemos ver el proceso del algoritmo iterativo donde partimos de una imagen binaria (*a*) con dos partes diferenciadas; en (*b*) se pueden ver los resultados después de la inicialización de cada píxel con valor 1 a una única etiqueta; (*c*) muestra el resultado después de la primera pasada de arriba-abajo, en la cual el valor de cada píxel distinto de cero es reemplazado por el mínimo de sus vecinos distintos de cero, de una

forma recursiva de izquierda a derecha y de arriba abajo; (d) resultados después del primer paso abajo-arriba.

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (a) \qquad \begin{bmatrix} 0 & 1 & 2 & 0 & 3 & 4 & 0 \\ 0 & 5 & 6 & 0 & 7 & 8 & 0 \\ 0 & 9 & 10 & 11 & 12 & 13 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 14 & 15 & 16 & 0 & 0 & 0 \\ 0 & 17 & 18 & 19 & 0 & 0 & 0 \end{bmatrix} \quad (b)$$

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 2 & 2 & 0 \\ 0 & 1 & 1 & 0 & 2 & 2 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 14 & 14 & 14 & 0 & 0 & 0 \\ 0 & 14 & 14 & 14 & 0 & 0 & 0 \end{bmatrix} \quad (c) \qquad \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 14 & 14 & 14 & 0 & 0 & 0 \\ 0 & 14 & 14 & 14 & 0 & 0 & 0 \end{bmatrix} \quad (d)$$

Figura 6.9. Algoritmo iterativo para el etiquetado de componentes conexas.

Siguiendo con el ejemplo utilizado en este capítulo, 6.1(*a*), si aplicamos el algoritmo de etiquetado de componentes conexas iterativo a la imagen binaria mostrada en la figura 6.10(*a*), obtenemos tres componentes como se puede ver en la figura 6.10(*b*). Con el fin de facilitar la visualización del etiquetado se ha puesto cada etiqueta de un color, como se aprecia en la imagen.

(*a*) (*b*)

Figura 6.10.(*a*) Imagen binaria. (*b*) Resultado obtenido por el algoritmo iterativo para el etiquetado de componentes conexas, aplicado sobre la imagen binaria en (*a*).

6.4. Crecimiento y división

Como se ha indicado previamente, las técnicas de segmentación permiten encontrar regiones en la imagen. Además de las técnicas ya vistas, a continuación se describe la que se conoce como de crecimiento y división (Fu y col., 1988; Sonka y col., 1995; Gonzalez y Woods, 1993; Coiras y col., 1998; Shen y col., 1998).

Para la aplicación de estas técnicas se definen las propiedades topológicas y de conjunto que definen las regiones, donde *R*es la región que incluye la imagen completa. Para la segmentación partimos de un proceso que divide *R* en *n* subregiones cumpliéndose:

1) *La unión de todas las regiones forman la región inicial. La segmentación debe ser completa situándose todos los píxeles en alguna región.*
2) *Todas las regiones son regiones conectadas, estando todos los píxeles de una región conectados.*
3) *La intersección entre cualesquiera dos regiones diferentes es vacía.*
4) *Todos los píxeles de una región cumplen determinadas propiedades.*
5) *Los píxeles de regiones diferentes no tienen que cumplir las mismas propiedades.*

6.4.1 Crecimiento o unión de regiones

Esta técnica consiste en agrupar píxeles para formar regiones que tengan características similares. Comenzamos estableciendo las *semillas*, que serán aquellos píxeles iniciales a partir de los cuales vaya creciendo la región. A estos píxeles se les van añadiendo aquellos adyacentes que cumplan alguna propiedad similar a la de la semilla, pasando a formar parte de la misma región. Con el fin de encontrar estos píxeles adyacentes definiremos los criterios de agregación. Estos criterios suelen estar basados en intensidades, colores o texturas similares de los píxeles vecinos de la región con respecto a los de la semilla. De esta manera, una vez encontremos píxeles que no cumplan estos criterios se dará por finalizado el crecimiento de la región.

Dependiendo de la naturaleza del problema tendremos que tener en cuenta el criterio a utilizar para seleccionar las semillas, los criterios de agregación y las restricciones de parada en el crecimiento de la región.

6.4.2 División de regiones

Este proceso surge de la misma idea que el proceso anterior pero de manera opuesta. Partimos de una única región, que será toda la imagen. Definimos un criterio de homogeneidad según la naturaleza de nuestro problema. Si la región de la que hemos partido no cumple dicho criterio la dividimos en subregiones de las cuales estudiaremos su homogeneidad bajo este criterio. Una vez encontremos que los píxeles de una región cumplen el criterio de homogeneidad dejará de dividirse.

Estos criterios de homogeneidad son similares a los criterios de agregación vistos en la sección anterior; a veces sólo difiere la dirección en la que se aplican.

6.5. Extracción de regiones por el color

La extracción de regiones utilizando el color es una técnica muy utilizada a la par que factible. Las imágenes que extraemos en su mayoría, son imágenes a color. Esto nos motiva a diseñar diferentes técnicas en las que haciendo uso de las componentes espectrales RGB seamos capaces de discriminar unas regiones de otras (Ohta, 1985; Gevers y Smeulders, 1999; Luong, 1995; Hoover y col., 1996; Escalera, 2001).

Basándonos en el modelo de color RGB se pueden extraer de la imagen aquellas regiones en las que predomine una determinada componente de color. Por ejemplo dada la imagen de la figura 6.11(*a*) donde podemos apreciar un campo de cultivo, queremos extraer las regiones correspondientes a las plantas. Para ello, será la componente G correspondiente al verde, la que siendo predominante marque dicha región. De tal forma, siguiendo este ejemplo, aquellos píxeles cuya componente G sea mayor que la R y que la B serán marcados como blancos y el resto negros.

(*a*)

(*b*)

Figura6.11.(*a*) Imagen RGB. (*b*) Binarización de la imagen (*a*) por medio del color.

Se puede utilizar cualquier otro criterio, por ejemplo que las componentes de color sobrepasen un determinado umbral, o que una de ellas sea superior a las otras en un determinado porcentaje, etc.

6.6. Bibliografía

Ajenjo, A.D. (1993) *Tratamiento Digital de Imágenes*; Anaya, Madrid, España.

Coiras, E., Santamaría, J. Miravet, C. (1998) Hexadecagonal region growing. *Pattern Recognition Letters*, vol. 19(12), pp. 1111-1117.

De la Escalera, A. (2001) *Visión por Computador: fundamentos y métodos*; Prentice Hall, Madrid, España.

Fu, K.S., Mui, J.K. (1981) A Survey of Image Segmentation. *Pattern Recognition*, vol. 13(1), pp. 3-16.

Fu, K.S., González, R.C., Lee, C.S.G. (1988)*Robótica: Control, Detección, Visión e Inteligencia*; McGraw-Hill, Madrid, España.

Gevers, T.,Smeulders, A.W.M. (1999) Color-based object recognition. *Pattern Recognition*, vol. 32(3), pp. 453-464.

Gonzalez, R.C.,Woods, R.E. (2008)*Digital Image Processing*; Addison-Wesley, Reading, USA.

Haralick, R.M.,Shapiro, L.G. (1993)*Computer and Robot Vision*, vol. II; Addison-Wesley, Reading, USA.

Hoover, A., Jean-Baptiste, G., Xiang, X., Flynn, P.J., Bunke, H., Goldgof, D.B., Bowyer, K., Eggert, D.W., Fitzgibbon, A.W.,Fisher, R.B. (1996) An Experimental Comparison of Range Image

Segmentation Algorithm. *IEEE Transactionson Pattern Analysisand Machine Intelligence*,vol. 18(7), pp. 673-689.

Luong, Q.T. (1995) Color in Computer Vision. En *Handbook of Pattern Recognition and Computer Vision*;Chen, C.H.; Pau, L.F.; Wang, P.S.P., Eds.; World Scientific Publishing Co., Singapore;pp. 311-368.

Ohta, Y.,Kanade, T. (1985) Stereo by intra- and inter-scanline search. *IEEE Transactionson Pattern Analysisand Machine Intelligence*, vol. 7(2), pp. 139-154.

Otsu, N. (1979). A threshold selection method from gray-level-histograms. *IEEE Transactionson Systems, Man, and Cybernetics*,vol SMC-9, pp. 62-66.

Pajares, G., de la Cruz, J.M. (2007)*Visión por computador: imágenes digitales y aplicaciones*, 2ª ed.;RA-MA, Madrid, España.

Ridler, T.W., Calvard, S. (1978) Picture thresholding using an iterative selection method.*IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-8, pp. 630-632.

Shen, X., Spann, M. Nacken, P. (1998) Segmentation of 2D and 3D images through a hierarchical clustering based on region modelling. *Pattern Recognition*, vol. 31(9), pp. 1295-1309.

Sonka, M., Hlavac, V. Boyle, R. (1995)*Image Processing, Analysis and Machine Vision*; Chapman &Hall, Cambridge, UK.

Tou, J.T., González, R.C. (1974)*Pattern Recognition Principles*; Addison-Wesley, Reading, USA.

CAPÍTULO 7

DESCRIPCIÓN DE TEXTURA EN IMÁGENES UTILIZANDO LOCAL BINARY PATTERN (LBP)

Oscar GARCÍA-OLALLA[1], Enrique ALEGRE[1]

[1] Universidad de León, Departamento de Ingeniería eléctrica y de Sistemas y Automática, Escuela de Ingenierías, León, España.

En este capítulo vamos a estudiar uno de los descriptores de textura más utilizados en la actualidad, llamado Local Binary Pattern (LBP) o patrón binario local, que tiene numerosas aplicaciones en el análisis de imágenes digitales y, en general, en la visión por computador. Previamente, se introducirá el concepto de textura en una imagen digital y los tipos de descriptores que se utilizan para caracterizarla. Posteriormente se explicará con detalle qué es y cómo se obtiene un Patrón Binario Local (LBP), algunas variaciones en su cálculo que fueron propuestas como extensiones al descriptor básico y tres métodos derivados del LBP como son el ALBP (Adaptive Local Binary Pattern), el LBPV (Local Binary Pattern Variance) y el CLBP (Completed Local Binary Pattern). Finalmente se comentan algunas aplicaciones tanto del LBP como de sus métodos derivados.

7.1. Introducción

7.1.1. Concepto de textura

A pesar de que su estudio es un campo de investigación importante en visión por computador, no hay una descripción formal de lo que es una textura. El principal motivo para que la definición de textura no sea universal, es que las características propias de una textura a menudo son opuestas en diferentes tipos de imágenes: Regularidad frente a aleatoriedad, uniformidad frente a heterogeneidad,… A lo largo de los últimos años, distintos investigadores han tratado de definir la textura desde una perspectiva concreta de su naturaleza. Haralick (1979), considera una textura como una "***región organizada que puede ser descompuesta en primitivas teniendo unas distribuciones espaciales concretas***". Esta definición, conocida como la aproximación estructural, se basa en la idea de que cada textura está compuesta por una serie de elementos básicos (puntos, aristas, etc.), similar a

la que utiliza el ser humano. Alternativamente, Cross y Jain (1983) definieron textura, desde un punto de vista estocástico, como "***una región de la imagen bidimensional, aleatoria y posiblemente periódica***". Además de las definiciones clásicas, se pueden encontrar otras más orientadas al análisis de imágenes, en las que se define una textura como "***la distribución espacial de color o intensidad en una imagen o en una región de la misma"***. (Zhou, 2006)

7.1.2. Tipos de textura

En la actualidad, las texturas se pueden clasificar en diferentes categorías dependiendo del problema que se desee resolver (Pietikäinen y Zhao, 2009), como son:

- Microtexturas o Macrotexturas.
- Irregulares o regulares.
- Repetitivas o no repetitivas.
- Con alto contraste y no direccionales frente a bajo contraste y direccionales
- Granulada y de baja complejidad frente a no granuladas y de alta complejidad.

Estas tres últimas categorías pertenecen a las tres dimensiones ortogonales de las texturas propuestas por Rao y Lohse en 1993.

7.1.3. Características de los descriptores de textura

Debido a la alta complejidad y variabilidad de las texturas, es muy difícil crear un descriptor que se adapte a todas ellas y sea capaz de identificarlas a la perfección. Sin embargo, un buen descriptor de textura debe cumplir los siguientes requisitos:

1. Eficiente, diferenciando entre múltiples tipos de texturas.
2. Robusto frente a variaciones en el escalado y la pose.
3. Robusto frente a variaciones en la iluminación.
4. Robusto frente a la falta de uniformidad espacial.
5. Debe funcionar para imágenes de pequeño tamaño.
6. Debe tener baja complejidad computacional.

Sin embargo, es muy difícil encontrar técnicas de descripción de texturas que cumplan con todas estas características. Es por ello que en la actualidad hay gran actividad investigadora en este campo y cada año aparecen nuevas técnicas que intentan satisfacer los requisitos anteriores.

7.1.4. Tipos de descriptores de textura

Los descriptores de textura pueden clasificarse en diferentes tipos en función de las propiedades de la imagen que se tienen en cuenta a la hora de obtener sus características. Según Tuceryan y Jain (1993) los descriptores de textura se dividen en las siguientes categorías:

- **Estructurales**: Representan la textura por medio de una jerarquía y primitivas bien definidas. Para ello primero hay que especificar las primitivas y luego las reglas de posicionamiento. Estos métodos tratan de expresar de manera rigurosa la estructura de la región, por lo que funcionan mejor en texturas regulares y repetitivas, mientras que se comportan peor cuando se tratan de texturas naturales debido a la alta variabilidad que presentan.

- **Estadísticos**: Estas técnicas no pretenden describir la estructura jerárquica de la textura, sino que la representan mediante propiedades basadas en la distribución y la relación entre los niveles de gris de la imagen. Los métodos basados en estadísticas de segundo orden han demostrado obtener buenos resultados siendo el más conocido la matriz de co-ocurrencia (Haralick 1979).

- **Basados en modelos**: Estos descriptores utilizan fractales y modelos aleatorios para describir la estructura de la imagen, ajustando los parámetros de dichos modelos hasta encontrar los que mejor representan a la textura. La estimación de los parámetros óptimos conlleva una complejidad computacional elevada, por lo que no son apropiados en sistemas donde el tiempo es un factor clave.

- **Basados en transformaciones**: Representan la imagen en un espacio cuyo sistema de coordenadas tiene una interpretación que está relacionada con las características propias de la textura como la frecuencia o el tamaño. Técnicas como la transformada de Fourier (Rosenfeld y Weszka, 1980), Gabor (Daugman, 1985) o Wavelet (Mallat, 1989) han sido frecuentemente utilizadas, obteniendo esta última resultados muy interesantes, sobre todo para segmentación basada en textura.

7.2. Métodos de descripción de textura

En Haralick y col. (1973) se propuso un método de descripción de textura basado en la matriz de co-ocurrencia. Esta matriz refleja la distribución de los niveles de gris para dos píxeles situados a una determinada distancia y ángulo. Utilizando como base esta matriz, Haralick extrae un conjunto de 14 momentos para describir la textura. Es la técnica que más se ha utilizado para describir texturas desde que se desarrolló y funciona muy bien con texturas estocásticas, sin embargo su coste computacional es elevado y requiere de una normalización previa del nivel de gris.
En Laws (1980), Laws desarrolló como proyecto de tesis doctoral un descriptor basado en la utilización de nueve máscaras de tamaño 3x3 para realzar características de la imagen como pueden ser bordes, media del nivel de gris, crestas, etc. Las máscaras se obtienen multiplicando entre sí tres vectores:

- Level: L3 = [1 2 1]
- Edge: E3 = [-1 0 1]
- Spot : S3 = [-1 2 -1]

El mayor inconveniente de los descriptores de Laws es el tiempo computacional que requiere realizar la convolución de las máscaras seleccionadas con la imagen original, por lo que este método tampoco sería apropiado para situaciones en las que el tiempo sea una restricción.

Durante más de 40 años, el espectro de Fourier ha sido utilizado para múltiples aplicaciones en el campo del tratamiento digital de imágenes. Diferentes trabajos han demostrado su gran capacidad a la hora de describir patrones bidimensionales periódicos en una imagen, obteniendo resultados que son muy difíciles de lograr con técnicas locales debido a su propia naturaleza. Sin embargo, el uso de esta técnica en la mayoría de texturas ofrece un rendimiento bajo debido a su falta de localización espacial. Los filtros de Gabor proporcionan herramientas para mejorar esta localización espacial. Su respuesta es similar a la del cortex visual del ser humano. Sin embargo, su coste computacional es muy alto y en casos donde las texturas son muy irregulares el rendimiento disminuye. Wavelet es una de las técnicas basadas en transformadas que mejor funciona en el reconocimiento de texturas (Stavros 2009). Esta transformada mapea la imagen en una sub-imagen de baja resolución, normalmente llamada "imagen tendencia" y tres imágenes de detalles para distintas orientaciones. Esta imagen tendencia se obtiene aplicando filtrado paso bajo, lo que produce un efecto de procesos de desenfoque a la imagen original y las imágenes detalladas se obtienen extrayendo la información que se pierde al realizar este proceso. La energía o la desviación típica de las imágenes detalladas proporcionan información muy valiosa a la hora de detectar texturas, que finalmente formarán el descriptor de éstas. La principal ventaja que otorga el uso de Wavelet para describir texturas es que permite realizar análisis multiresolución de la imagen, facilitando la descripción de todo tipo texturas.

7.3 Local Binary Pattern (LBP)

7.3.1. Concepto

Local Binary Pattern (LBP) (Ojala y col. 1994) es un operador de textura simple y eficiente que etiqueta cada píxel de la imagen analizando su vecindario, estudiando si el nivel de gris de cada píxel supera un determinado umbral y codificando dicha comparación mediante un número binario. Debido a su bajo coste computacional y al gran poder discriminativo que ha demostrado tener, LBP se ha convertido en los últimos años en una de las soluciones más utilizadas en numerosas aplicaciones relacionadas con textura. La característica más importante de LBP en aplicaciones reales es la robustez que ofrece frente a variaciones en la intensidad del nivel de gris, causado, entre otras muchas cosas, por diferencias en la iluminación.
Para calcular LBP sobre una imagen en escala de grises se utiliza la ecuación 7.1.

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p, \quad s(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ 0 & \text{si } x < 0 \end{cases} \tag{7.1}$$

Donde P es el número de vecinos que se van a considerar, R es el tamaño del vecindario y, g_c y g_p son los valores de gris del píxel central y cada uno de los p píxeles del vecindario respectivamente.
En la figura 7.1 podemos observar un ejemplo del cálculo del LBP de manera gráfica sobre un píxel cualquiera de una imagen para los parámetros P=8 y R=1.

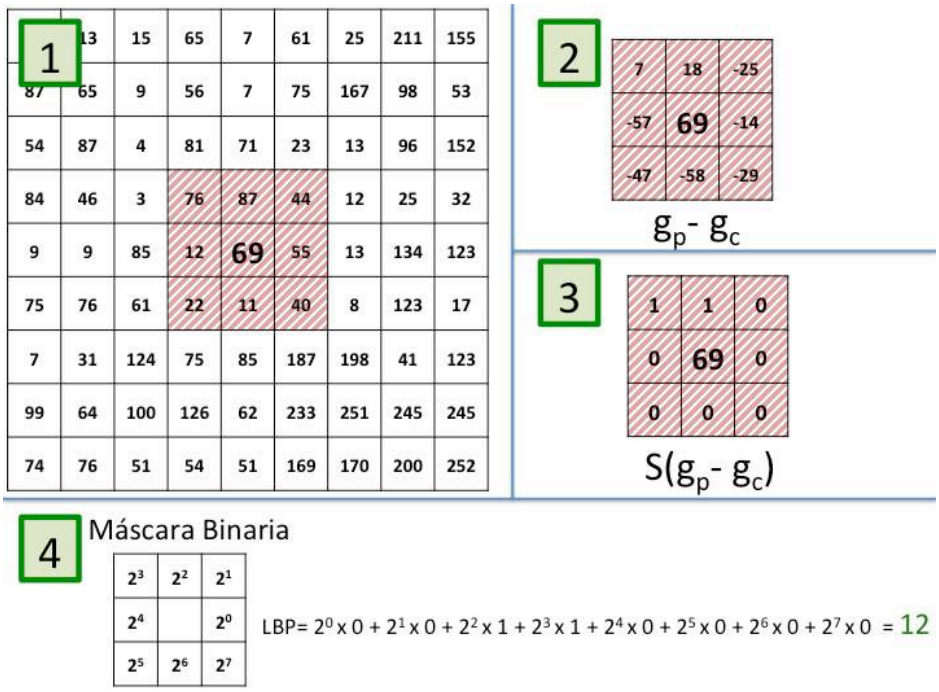


Figura 7.1. Esquema de los pasos necesarios para obtener el valor de LBP de un píxel concreto.

En 2002, Ojala y col. (2002) propusieron una mejora del método, incorporando vecindarios circulares, donde los vecinos se ubicaban igualmente espaciados, realizando una interpolación bilineal en caso de que éstos no coincidieran con el centro de un píxel concreto de la imagen. En el caso de la figura 7.1, el proceso sería idéntico salvo que los valores de los vecinos colocados en las esquinas no se corresponderían con 44, 76, 22 y 40 para los vecinos 1, 3, 5 y 7 respectivamente, ya que serían el resultado de una interpolación entre ese valor de gris y los adyacentes a éstos. En lo sucesivo, cuando se calculen valores de LBP en este capítulo, se hará teniendo en cuenta el vecindario cuadrado debido a que facilita mucho la comprensión por parte del lector.

Una vez obtenido el valor del LBP para cada uno de los píxeles de la imagen, se obtiene el histograma de la imagen, dando como resultado un vector de 256 elementos.

Ejemplo 7.1. *Obtener el histograma LBP para el fragmento de la imagen que se muestra en la figura 7.2 utilizando 13, 24 y 10 bins (agrupaciones) con radio 1 y 8 vecinos (sin considerar interpolaciones).*

	1	2	3	4	5	6	7
1	0	0	0	0	0	0	0
2	0	30	30	20	30	30	0
3	0	30	20	20	20	30	0
4	0	20	10	10	10	20	0
5	0	5	10	10	10	5	0
6	0	5	10	10	10	5	0
7	0	5	10	10	10	5	0

Figura 7.2. Fragmento de imagen en escala de grises.

Solución: El primer paso es determinar la región sobre la que se va a calcular LBP. Al ser un vecindario de radio 1, se debe eliminar del cálculo 1 píxel de cada borde de la imagen. El resultado se puede ver en la figura 7.3.

	1	2	3	4	5	6	7
1							
2		30	30	20	30	30	
3		30	20	20	20	30	
4		20	10	10	10	20	
5		5	10	10	10	5	
6		5	10	10	10	5	
7							

Figura 7.3. Área de la imagen sobre la que se debe calcular LBP

A continuación, en la figura 7.4, se muestran ejemplos de la extracción del valor de LBP para 4 píxeles de la imagen.

Píxel (1,1)									1	2	4	8	16	32	64	128
0	0	0	0	0	0											
0	30	30	0	30	1				0	0	0	1	0	1	0	0
0	30	20	0	1	0	LBP8,1 (1,1) =	**40**		0	0	0	8	0	32	0	0
Píxel (4,4)																
20	20	20	1	1	1											
10	10	10	1	10	1				1	1	1	1	1	1	1	1
10	10	10	1	1	1	LBP8,1 (4,4) =	**255**		1	2	4	8	16	32	64	128
Píxel (6,6)																
10	5	0	1	1	0											
10	5	0	1	5	0				1	1	0	0	0	1	1	1
10	5	0	1	1	0	LBP8,1 (6,6) =	**227**		1	2	0	0	0	32	64	128
Píxel (4,6)																
20	30	0	1	1	0											
10	20	0	0	20	0				1	1	0	0	0	0	0	0
10	5	0	0	0	0	LBP8,1 (4,6) =	**3**		1	2	0	0	0	0	0	0

Figura 7.4. Cálculo detallado de LBP para 4 píxeles de la imagen.

Realizando esta operación para todos los píxeles obtendríamos los valores que se muestran en la figura 7.5:

	Gris	LBP	1	2	4	8	16	32	64	128
(2,2)	30	40	0	0	0	1	0	1	0	0
(2,3)	30	192	0	0	0	0	0	0	1	1
(2,4)	20	248	0	0	0	1	1	1	1	1
(2,5)	30	24	0	0	0	1	1	0	0	0
(2,6)	30	160	0	0	0	0	0	1	0	1
(3,2)	30	6	0	1	1	0	0	0	0	0
(3,3)	20	207	1	1	1	1	0	0	1	1
(3,4)	20	143	1	1	1	1	0	0	0	1
(3,5)	20	159	1	1	1	1	1	0	0	1
(3,6)	30	3	1	1	0	0	0	0	0	0
(4,2)	20	6	0	1	1	0	0	0	0	0
(4,3)	10	191	1	1	1	1	1	1	0	1
(4,4)	10	255	1	1	1	1	1	1	1	1
(4,5)	10	239	1	1	1	1	0	1	1	1
(4,6)	20	3	1	1	0	0	0	0	0	0

	Gris	LBP	1	2	4	8	16	32	64	128
(3,2)	5	62	0	1	1	1	1	1	0	0
(3,3)	10	63	1	1	1	1	1	1	0	0
(3,4)	10	255	1	1	1	1	1	1	1	1
(3,5)	10	231	1	1	1	0	0	1	1	1
(3,6)	5	195	1	1	0	0	0	0	1	1
(4,2)	5	62	0	1	1	1	1	1	0	0
(4,3)	10	62	0	1	1	1	1	1	0	0
(4,4)	10	255	1	1	1	1	1	1	1	1
(4,5)	10	227	1	1	0	0	0	1	1	1
(4,6)	5	227	1	1	0	0	0	1	1	1

Figura 7.5. Cálculo detallado de LBP para todos los píxeles de la región.

En la figura 7.6 se puede ver como quedaría el mapa de valores LBP.

		1	2	3	4	5	6	7
1		0	0	0	0	0	0	0
2		0	40	192	248	24	160	0
3		0	6	207	143	159	3	0
4		0	6	191	255	239	3	0
5		0	62	63	255	231	195	0
6		0	62	62	255	227	227	0
7		0	0	0	0	0	0	0

Figura 7.6. Representación visual de los valores LBP con 8 vecinos y radio 1.

A continuación, tenemos que calcular la frecuencia de los valores para obtener el histograma. En el primer caso, se pide utilizar 13 bins, por lo que se divide el espacio de posibles valores [0,255] en 13 regiones iguales. En este caso no es posible por lo que la última región contará con tan solo 15 posibles valores, mientras que el resto abarcará 20. En la figura 7.7 se puede ver la representación gráfica del histograma.

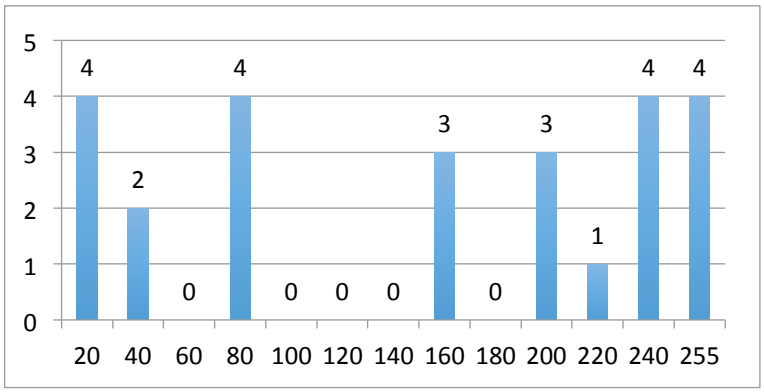


Figura 7.7. Histograma utilizando 13 bins del LBP de la región de la imagen de la figura 7.2

Realizando el mismo proceso se obtienen los histogramas para los dos casos restantes. En la figura 7.8 para 24 bins y en la figura 7.9 para 10 bins.

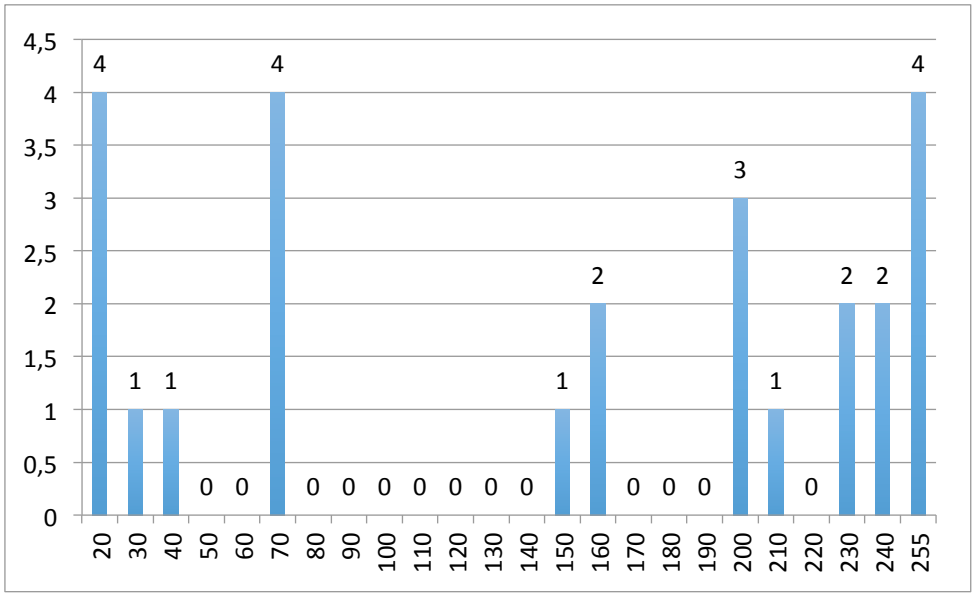


Figura 7.8. Frecuencias e histograma utilizando 24 bins del LBP de la región de la imagen.

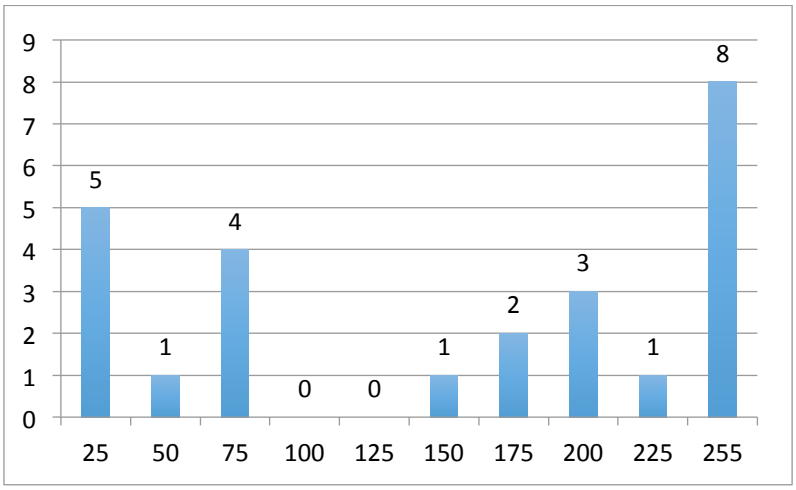


Figura 7.9. Histograma utilizando 10 bins del LBP de la región de la imagen.

7.3.2. Variaciones

7.3.2.1 Invariancia a la rotación

La primera mejora que se propuso, de gran importancia para el campo de la descripción de textura, fue la invariancia a la rotación. Atendiendo a la definición de Local Binary Pattern, es fácil darse cuenta que el método no es invariante a la rotación. En la figura 7.10 podemos ver un ejemplo de un patrón una vez aplicado el umbral a la diferencia de valor de gris, y el mismo rotado 90 grados, viendo que el resultado final es diferente

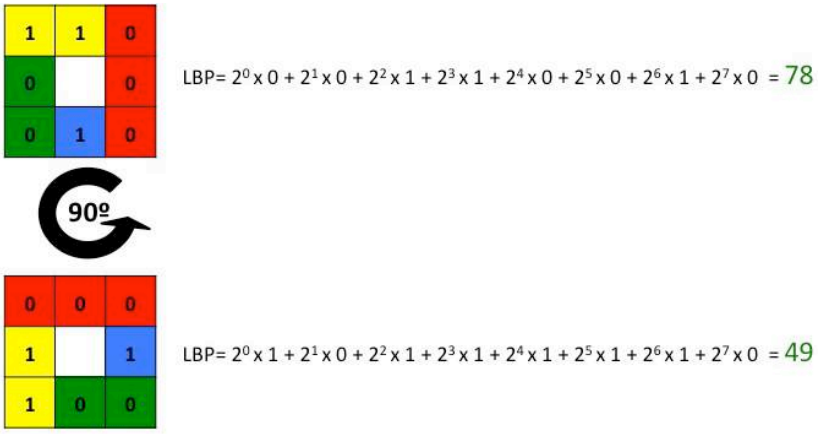


Figura 7.10 Ejemplo de dos patrones iguales pero con diferente orientación.

Para corregir este falta de robustez, se propuso un método basado en asignar a cada píxel el menor valor resultante de LBP de entre todas las posibles rotaciones del patrón. En la figura 7.11 se pueden ver todas las posibles combinaciones de un patrón concreto y el valor asignado, correspondiente a la rotación 7.

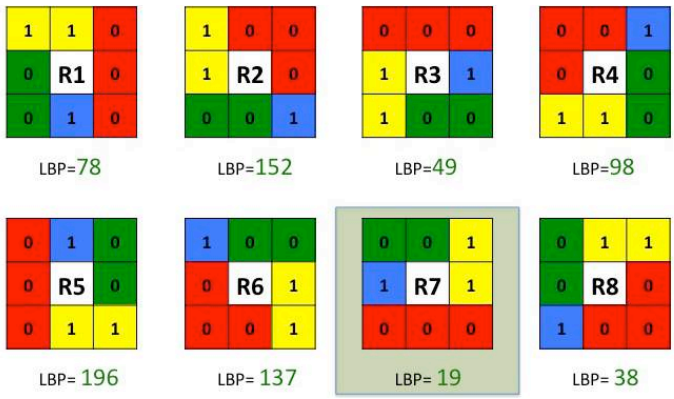


Figura 7.11. Ejemplo de todas las posibles rotaciones de un vector y su valor de LBP. En el método invariante a la rotación, se selecciona el valor mínimo para representar al píxel central.

Utilizando la ecuación 7.2 se puede obtener una descripción invariante a la rotación aplicando la idea que acabamos de mencionar:

$$LBP_{P,R}^{ri} = \min\left\{ROR(LBP_{P,R},i) \mid i = 0,\ldots,P-1\right\} \tag{7.2}$$

donde ROR(x,i) es una desplazamiento hacia la derecha de i píxeles para el patrón x.

7.3.2.2 Uniformidad

En 2002 se realizó una nueva propuesta por parte de varios de los autores del método original (Ojala y col. 2002), al darse cuenta de que la mayoría de la información relevante de la textura era descrita mediante un patrón uniforme. Se puede definir patrón uniforme como aquel patrón que tiene como mucho dos saltos entre 0 y 1 o entre 1 y 0 en su cadena, asumiéndose que es un patrón circular,

por lo que es necesario comprobar también si hay transición entre el último elemento y el primero. En la figura 7.12 podemos observar los distintos patrones uniformes que existen y algunos ejemplos de aquellos que no lo son.

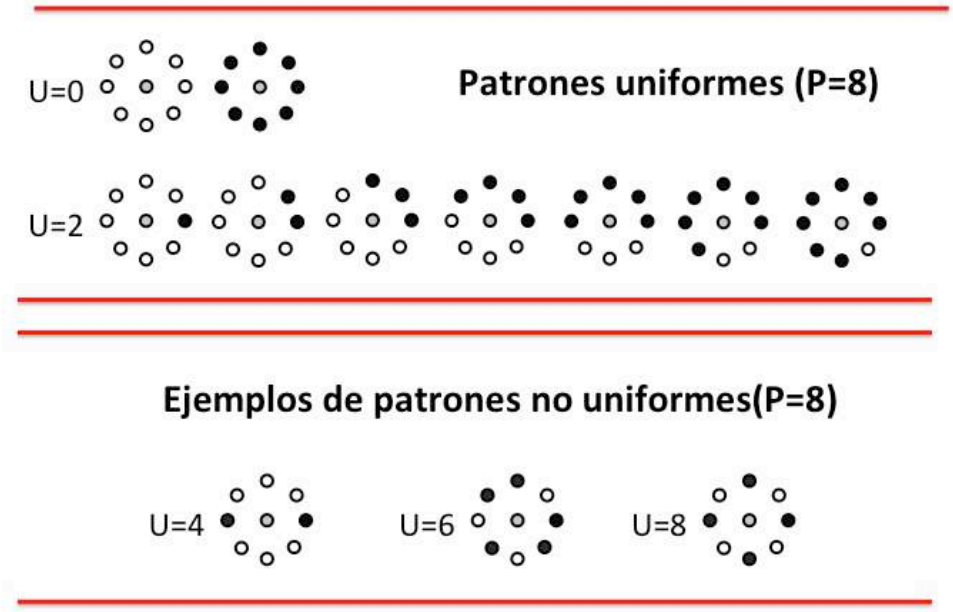


Figura 7.12. Patrones uniformes y no uniformes en LBP.

En la figura 7.13 podemos ver las primitivas que representan los valores de algunos de los patrones uniformes.

Teniendo en cuenta únicamente estos patrones, cualquier píxel de la imagen se puede representar mediante P+2 valores diferentes, asignando desde 0 hasta P el valor que corresponde con el número de píxeles cuyo valor es 1, y el valor P+1 a cualquier patrón no uniforme.

Figura 7.13. Patrones uniformes y no uniformes en LBP.

En la ecuación 7.3 se puede ver de manera matemática el cálculo del LBP uniforme.

$$LBP_{P,R}^{riu2} = \begin{cases} \sum_{p=0}^{P-1} s(g_p - g_c) & si\, U(LBP_{P,R}) \le 2 \\ P+1 & otherwise \end{cases} \tag{7.3}$$

Donde U(LBP) se define como indica la ecuación 7.4

$$U(LBP) = \left| s(g_{P-1} - g_c) - s(g_0 - g_c) \right| + \sum_{p=1}^{P-1} s(g_p - g_c) - s(g_{p-1} - g_c) \tag{7.4}$$

Si el valor de U para un determinado patrón del LBP es menor o igual que 2, se designa como patrón uniforme. Los patrones que tienen un valor de U mayor que 2 son catalogados como no uniformes. El patrón L1=00001000 es uniforme porque U(L1)=2, mientras que el patrón L2=01100100 es no uniforme porque U=4.

Ejemplo 7.2. *Obtener el histograma LBP Uniforme para el fragmento de la imagen que se muestra en la figura 7.14 con radio 1 y 8 vecinos (sin considerar interpolaciones).*

	1	2	3	4	5	6	7
1	0	0	0	0	0	0	0
2	0	30	30	20	30	30	0
3	0	30	20	20	20	30	0
4	0	20	10	10	10	20	0
5	0	5	10	10	10	5	0
6	0	5	10	10	10	5	0
7	0	5	10	10	10	5	0

Figura 7.14. Fragmento de imagen en escala de grises.

Solución: El primer paso es determinar la región sobre la que se va a calcular LBP Uniforme. Al ser un vecindario de radio 1, se debe eliminar del cálculo 1 píxel de cada borde de la imagen. El resultado se puede ver en la figura 7.15. Este paso es igual que el realizado a la hora de calcular el método simple de LBP.

	1	2	3	4	5	6	7
1							
2		30	30	20	30	30	
3		30	20	20	20	30	
4		20	10	10	10	20	
5		5	10	10	10	5	
6		5	10	10	10	5	
7							

Figura 7.15. Área de la imagen sobre la que se debe calcular LBP Uniforme

A continuación, en la figura 7.16, se muestran ejemplos de la extracción del valor de LBP Uniforme para 4 píxeles de la imagen.

Píxel (1,1)												1	2	4	8	16	32	64	128	
	0	0	0		0	0	0													
	0	30	30		0	30	1													
	0	30	20		0	1	0		LBP8,1 (1,1) =		9	0	0	0	1	0	1	0	0	NO UNIFORME
Píxel (4,4)																				
	20	20	20		1	1	1													
	10	10	10		1	10	1													
	10	10	10		1	1	1		LBP8,1 (4,4) =		8	1	1	1	1	1	1	1	1	UNIFORME
Píxel (5,6)																				
	10	5	0		1	1	0													
	10	5	0		1	5	0													
	10	5	0		1	1	0		LBP8,1 (5,6) =		5	1	1	0	0	0	1	1	1	ESQUINA
Píxel (4,6)																				
	20	30	0		1	1	0													
	10	20	0		0	20	0													
	10	5	0		0	0	0		LBP8,1 (4,6) =		2	1	1	0	0	0	0	0	0	FIN DE LÍNEA

Figura 7.16. Cálculo detallado de LBP Uniforme para 4 píxeles de la imagen.

Aplicando la fórmula de LBP uniforme para todos los píxeles obtendríamos los valores que se muestran en la figura 7.17:

	Gris	LBP	1	2	4	8	16	32	64	128
(2,2)	30	9	0	0	0	1	0	1	0	0
(2,3)	30	2	0	0	0	0	0	0	1	1
(2,4)	20	5	0	0	0	1	1	1	1	1
(2,5)	30	2	0	0	0	1	1	0	0	0
(2,6)	30	9	0	0	0	0	0	1	0	1
(3,2)	30	2	0	1	1	0	0	0	0	0
(3,3)	20	6	1	1	1	1	0	0	1	1
(3,4)	20	5	1	1	1	1	0	0	0	1
(3,5)	20	6	1	1	1	1	1	0	0	1
(3,6)	30	2	1	1	0	0	0	0	0	0
(4,2)	20	2	0	1	1	0	0	0	0	0
(4,3)	10	7	1	1	1	1	1	1	0	1
(4,4)	10	8	1	1	1	1	1	1	1	1
(4,5)	10	7	1	1	1	1	0	1	1	1
(4,6)	20	2	1	1	0	0	0	0	0	0

	Gris	LBP	1	2	4	8	16	32	64	128
(3,2)	5	5	0	1	1	1	1	1	0	0
(3,3)	10	6	1	1	1	1	1	1	0	0
(3,4)	10	8	1	1	1	1	1	1	1	1
(3,5)	10	6	1	1	1	0	0	1	1	1
(3,6)	5	4	1	1	0	0	0	0	1	1
(4,2)	5	5	0	1	1	1	1	1	0	0
(4,3)	10	5	0	1	1	1	1	1	0	0
(4,4)	10	8	1	1	1	1	1	1	1	1
(4,5)	10	5	1	1	0	0	0	1	1	1
(4,6)	5	5	1	1	0	0	0	1	1	1

Figura 7.17. Cálculo detallado de LBP Uniforme para todos los píxeles de la región.

En la figura 7.18 se puede ver cómo quedaría el mapa de valores LBP Uniforme distribuidos en la propia región.

	1	2	3	4	5	6	7
1	0	0	0	0	0	0	0
2	0	9	2	5	2	9	0
3	0	2	6	5	6	2	0
4	0	2	7	8	7	2	0
5	0	5	6	8	6	4	0
6	0	5	5	8	5	5	0
7	0	0	0	0	0	0	0

Figura 7.18. Representación visual de los valores LBP Uniforme con 8 vecinos y radio 1.

A continuación, tenemos que calcular la frecuencia de los valores para obtener el histograma. En este caso como tan solo hay 10 posibles valores, el histograma lo generaremos con 10 bins, uno para cada valor. En la figura 7.19 se puede ver las distintas frecuencias para cada valor y el histograma representado gráficamente.

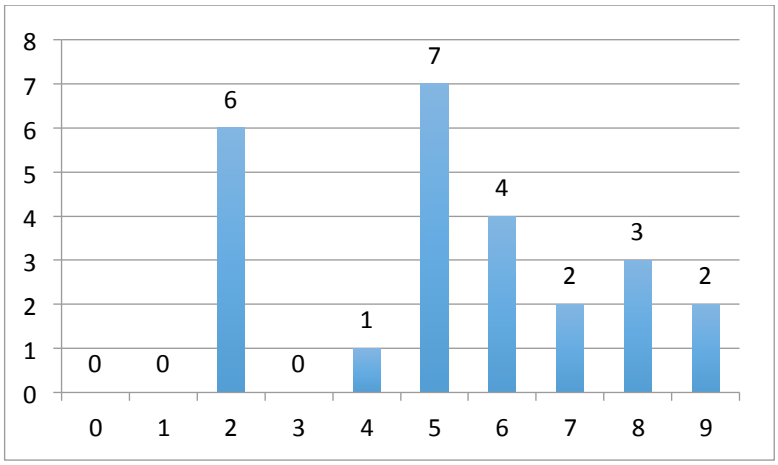


Figura 7.19. Histograma utilizando 10 bins del LBP Uniforme de la región de la imagen de la figura 7.2

7.4. Métodos derivados de LBP

7.4.1. ALBP

En Guo y col. (2010) se propuso un nuevo descriptor de textura basado en LBP, centrado en incorporar información acerca de la orientación de las texturas. El método trata de minimizar las variaciones de la media y la desviación estándar de las diferencias direccionales (diferencias del valor de gris del píxel central y cada uno de los vecinos dependiendo de su posición respecto al centro). Para ello, se propuso añadir un parámetro extra w en la ecuación $|g_c - w_p * g_p|$. La función objetivo es definida según la ecuación (7.5).

$$w_p = \arg_w \min \left\{ \sum_{i=1}^{N} \sum_{j=1}^{M} \left| g_c(i,j) - w \cdot g_p(i,j) \right|^2 \right\} \tag{7.5}$$

donde w_p es el elemento utilizado para minimizar la diferencia de la dirección p y, N y M son el número de filas y de columnas de la imagen. Una vez dicho esto, la ecuación final del Adaptive Local binary Pattern (ALBP) se define como:

$$ALBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - w_p \cdot g_c)2^p, s(x) = \begin{cases} 1 \; if \; x \geq 0 \\ 0 \; if \; x < 0 \end{cases} \tag{7.6}$$

7.4.2. LBPV

Local Binary Pattern Variance fue propuesto por Guo y su grupo de investigación como una combinación de LBP y un método de distribución de contraste (Guo y col. 2010). Esté método utiliza la varianza de la imagen como peso adaptativo para ajustar la contribución de cada valor de LBP en el cálculo del histograma. LBPV se calcula de la siguiente manera:

$$LBPV_{P,R} = \sum_{i=1}^{N} \sum_{j=1}^{M} w(LBP_{P,R}(i,j),k), k \in [0,K] \tag{7.7}$$

Donde *k* es cada uno de los valores del histograma, K es el valor máximo de LBP y w es definido como:

$$w(LBP_{P,R}(i,j),k) = \begin{cases} VAR_{P,R}(i,j), & LBP_{P,R}(i,j) = k \\ 0 & otherwise \end{cases} \tag{7.8}$$

Cuya varianza del vecindario VARR,P se define como:

$$VAR_{P,R} = \frac{1}{P} \sum_{p=0}^{P-1} (g_p - \mu)^2 \tag{7.9}$$

donde μ es la media del vecindario.

7.4.3. CLBP

El mismo grupo de investigación que desarrolló LBPV y ALBP, presentó otro método llamado CLBP (Compound Local Binary Pattern) que trata de generalizar y completar la información que aporta el LBP clásico (Guo y col. 2010). En este método, una región local de la imagen es representada por su píxel central y una transformada local de diferencias signo-magnitud llamada LDSMT por sus iniciales en inglés. LDSMT descompone estructura local de la imagen en dos componentes complementarios: la diferencia de signos, que se corresponde con el clásico LBP y lo llaman CLBP_S y la diferencia de magnitudes CLBP_M que se define utilizando la ecuación:

$$CLBP\_M_{P,R} = \sum_{p=0}^{P-1} t(g_p - g_c, c)2^p, t(x,c) = \begin{cases} 1 \; si \; x \geq c \\ 0 \; si \; x < c \end{cases} \tag{7.10}$$

siendo c un umbral determinado adaptativamente. Normalmente se utiliza el valor medio de todas las diferencias de magnitudes de la imagen.

7.5. Aplicaciones de LBP

LBP se ha convertido en un método muy utilizado en visión por computador debido a su alto poder discriminante, la tolerancia frente a cambios de iluminación y su simplicidad computacional.
Se ha utilizado LBP para realizar segmentación no supervisada basada en características de textura de la imagen, Ojala y Pietikäinen (1999). Para ello primero se divide la imagen de manera jerárquica en un número de regiones escogido por el usuario. A continuación se realiza una fusión de aquellas texturas contiguas que tengan un descriptor similar dejando para el final refinar las regiones, eliminando las aristas producidas en el paso de la división jerárquica, obteniendo una segmentación más ajustada y precisa.

En el ámbito del reconocimiento de rostros, LBP ha sido utilizado obteniendo buenos resultados. Yang y Chen (2013), realizaron un estudio evaluando diferentes técnicas basadas en LBP para el reconocimiento de rostros demostrando, entre otras cosas, que al dividir el rostro en subregiones para posteriormente describirlas usando LBP incrementaba notablemente la precisión de los sistemas de clasificación frente a la descripción del rostro por completo.

También se ha utilizado LBP para la detección de rostros en imágenes. En este campo se pueden utilizar diferentes estrategias para detectar rostros, por ejemplo en el trabajo de Jiayu y Chengjun (2013), se centran en detectar los ojos de las personas para estimar la posición de la cara teniendo en cuenta que el ojo es una región de la imagen con una alta variabilidad con respecto a las zonas adyacentes del mismo (resto de la cara).

Aunque el campo de la detección de objetos está muy focalizado en técnicas de características locales invariantes tipo SIFT o SURF, es posible encontrar trabajos que también han utilizado estrategias basadas en LBP para la detección de objetos. En el trabajo de Dornaika y col. (2014), se propone una técnica basada en la construcción de un grafo adaptativo que utiliza está técnica (LBP) para la descripción de las regiones. En este trabajo, se centran en la idea de que el descriptor de una imagen está formado por una combinación del resto de descriptores existentes en la base de datos a los que se les asigna unos pesos concretos. De esta manera, si una imagen contiene un objeto concreto, el peso asociado a ese objeto existente en la base de datos será alto, permitiendo así su detección.

7.6. Bibliografía

Bo, Y.; Songcan C. (2013) A comparative study on local binary pattern (LBP) based face recognition: LBP histogram versus LBP image, *Neurocomputing*, 120, 365-379.

Cross, G.R.; Jain, A.K. (1983) Markov Random Field Texture Models. *TransPAMI, 5,* 25-39.

Daugman, J. (1985) Uncertainty relation for resolution in space, spatial frequency and orientation optimised by two-dimensional visual cortical filters. *Journal of the Optical Society of America, 2,*1160-1169.

Dornaika, F.; Bosaghzadeh, A.; Salmane, H.; Ruichek, Y. (2014) Graph-based semi-supervised learning with Local Binary Patterns for holistic object categorization, *Expert Systems with Applications*, *41*, 7744-7753.

Jiayu, G.; Chengjun, L.; Feature local binary patterns with application to eye detection, *Neurocomputing*, *113*, 138-152.

Guo, Z.; Zhang, L.; Zhang, D.; Zhang, S. (2010) Rotation invariant texture classification using adaptive LBP with directional statistical features, *Image Processing (ICIP)*, 285 –288.

Guo, Z.; Zhang, L.; Zhang, D. (2010) Rotation invariant texture classification using lbp variance (LBPV) with global matching, *Pattern Recognition, 43 (3)*, 706–719.

Guo, Z.; Zhang, L.; Zhang, D. (2010) A completed modeling of local binary pattern operator for texture classification, *Image Processing, IEEE Transactions on, 19*(6), 1657–1663.

Haralick, R. M.; Shanmugam, K.; Dinstein, I. (1973) Textural features for image classification. *IEEE Transactions on Systems, Man and Cybernetics, 3(6)*, 610–621.

Haralick, R.M. (1979) Statistical and structural approaches to texture. *Proc IEEE, 67(5),* 786-804.

Laws, K.I. (1980) Textured image segmentation. *Ph.D. dissertation*, Univ. Southern California, Los Angeles.

Mallat, S. (1989) Multifrequency cannel decomposition of images and wavelet models. *IEEE Trans. Acoustic, Speech and Signal Processing, 37(12),* 2091-2110.

Ojala, T.; Pietikäinen, M.; Harwood, D. (1994) Performance evaluation of texture measures with classification based on kullback discrimination of distributions. *Proceedings of the 12th IAPR International Conference on Pattern Recognition (ICPR 1994).*

Ojala, T.; Pietikäinen M.; Mäenpää T. (2002) Multiresolution gray-scale and rotation invariant texture classification with Local Binary Patterns, *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 24* (7).

Rao, A.; Lohse, G.L. (1993). Identifying High level features of texture perception. *CVGIP: Graphical Models and Image Processing, 55(3),*218-233.

Rosenfeld, A.; Weszka, J. (1980) Picture Recognition. *Digital Pattern Recognition.*135-166.

Tsantis, S.; Dimitropoulos, N.; Cavouras, D.; Nikiforidis, G. (2009) Morphological and wavelet features towards sonographic thyroid nodules evaluation. *Comput Med Imaging Graph*, *33*(2), 91-99.

Turceryan, M.; Jain, A.K. (1993) *Texture Analysis*, chapter 2, World scientific publishing Co.,235-276.

Zhou, D. (2006) Texture Analysis and Synthesis using a Generic Markov-Gibbs Image Model. *Master theses.* University of Auckland.

CAPÍTULO 8

SIFT (SCALE INVARIANT FEATURE TRANSFORM)

Enrique ALEGRE[1], Laura FERNÁNEZ-ROBLES[2]

[1] Dpto. de Ingeniería Eléctrica y de Sistemas y Automática, Universidad de León, León, España.
[2] Dpto de Ingeniería Mecánica, Informática y Aeroespacial, Universidad de León, León, España.

SIFT es un método que permite detectar puntos característicos en una imagen y luego describirlos mediante un histograma orientado de gradientes. Y además, lo hace de forma que la localización y la descripción presenta una gran invarianza a la orientación, la posición y la escala. Cada punto característico queda, por lo tanto, definido mediante su vector de características de 128 elementos, y se obtiene la información de su posición en coordenadas de la imagen, la escala a la que se encontró y la orientación dominante de la región alrededor de dicho punto.

En este capítulo se explican los pasos necesarios para obtener descriptores SIFT en una imagen. Se presenta un ejercicio sencillo que sirve para ilustrar numéricamente cómo se obtiene el descriptor a partir de la región que rodea un punto característico. También se comentan las posibilidades de SIFT para realizar reconocimiento de objetos presentes en una imagen. Y, finalmente, se habla brevemente de algunas extensiones del método así como de otros descriptores de imagen relacionados que han surgido posteriormente.

8.1. Introducción

La publicación del primer artículo donde Lowe presentó SIFT (Lowe, 1999) supuso un antes y un después en el campo de la visión por computador. Si bien las publicaciones relacionadas con la detección de características locales se remonta a cuarenta y cinco años atrás, cuando Attneave (1954) observó que la información de la forma se concentra en puntos dominantes que presentan alta

curvatura, Lowe destiló el conocimiento generado durante esos años para presentar un método robusto y eficiente. A partir de ese momento, gran parte de la actividad de los mejores investigadores del campo de la visión se centró en el desarrollo, la mejora o la aplicación de SIFT y en la propuesta de métodos alternativos. Dos de los autores que más han contribuido a este campo, después de Lowe, han sido Kristian Mikolajczyk y Tinna Tuytelaars que en su trabajo, "Local Invariant Feature Detectors: A Survey", (Tuytelaars y Mikolajczyk, 2008), presentan, a lo largo de 104 páginas, una revisión de los distintos métodos y aportaciones que fueron necesarios y en los que se basa el método de SIFT (Scale Invariant Feature Transform).

Si bien, no es fácil entender SIFT sin explicar qué es una característica local, qué es y cómo funciona un detector de esquinas (o corners) y otros conceptos como la detección de blobs, la detección de regiones y la forma de implementar todo ello de manera eficiente, en este capítulo vamos a intentar hacerlo. Nuestro objetivo es presentar de forma completa y concisa las distintas etapas del método de manera que facilite su compresión o sirva, para aquellos que no lo conocen, como un primer acercamiento a esta técnica.

8.2. SIFT (Scale Invariant Feature Transform)

Para obtener un conjunto de descriptores SIFT de una imagen es necesario por un lado obtener los puntos característicos y después, para cada punto de interés, calcular su vector descriptor a partir de la información de los píxeles que lo rodean. SIFT fue propuesto para imágenes en escala de grises, por lo que el vector de características de 128 elementos que define cada píxel, contiene información sobre cómo se distribuyen los niveles de intensidad alrededor de cada punto de interés previamente obtenido.

Por lo tanto, el algoritmo consta de **dos partes** claramente diferenciadas:

a) *Obtención* de los *puntos característicos*

b) *Descripción* de la región alrededor de cada punto de interés

La **obtención de los puntos característicos** o puntos de interés, a los que habitualmente se llama en inglés *keypoints*, consiste en detectar aquellas regiones de la imagen en las que se producen diferencias de gradiente significativas a ambos lados de dicho punto. Si el método solamente hiciera eso, se podría pensar que esta etapa se podría realizar utilizando un detector de esquinas, como por ejemplo el detector de esquinas de Harris (Harris y Stephens, 1988). La propuesta de Lowe (Lowe, 1999) va un poco más allá y propone no detectar únicamente esquinas, sino **blobs**, y hacerlo de manera que esa detección sea consistente cuando el punto característico aparezca a diferentes escalas.

La diferencia entre una esquina, *corner,* y un *blob,* puede resultar, a priori, un poco difusa. Si nos fijamos en su definición, una esquina es un punto, o al menos un área pequeña en una imagen, donde confluyen –al menos- dos bordes. Se define un **blob** como una región de una imagen que se caracteriza porque algunas de sus propiedades se mantienen aproximadamente constantes. Dentro del contexto de la detección de puntos característicos, *keypoints*, la propiedad que se suele considerar constante en un blob es la similitud en su nivel de gris, típicamente medida a partir de la variación del gradiente en esa

región a lo largo de diferentes direcciones. Según esto, claramente un *blob* es una región mayor que un punto y una esquina, *corner*, es "un punto". La confusión se puede producir cuando, en el método de SIFT y en otros métodos similares, para el cálculo de características locales invariantes, se habla de "puntos de interés", *kepoints*, y cuando se estudia y compara el detector de esquinas de Harris con el Laplaciano de la Gaussiana (LoG), o la Diferencia de Gausianas (DoG) como posibles métodos para obtener "los puntos característicos". En resumen, buscamos puntos que pueden ser pequeñas regiones con un nivel de intensidad estable y alrededor de las cuales se produce una variación significativa del gradiente, en más de una dirección.

La detección de esos puntos característicos a diferentes escalas requiere crear un espacio-escala, detectar en él puntos de interés, y eliminar aquellos que se consideren poco estables, como explicaremos en las siguientes secciones.

Para obtener el **descriptor de cada punto característico**, Lowe (Lowe 1999, 2004) propuso calcular un *histograma de direcciones del gradiente local* alrededor del punto de interés. El descriptor que se obtiene se convierte en *invariante a la escala* al normalizar el tamaño del vecindario local al punto de interés en función de ella. Además, es *invariante a la rotación* porque se determina la orientación dominante de los vectores del gradiente en el vecindario del punto característico, y se utiliza dicha información para orientar la rejilla que se usa para calcular el histograma.

Lowe nombró al método SIFT: *Scale Invariant Feature Transform* porque, en base a lo anterior, transforma datos de la imagen en características locales que se obtienen y expresan en coordenadas de la imagen invariantes a la escala.

8.2.1. Etapas del algoritmo

Si bien Lowe presenta por primera vez SIFT en el CVPR'99 (Computer Vision and Pattern Recognition Conference), (Lowe, 1999), es el artículo publicado cinco años después, en la *International Journal of Computer Vision* (Lowe, 2004) donde el método es explicado con un mayor nivel de detalle. En esta última publicación se dice que las etapas para el cálculo del método son las siguientes:

1. **Detección de extremos** en el espacio-escala

   Se buscan puntos de interés en toda la imagen y en todas las escalas consideradas utilizando una diferencia de Gaussianas.

2. **Localización precisa** de puntos característicos

   Para cada uno de los puntos de interés anteriores se ajusta un modelo que permite determinar su localización y escala. Además, se seleccionan los puntos característicos, *keypoints,* eliminando los que están próximos a los bordes o tienen bajo contraste.

3. Asignación de la **orientación**

   A cada punto característico se le asigna una o varias *orientaciones* en función de las *direcciones del gradiente local*. Esta orientación conjuntamente con la ubicación y la escala calculadas anteriormente permiten que el descriptor sea invariante a estas tres situaciones.

4. **Descripción** del punto característico

   Alrededor de cada punto característico se miden los *gradientes locales de la imagen* y se utiliza su histograma para obtener una representación de esa región que es robusta a cambios significativos en la iluminación y a pequeñas distorsiones en la forma.

8.2.2. El espacio escala

La distancia a la que están ubicados los objetos afecta a su percepción. Un objeto que se encuentra lejos presenta un tamaño menor que el mismo objeto cuando está situado más próximo. Este fenómeno produce que el mismo objeto pueda aparecer en una imagen con diferentes dimensiones en función de la distancia al observador. Para obtener una descripción de los objetos que sea robusta ante estas variaciones en tamaño se utiliza el "espacio-escala".

Koenderink (Koenderink, 1984) y Lindeberg (Lindeberg, 1994) mostraron, tras realizar diversas asunciones razonables, que el único kernel posible para el espacio-escala es la función gaussiana. Por lo tanto, el **espacio escala de una imagen**, $I(x,y)$, *consiste en una familia de imágenes derivadas*, $L(x,y,\sigma)$, que se obtienen por la convolución de una gaussiana de desviación típica σ y escala variable, $G(x,y,\sigma)$, con la imagen de entrada $I(x,y)$:

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y) \quad , \tag{8.1}$$

donde * denota la operación de convolución en las coordenadas *x* e *y,* y donde

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{8.2}$$

Un ejemplo de dicho espacio escala puede verse en la figura 8.1, donde diferentes valores de la desviación típica de la gaussiana produce sucesivas imágenes con menor nivel de detalle.

Para detectar de forma eficiente ubicaciones estables de puntos característicos, Lowe (Lowe 1999) propuso utilizar los valores extremos del espacio escala de la diferencia de Gaussianas de la imagen, $D(x,y,\sigma)$, que se puede calcular como la diferencia de dos escalas consecutivas separadas por un factor multiplicativo constante, *k*:

$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) \tag{8.3}$$

$$= L(x,y,k\sigma) - L(x,y,\sigma)$$

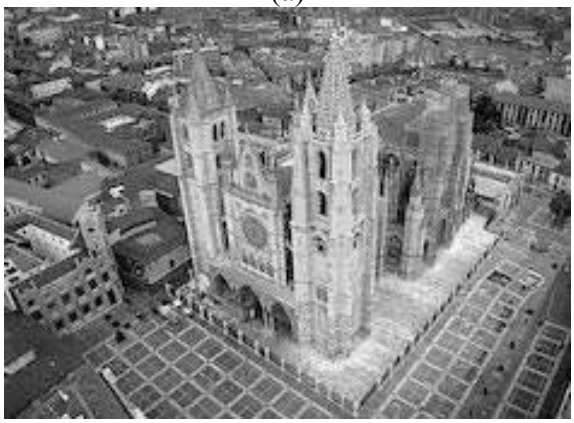
Las principales razones que llevaron a Lowe a escoger esta función, fueron:

- Es *particularmente eficiente de calcular*, ya que las imágenes suavizadas, $L(x,y,\sigma)$, tienen que obtenerse obligatoriamente para la descripción de características en el espacio escala y, por lo tanto, $D(x,y,\sigma)$, puede obtenerse simplemente restando las imágenes anteriores.

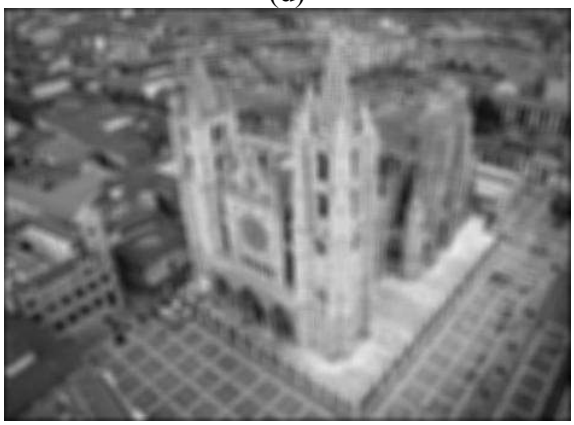
- Es una *buena aproximación al Laplaciano de la Gausiana normalizado a la escala*, $\sigma^2\nabla^2 G$, como estudió Lindeberg (Lindeberg 1994), ya que la normalización del Laplaciano utilizando el factor σ^2 es necesaria para que exista una invarianza real a la escala.

- Además, según demostró Mikolajczyk (Mikolajczyk y Schmid, 2002), los máximos y mínimos de $\sigma^2\nabla^2 G$,producen las características de imagen más estables, al compararlas con otras funciones de detección de esquinas como Harris, el Hessiano o el gradiente.

(a) (b)



(c) (d)



(e) (f)

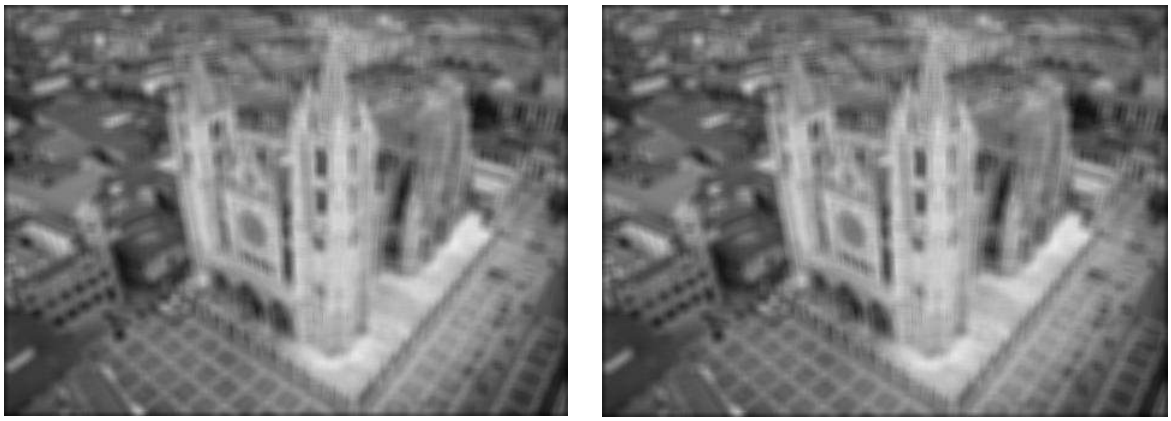


Figura 8.1. Catedral de León (*a*) Imagen original. (*b*) $\sigma = 1$ (*c*) $\sigma = 2$; *d*) $\sigma = 4$; (*e*) $\sigma = 8$; (*f*) $\sigma = 16$;

8.2.3. El espacio escala en SIFT

En SIFT el espacio escala se lleva un paso más allá porque además de construir el espacio escala de cada imagen, mediante la convolución con diferentes gaussianas, se crean varios espacios reduciendo sucesivamente el tamaño de la imagen original. A cada uno de esos espacios escala se llama una octava y se obtiene eliminando una de cada dos filas y columnas sobre la imagen de la octava anterior, reduciendo de esta forma sus dimensiones a la mitad. A este procedimiento se le conoce como la obtención de puntos característicos a partir de los extremos del espacio-escala generado a partir de diferencias de Gaussianas (DoG) dentro de una pirámide de diferencia de Gaussianas.

El concepto de pirámides de paso banda de diferencia de Gaussianas fue propuesto originalmente por Burt y Adelson (Burt y Adelson, 1983) y por Crowley y Stern (Crowley y Stern, 1984). Una pirámide Gaussiana se construye suavizando y reduciendo repetidamente la dimensión de la imagen original. La pirámide de diferencias de Gaussianas se calcula a partir de las diferencias entre los niveles adyacentes de la pirámide de Gaussiana.

Octavas y escalas

Como se ha explicado anteriormente, se parte de la imagen original y se generan imágenes progresivamente más suavizadas mediante la convolución con una gaussiana con desviación típica cada vez mayor. La relación entre las desviaciones típicas viene determinada por la constante *k*, de manera que si a la primera imagen se le aplica una σ, a la segunda se le aplica $k*\sigma$. Se dice que el conjunto de estas *n* imágenes con suavizado progresivo pertenecen a la misma ***octava***. Al dividir el tamaño de la imagen a la mitad, aparece la segunda octava, y así sucesivamente.

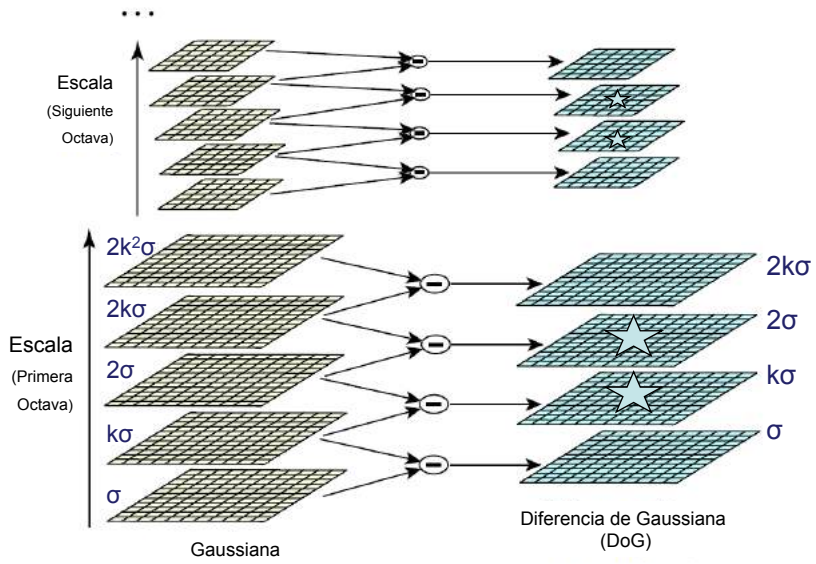


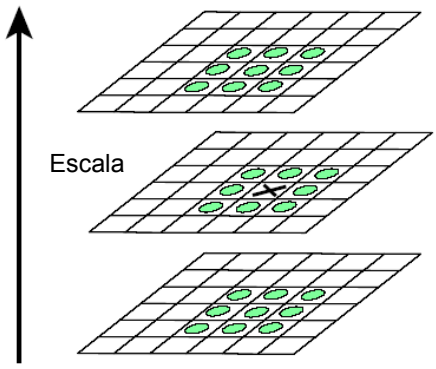
Figura 8.2. En la izquierda puede verse el espacio escala donde se representan dos octavas, la primera y la segunda con 5 escalas cada una de ellas. En la parte derecha (columna) del dibujo se ve cómo se obtiene el espacio escala de diferencias de gaussianas. Las estrellas indican las dos únicas imágenes, de cada octava, en las que puede buscarse máximos o mínimos, al ser las únicas que tienen una imagen vecina superior y otra inferior

El número de octavas y escalas dependen de la implementación que se desee realizar y suelen estar condicionadas por el tamaño y el detalle de la imagen original. Pero Lowe indicó en su trabajo (Lowe 2004) que 4 octavas y 5 escalas son lo ideal para el resto del proceso de detección de características invariantes.

Un detalle importante es que el método asume que la imagen original, $I(x,y)$ ha sido previamente suavizada con el objetivo de capturar de forma burda el efecto de que el tamaño de píxel en un CCD es finito. Esto se asume considerando la imagen de entrada como $I(x,y,\sigma_n)$, donde σ_n es un suavizado nominal, tomando $\sigma = \sigma_n = 0.5$, u otro valor similar, como el primer valor de la escala que se calcula.

8.2.4. Detección de extremos locales

Para detectar los máximos y mínimos locales de la función $D(x,y,\sigma)$ (ver ecuación 8.3), cada punto de la muestra se compara con sus ocho vecinos de la imagen en la que él se encuentra y también con los nueve vecinos de la imagen que se encuentra a una escala superior e inferior (ver figura 8.3). En dicha imagen pueden verse los 25 vecinos marcados con un círculo en verde. El punto se selecciona como un posible extremo solo si el valor de *D* es mayor que todos sus 25 vecinos o bien es menor que todos ellos. Esta comprobación tiene un coste relativamente bajo ya que la mayoría de los candidatos son descartados después de unas pocas comparaciones.



Espacio escala de DoG

Figura 8.3. Los círculos en verde indican aquellos píxeles que son vecinos de un punto y que son evaluados para determinar si el píxel marcado con "X" es un máximo o un mínimo local.

8.2.5. Localización precisa de los puntos de interés

Inicialmente (Lowe, 1999), los puntos en SIFT se ubicaban en la localización y la escala en la que se detectaban los extremos del espacio escala. Pero posteriormente, Brown (Brown y Lowe, 2002) desarrolló un método para ajustar una función cuadrática 3D a los puntos de una muestra local de manera que se pudiera determinar la localización interpolada del máximo. Sus experimentos mostraron que este ajuste proporciona una mejora sustancial en la búsqueda de correspondencias y la estabilidad ya que se eliminan puntos que tienen bajo contraste y por lo tanto son sensibles al ruido, o puntos que no están bien localizados al encontrarse a lo largo de un borde.

La propuesta de Brown utiliza la expansión de Taylor hasta los términos cuadráticos de la función espacio-escala, $D(x,y,\sigma)$, desplazada de forma que el origen esté en el punto de la muestra. La ubicación del extremo se determina calculando las derivadas de dicha función con respecto a x e igualando a cero. Brown sugirió aproximar el Hessiano y la derivada de D utilizando diferencias entre los vecinos del punto de la muestra. El valor de la función en el extremo, $D(\hat{x})$, es útil para rechazar extremos inestables con bajo contraste en la siguiente función:

$$D(\hat{x}) = D + \tfrac{1}{2}\frac{\partial D^T}{\partial x}\hat{x}\,, \tag{8.4}$$

En su artículo sobre SIFT (Lowe, 2004), Lowe propuso descartar todas los extremos de $|D(\hat{x})|$ con un valor menor de 0.03, asumiendo que los niveles de gris de la imagen están en el rango [0, 1].

Supresión de la respuesta de los puntos de interés a lo largo de los bordes

Los puntos característicos detectados a lo largo de los bordes son puntos menos estables ya que al estar su ubicación pobremente determinada pueden ser similares a pequeñas cantidades de ruido y, por ese motivo, SIFT propone su eliminación.

Al ser la diferencia de Gaussianas una función equivalente al Laplaciano de la Gausiana, es de entender que la respuesta que tiene a lo largo de los bordes sea muy fuerte. Un punto ubicado en un borde y que esté pobremente definido en la función de diferencia de Gaussianas presentará una gran curvatura a lo largo del borde pero una pequeña curvatura en la dirección principal. La curvatura en un punto, a una escala determinada se puede obtener mediante el Hessiano de 2x2, ***H****:*

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix}, \tag{8.5}$$

Donde D_{xx} es la derivada segunda en *x*, D_{xy} es la derivada segunda en *x* e *y*. Las derivadas se estiman a partir de las diferencias de los vecinos del punto de muestreo.

Aunque se conoce que los valores propios de ***H*** son proporcionales a las curvas principales, D. Lowe utilizó la aproximación propuesta por Harris (Harris y Stephens, 1988) para calcular esquinas. De esta manera, utilizando la traza y el determinante del Hessiano, se evita calcular explícitamente los valores propios y estudiando la relación entre ellos se eliminan los puntos de bajo contraste. Siguiendo la nomenclatura de Lowe (Lowe, 2004), llamemos α al valor propio de mayor magnitud y β al de menor. La suma y el producto de ambos puede obtenerse de la siguiente manera:

$$\begin{aligned} Tr(H) &= D_{xx} + D_{yy} = \alpha + \beta, \\ Det(H) &= D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta \end{aligned} \tag{8.6}$$

En los casos poco frecuentes en los que el determinante sea negativo el punto también se descarta como posible extremo ya que las curvaturas tienen diferentes signos.

Si se considera *r* la relación entre ambos valores propios, de manera que $\alpha = r\,\beta$, se puede expresar esta relación en función de la traza y el determinante del Hessiano, de la siguiente manera:

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}, \tag{8.7}$$

Cuando ambos valores propios son iguales el resultado de $(r+1)^2/r$ será mínima y se irá incrementando al hacerlo *r*. Por lo tanto, para comprobar que la relación entre las curvaturas principales es inferior a un umbral determinado, *r*, Lowe propone evaluar la siguiente expresión, que se calcula de forma muy eficiente, y conservar los puntos que la cumplan:

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}, \tag{8.8}$$

La propuesta en SIFT es utilizar un valor de $r=10$ para eliminar puntos clave con una relación entre las curvaturas principales con valores mayores de 10.

8.2.6. Asignación de la orientación

Con el objetivo de obtener invarianza a la rotación, a cada descriptor se le asigna una orientación en función de las propiedades locales de la imagen.

Para realizar este cálculo de forma que sea invariante a la escala se selecciona la imagen Gaussiana suavizada, *L,* con la escala más próxima a la del punto característico del que se va a obtener su descriptor. Para cada muestra de imagen a la escala especificada, $L(x,y)$, se calcula la magnitud del gradiente, $m(x,y)$, y su orientación, $\theta(x,y)$, utilizando diferencias entre los píxeles vecinos.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + ((L(x,y+1) - (L(x,y-1))^2}\,,$$
$$\theta(x,y) = \tan^{-1}\frac{L(x,y+1) - L(x,y-1))}{L(x+1,y) - L(x-1,y))} \tag{8.9}$$

A partir de las orientaciones obtenidas de la región alrededor del punto característico, se calcula un histograma de 36 bins conteniendo cada bin 10º hasta cubrir los 360º. Cada uno de los valores que se añaden al histograma se ponderan por la magnitud del gradiente utilizando una ventana circular Gaussiana con una σ que es 1,5 veces el tamaño de la escala a la que se localizó el punto característico.

Las direcciones dominantes de los gradientes locales se corresponderán con los picos del histograma. Se detecta el pico más alto y para él y todos aquellos con un valor superior al 80% de ese pico más alto, se creará un punto característico en la misma posición pero con las correspondientes orientaciones dominantes. Aunque los puntos con múltiples orientaciones son únicamente alrededor del 15%, éstos contribuyen mucho a que la búsqueda de correspondencias sea estable. El último paso consiste en ajustar una parábola a los 3 valores del histograma que están más próximos a cada pico elegido de manera que se interpola la posición del pico de una forma más precisa.

8.2.7. Descriptor

Mediante las operaciones explicadas en las secciones anteriores, a cada uno de los puntos característicos obtenidos, se les ha asignado una posición en la imagen, una escala y una orientación. El último paso del método consiste en calcular un descriptor que sea muy característico y al mismo tiempo lo más invariante posible a las variaciones que quedan, como son los cambios en iluminación y en el punto de vista 3D.

El descriptor propuesto por Lowe (Lowe, 1999) está inspirado en un trabajo realizado por Edelman (Edelman y col., 1997) sobre un modelo de visión biológica. Estos autores indicaron que determinadas neuronas del cortex visual primario responden al gradiente a una determinada orientación y frecuencia espacial, pero que dicho gradiente no es localizado de forma precisa en la retina sino que se permite su desplazamiento sobre un pequeño campo receptivo. Según Edelman, el funcionamiento de estas neuronas complejas permite realizar correspondencias y reconocer objetos 3D desde diferentes puntos de vista.

De forma resumida el cálculo del descriptor consta de los siguientes pasos:

1. Se obtiene la *magnitud y la orientación del gradiente* en una ventana de 16x16 muestras centrada en el punto característico.

2. Se calcula un *histograma de 8 bins con las orientaciones del gradiente para cada una de las 16 regiones* que se obtienen al dividir la ventana inicial en regiones menores de 4x4 muestras. Cada bin se corresponde con 45 grados, el primero de 0 a 44, el segundo de 45 a 89 y así sucesivamente. El histograma contendrá las orientaciones de los gradientes que aparecen en esa ventana.

3. Se *normaliza el vector de 128 elementos* que se obtiene al concatenar los histogramas de 8 valores para cada uno de las 16 divisiones que se han realizado de la región.

Se explican a continuación los detalles de cada uno de los pasos necesarios para calcular el descriptor.

1. Obtención de los gradientes en una ventana de 16x16 muestras alrededor del punto característico

Para obtener los valores de estos gradientes se realizan las siguientes operaciones.

1. Se *calculan a la escala apropiada.*

   Por lo que se selecciona la imagen con el nivel de suavizado Gaussiano que corresponde con la escala del punto característico.

2. Se obtienen con *invarianza a la orientación*.

   Para ello se rotan las coordenadas del descriptor y las orientaciones del gradiente en relación con la orientación del punto característico.

3. Se hace de forma *eficiente.*

   Todos los gradientes se calculan previamente para todos los niveles de la pirámide de Gaussianas. El cálculo es el mismo al realizado para asignar la orientación al descriptor, ya explicado previamente.

4. Se obtienen los valores *ponderados* por una gaussiana.

   Con el objetivo de evitar cambios bruscos en el descriptor ante pequeños cambios en la posición de la ventana y para reducir la influencia de los gradientes alejados del centro del descriptor, al verse éstos más afectados por errores de registro.

Para hacerlo se asigna un peso a la magnitud de cada punto mediante una función gaussiana con una σ igual a la mitad del ancho de la ventana del descriptor.

2. Histograma de orientaciones del gradiente para subregiones de 4x4 muestras

1. Se obtiene un *histograma de 8 direcciones para cada subregión*

   Para cada una de las 16 subregiones de 4x4 muestras se calcula un histograma de 8 intervalos, que son las orientaciones que resultan al subdividir el espacio en regiones de 45 grados.

   El valor que se asigna a cada uno de los 8 intervalos del histograma proviene de la suma de las magnitudes del gradiente de todas las muestras cuya orientación cae en la correspondiente región del histograma.

2. Se *distribuye el valor de cada gradiente en los intervalos adyacentes* del histograma, mediante una interpolación tri-lineal.

   Esto permite evitar los efectos de frontera de manera que el descriptor pueda cambiar de forma abrupta cuando una muestra pudiera pasar de un histograma a otro o de una orientación a otra.

   Cada una de las entradas a un intervalo del histograma se multiplica por un peso de $1 - d$ para cada dimensión, donde d es la distancia de la muestra al valor central del intervalo medida en unidades del espaciado de los intervalos del histograma.

Una de las ventajas de este método es que una muestra del gradiente puede desplazarse diversas posiciones, dentro de estas subregiones de 4x4 muestras, y aún contribuir de la misma forma al histograma obtenido. De esta forma el descriptor es robusto ante posibles desplazamientos locales de la posición de los puntos característicos.

El descriptor se forma concatenando los 16 histogramas de 8 direcciones de manera que el vector de características tendrá una longitud de 4x4x8 = 128 elementos, para cada punto característico.

3. Normalización del vector de características

Como los valores de los gradientes se calculan a partir de diferencias entre niveles de gris de los píxeles, el descriptor obtenido es invariante a cambios afines en la iluminación. Sin embargo, pueden darse cambios no lineales en la iluminación debidos a la saturación de la cámara o producidos como consecuencia de los cambios en la reflexión sobre superficies 3D que presenten diferentes orientaciones.

Para evitar los problemas anteriores y también que medidas locales de alto contraste afecten en exceso al descriptor, se realiza la siguiente normalización en dos pasos:

1. Se *normaliza* el vector *a longitud unidad*.

   Dado el vector de 128 elementos obtenido para un punto característico, se obtiene la longitud de dicho vector, calculado como su norma, y luego se divide cada uno de los elementos de dicho vector por dicha magnitud. De esta forma el vector pasará a tener longitud unidad.

$$\hat{u} = \frac{u}{||u||}, \tag{8.10}$$

2. Umbralización de cada valor a 0.2 y nueva normalización

   Para reducir la influencia de grandes gradientes, cada uno de los elementos del vector de características se umbraliza de manera que ninguno tenga un valor superior a 0.2. Después, el vector es normalizado de nuevo a longitud unidad.

Ejemplo 8.1. *Dada la siguiente región de una imagen, calcular el descriptor SIFT para los 16x16 valores del interior de la línea de trazo grueso de la figura 8.3.*

En este ejercicio se parte de que el punto característico se localizó en el centro de la región indicada. Se asume también que cada una de las muestras corresponde con un píxel y se realizan algunas simplificaciones sobre el método propuesto por Lowe que se irán comentando a lo largo del ejemplo.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
2	0	0	0	0	5	0	0	0	0	0	5	0	0	0	0	5	0	0	0	0
3	5	5	5	5	0	5	5	10	10	10	0	5	5	10	10	0	5	5	5	0
4	0	0	5	10	5	10	15	0	0	0	5	10	15	0	0	5	0	0	10	5
5	5	10	5	5	5	10	20	15	15	15	5	10	20	15	15	5	10	10	5	0
6	5	10	10	15	20	25	200	25	25	25	20	25	200	25	25	20	10	10	15	5
7	0	0	5	5	10	130	150	150	150	20	10	130	150	150	150	10	0	0	5	0
8	0	0	5	5	5	120	140	140	140	10	5	120	140	140	140	5	0	0	5	0
9	5	10	5	5	5	100	100	100	100	10	5	100	100	100	100	5	10	10	5	0
10	0	5	5	10	100	120	250	250	130	130	100	120	250	190	130	100	5	5	10	0
11	5	5	10	10	120	130	250	250	120	120	120	130	250	250	120	120	5	5	10	0
12	5	10	5	100	160	200	150	250	110	110	160	200	150	250	110	160	100	10	5	0
13	0	0	5	130	180	200	210	120	120	255	255	200	210	120	120	180	130	0	5	0
14	5	10	0	0	190	210	200	120	120	255	255	210	200	120	120	190	10	10	0	0
15	0	0	0	0	200	180	190	130	130	130	200	180	190	130	130	200	0	0	0	10
16	0	5	10	10	200	190	170	140	140	140	200	190	170	140	140	200	5	5	10	0
17	0	5	5	5	10	0	5	0	0	0	10	0	5	0	0	10	5	5	5	0
18	0	0	10	10	0	5	0	0	0	0	0	5	0	0	0	0	0	0	10	0
19	5	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
20	0	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Figura 8.3. Región de la imagen sobre la que se calculará el descriptor SIFT

Solución: Como se explicó previamente el cálculo del descriptor se basa en (a) la obtención de la magnitud y la orientación del gradiente de la matriz 16x16 valores que rodean al punto característico, (b) el posterior cálculo del histograma de 8 bins con las orientaciones del gradiente para cada una de las 16 subregiones en las que se subdividen dichos 16x16 valores, y (c) la normalización del vector obtenido.

a) Obtención de la magnitud y la orientación del gradiente

En este primer paso, se asume que se ha seleccionado la escala de la imagen más similar a la escala del punto característico. Posteriormente se obtienen las derivadas horizontal y vertical. Se asume también que este cálculo se realizó previamente, en la etapa de *asignación de la orientación* del punto característico. En ese momento, se calculó la magnitud del gradiente, $m(x,y)$, y también su orientación, $\theta(x,y)$, utilizando diferencias entre los píxeles vecinos. Para facilitar el seguimiento de los cálculos, se ha omitido aplicar sobre la imagen de la figura 8.3 el filtrado gaussiano previo.

En primer lugar se calculan las derivadas horizontal y vertical, para cada píxel.

Derivada horizontal

L(x+1,y)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
2	0	0	0	0	5	0	0	0	0	0	5	0	0	0	0	5	0	0	0	0
3	5	5	5	0	5	5	10	10	10	0	5	5	10	10	0	5	5	5	5	0
4	0	0	10	5	10	15	0	0	0	5	10	15	0	0	5	0	0	10	10	5
5	5	10	5	5	10	20	15	15	15	5	10	20	15	15	5	10	10	5	5	0
6	5	10	15	20	25	200	25	25	25	20	25	200	25	25	20	10	10	15	15	5
7	0	0	5	10	130	150	150	150	20	10	130	150	150	150	10	0	0	5	5	0
8	0	0	5	5	120	140	140	140	10	5	120	140	140	140	5	0	0	5	5	0
9	5	10	5	5	100	100	100	100	10	5	100	100	100	100	5	10	10	5	5	0
10	0	5	10	100	120	250	250	130	130	100	120	250	190	130	100	5	5	10	10	0
11	5	5	10	120	130	250	250	120	120	120	130	250	250	120	120	5	5	10	10	0
12	5	10	100	160	200	150	250	110	110	160	200	150	250	110	160	100	10	5	5	0
13	0	0	130	180	200	210	120	120	255	255	200	210	120	120	180	130	0	5	5	0
14	5	10	0	190	210	200	120	120	255	255	210	200	120	120	190	10	10	0	0	0
15	0	0	0	200	180	190	130	130	130	200	180	190	130	130	200	0	0	0	0	10
16	0	5	10	200	190	170	140	140	140	200	190	170	140	140	200	5	5	10	10	0
17	0	5	5	10	0	5	0	0	0	10	0	5	0	0	10	5	5	5	5	0
18	0	0	10	0	5	0	0	0	0	0	5	0	0	0	0	0	0	10	10	0
19	5	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
20	0	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

L(x-1,y)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
2	0	0	0	0	5	0	0	0	0	0	5	0	0	0	0	5	0	0	0	0
3	5	5	5	5	5	0	5	5	10	10	10	0	5	5	10	10	0	5	5	0
4	0	0	0	5	10	5	10	15	0	0	0	5	10	15	0	0	5	0	10	5
5	5	10	10	5	5	5	10	20	15	15	15	5	10	20	15	15	5	10	5	0
6	5	10	10	10	15	20	25	200	25	25	25	20	25	200	25	25	20	10	15	5
7	0	0	0	5	5	10	130	150	150	150	20	10	130	150	150	150	10	0	5	0
8	0	0	0	5	5	5	120	140	140	140	10	5	120	140	140	140	5	0	5	0
9	5	10	10	5	5	5	100	100	100	100	10	5	100	100	100	100	5	10	5	0
10	0	5	5	5	10	100	120	250	250	130	130	100	120	250	190	130	100	5	10	0
11	5	5	5	10	10	120	130	250	250	120	120	120	130	250	250	120	120	5	10	0
12	5	10	10	5	100	160	200	150	250	110	110	160	200	150	250	110	160	100	5	0
13	0	0	0	5	130	180	200	210	120	120	255	255	200	210	120	120	180	130	5	0
14	5	10	10	0	0	190	210	200	120	120	255	255	210	200	120	120	190	10	0	0
15	0	0	0	0	0	200	180	190	130	130	130	200	180	190	130	130	200	0	0	10
16	0	5	5	10	10	200	190	170	140	140	140	200	190	170	140	140	200	5	10	0
17	0	5	5	5	5	10	0	5	0	0	0	10	0	5	0	0	10	5	5	0
18	0	0	0	10	10	0	5	0	0	0	0	0	5	0	0	0	0	0	10	0
19	5	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
20	0	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

L(x+1,y) -
L(x-1,y)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
2	0	0	0	0	5	0	0	0	0	0	5	0	0	0	0	5	0	0	0	0
3	5	5	0	-5	0	5	5	5	0	-10	-5	5	5	5	-10	-5	5	0	5	0
4	0	0	10	0	0	10	-10	-15	0	5	10	10	-10	-15	5	0	-5	10	10	5
5	5	10	-5	0	5	15	5	-5	0	-10	-5	15	5	-5	-10	-5	5	-5	5	0
6	5	10	5	10	10	180	0	-175	0	-5	0	180	0	-175	-5	-15	-10	5	15	5
7	0	0	5	5	125	140	20	0	-130	-140	110	140	20	0	-140	-150	-10	5	5	0
8	0	0	5	0	115	135	20	0	-130	-135	110	135	20	0	-135	-140	-5	5	5	0
9	5	10	-5	0	95	95	0	0	-90	-95	90	95	0	0	-95	-90	5	-5	5	0
10	0	5	5	95	110	150	130	-120	-120	-30	-10	150	70	-120	-90	-125	-95	5	10	0
11	5	5	5	110	120	130	120	-130	-130	0	10	130	120	-130	-130	-115	-115	5	10	0
12	5	10	90	155	100	-10	50	-40	-140	50	90	-10	50	-40	-90	-10	-150	-95	5	0
13	0	0	130	175	70	30	-80	-90	135	135	-55	-45	-80	-90	60	10	-180	-125	5	0
14	5	10	-10	190	210	10	-90	-80	135	135	-45	-55	-90	-80	70	-110	-180	-10	0	0
15	0	0	0	200	180	-10	-50	-60	0	70	50	-10	-50	-60	70	-130	-200	0	0	10
16	0	5	5	190	180	-30	-50	-30	0	60	50	-30	-50	-30	60	-135	-195	5	10	0
17	0	5	0	5	-5	-5	0	-5	0	10	0	-5	0	-5	10	5	-5	0	5	0
18	0	0	10	-10	-5	0	-5	0	0	0	5	0	-5	0	0	0	0	10	10	0
19	5	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
20	0	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Figura 8.4. Obtención de la derivada horizontal como diferencias entre los píxeles vecinos derecho e izquierdo.

Derivada vertical

L(x,y+1)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
2	0	0	0	0	5	0	0	0	0	0	5	0	0	0	0	5	0	0	0	0
3	5	5	0	0	5	0	0	0	0	0	5	0	0	0	0	5	0	0	5	0
4	0	0	5	5	0	5	5	10	10	10	0	5	5	10	10	0	5	5	10	5
5	5	10	5	10	5	10	15	0	0	0	5	10	15	0	0	5	0	0	5	0
6	5	10	5	5	5	10	20	15	15	15	5	10	20	15	15	5	10	10	15	5
7	0	0	10	15	20	25	200	25	25	25	20	25	200	25	25	20	10	10	5	0
8	0	0	5	5	10	130	150	150	150	20	10	130	150	150	150	10	0	0	5	0
9	5	10	5	5	5	120	140	140	140	10	5	120	140	140	140	5	0	0	5	0
10	0	5	5	5	5	100	100	100	100	10	5	100	100	100	100	5	10	10	10	0
11	5	5	5	10	100	120	250	250	130	130	100	120	250	250	130	100	5	5	10	0
12	5	10	10	10	120	130	250	250	120	120	120	130	250	250	120	120	5	5	5	0
13	0	0	5	100	160	200	150	250	110	110	160	200	150	250	110	160	100	10	5	0
14	5	10	5	130	180	200	210	120	120	255	255	200	210	120	120	180	130	0	0	0
15	0	0	0	0	190	210	200	120	120	255	255	210	200	120	120	190	10	10	0	10
16	0	5	0	0	200	180	190	130	130	130	200	180	190	130	130	200	0	0	10	0
17	0	5	10	10	200	190	170	140	140	140	200	190	170	140	140	200	5	5	5	0
18	0	0	5	5	10	0	5	0	0	0	10	0	5	0	0	10	5	5	10	0
19	5	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
20	0	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

L(x,y-1)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
2	0	0	0	0	5	0	0	0	0	0	5	0	0	0	0	5	0	0	0	0
3	5	5	5	10	5	10	15	0	0	0	5	10	15	0	0	5	0	0	5	0
4	0	0	5	5	5	10	20	15	15	15	5	10	20	15	15	5	10	10	10	5
5	5	10	10	15	20	25	200	25	25	25	20	25	200	25	25	20	10	10	5	0
6	5	10	5	5	10	130	150	150	150	20	10	130	150	150	150	10	0	0	15	5
7	0	0	5	5	5	120	140	140	140	10	5	120	140	140	140	5	0	0	5	0
8	0	0	5	5	5	100	100	100	100	10	5	100	100	100	100	5	10	10	5	0
9	5	10	5	10	100	120	250	250	130	130	100	120	250	250	130	100	5	5	5	0
10	0	5	10	10	120	130	250	250	120	120	120	130	250	250	120	120	5	5	10	0
11	5	5	5	100	160	200	150	250	110	110	160	200	150	250	110	160	100	10	10	0
12	5	10	5	130	180	200	210	120	120	255	255	200	210	120	120	180	130	0	5	0
13	0	0	0	0	190	210	200	120	120	255	255	210	200	120	120	190	10	10	5	0
14	5	10	0	0	200	180	190	130	130	130	200	180	190	130	130	200	0	0	0	0
15	0	0	10	10	200	190	170	140	140	140	200	190	170	140	140	200	5	5	0	10
16	0	5	5	5	10	0	5	0	0	0	10	0	5	0	0	10	5	5	10	0
17	0	5	10	10	0	5	0	0	0	0	0	5	0	0	0	0	0	0	5	0
18	0	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	10	0
19	5	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
20	0	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

L(x,y+1) -
L(x,y-1)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
2	0	0	0	0	5	0	0	0	0	0	5	0	0	0	0	5	0	0	0	0
3	5	5	-5	-10	0	-10	-15	0	0	0	0	-10	-15	0	0	0	0	0	5	0
4	0	0	0	0	-5	-5	-15	-5	-5	-5	-5	-5	-15	-5	-5	-5	-5	-5	10	5
5	5	10	-5	-5	-15	-15	-185	-25	-25	-25	-15	-15	-185	-25	-25	-15	-10	-10	5	0
6	5	10	0	0	-5	-120	-130	-135	-135	-5	-5	-120	-130	-135	-135	-5	10	10	15	5
7	0	0	5	10	15	-95	60	-115	-115	15	15	-95	60	-115	-115	15	10	10	5	0
8	0	0	0	0	5	30	50	50	50	10	5	30	50	50	50	5	-10	-10	5	0
9	5	10	0	-5	-95	0	-110	-110	10	-120	-95	0	-110	-110	10	-95	-5	-5	5	0
10	0	5	-5	-5	-115	-30	-150	-150	-20	-110	-115	-30	-150	-150	-20	-115	5	5	10	0
11	5	5	0	-90	-60	-80	100	0	20	20	-60	-80	100	0	20	-60	-95	-5	10	0
12	5	10	5	-120	-60	-70	40	130	0	-135	-135	-70	40	130	0	-60	-125	5	5	0
13	0	0	5	100	-30	-10	-50	130	-10	-145	-95	-10	-50	130	-10	-30	90	0	5	0
14	5	10	5	130	-20	20	20	-10	-10	125	55	20	20	-10	-10	-20	130	0	0	0
15	0	0	-10	-10	-10	20	30	-20	-20	115	55	20	30	-20	-20	-10	5	5	0	10
16	0	5	-5	-5	190	180	185	130	130	130	190	180	185	130	130	190	-5	-5	10	0
17	0	5	0	0	200	185	170	140	140	140	200	185	170	140	140	200	5	5	5	0
18	0	0	5	5	5	0	5	-10	-10	-10	5	0	5	-10	-10	5	5	5	10	0
19	5	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
20	0	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Figura 8.5. Obtención de la derivada vertical como diferencias entre los píxeles vecinos superior e inferior.

Una vez obtenidas las derivadas se obtiene la magnitud y la orientación del gradiente mediante las fórmulas indicadas en la ecuación 8.9.

Magnitud del gradiente $(L(x+1,y)-L(x-1,y))^2$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
2	0	0	0	0	5	0	0	0	0	0	5	0	0	0	0	5	0	0	0	0
3	5	5	0	25	0	25	25	25	0	100	25	25	25	25	100	25	25	0	5	0
4	0	0	100	0	0	100	100	225	0	25	100	100	100	225	25	0	25	100	10	5
5	5	10	25	0	25	225	25	25	0	100	25	225	25	25	100	25	25	25	5	0
6	5	10	25	100	100	32400	0	30625	0	25	0	32400	0	30625	25	225	100	25	15	5
7	0	0	25	25	15625	19600	400	0	16900	19600	12100	19600	400	0	19600	22500	100	25	5	0
8	0	0	25	0	13225	18225	400	0	16900	18225	12100	18225	400	0	18225	19600	25	25	5	0
9	5	10	25	0	9025	9025	0	0	8100	9025	8100	9025	0	0	9025	8100	25	25	5	0
10	0	5	25	9025	12100	22500	16900	14400	14400	900	100	22500	4900	14400	8100	15625	9025	25	10	0
11	5	5	25	12100	14400	16900	14400	16900	16900	0	100	16900	14400	16900	16900	13225	13225	25	10	0
12	5	10	8100	24025	10000	100	2500	1600	19600	2500	8100	100	2500	1600	8100	100	22500	9025	5	0
13	0	0	16900	30625	4900	900	6400	8100	18225	18225	3025	2025	6400	8100	3600	100	32400	15625	5	0
14	5	10	100	36100	44100	100	8100	6400	18225	18225	2025	3025	8100	6400	4900	12100	32400	100	0	0
15	0	0	0	40000	32400	100	2500	3600	0	4900	2500	100	2500	3600	4900	16900	40000	0	0	10
16	0	5	25	36100	32400	900	2500	900	0	3600	2500	900	2500	900	3600	18225	38025	25	10	0
17	0	5	0	25	25	25	0	25	0	100	0	25	0	25	100	25	25	0	5	0
18	0	0	100	100	25	0	25	0	0	0	25	0	25	0	0	0	0	100	10	0
19	5	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
20	0	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

$$(L(x,y+1)-L(x,y-1))^2$$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
2	0	0	0	0	5	0	0	0	0	0	5	0	0	0	0	5	0	0	0	0
3	5	5	25	100	0	100	225	0	0	0	0	100	225	0	0	0	0	0	5	0
4	0	0	0	0	25	25	225	25	25	25	25	25	225	25	25	25	25	25	10	5
5	5	10	25	25	225	225	34225	625	625	625	225	225	34225	625	625	225	100	100	5	0
6	5	10	0	0	25	14400	16900	18225	18225	25	25	14400	16900	18225	18225	25	100	100	15	5
7	0	0	25	100	225	9025	3600	13225	13225	225	225	9025	3600	13225	13225	225	100	100	5	0
8	0	0	0	0	25	900	2500	2500	2500	100	25	900	2500	2500	2500	25	100	100	5	0
9	5	10	0	25	9025	0	12100	12100	100	14400	9025	0	12100	12100	100	9025	25	25	5	0
10	0	5	25	25	13225	900	22500	22500	400	12100	13225	900	22500	22500	400	13225	25	25	10	0
11	5	5	0	8100	3600	6400	10000	0	400	400	3600	6400	10000	0	400	3600	9025	25	10	0
12	5	10	25	14400	3600	4900	1600	16900	0	18225	18225	4900	1600	16900	0	3600	15625	25	5	0
13	0	0	25	10000	900	100	2500	16900	100	21025	9025	100	2500	16900	100	900	8100	0	5	0
14	5	10	25	16900	400	400	400	100	100	15625	3025	400	400	100	100	400	16900	0	0	0
15	0	0	100	100	100	400	900	400	400	13225	3025	400	900	400	400	100	25	25	0	10
16	0	5	25	25	36100	32400	34225	16900	16900	16900	36100	32400	34225	16900	16900	36100	25	25	10	0
17	0	5	0	0	40000	34225	28900	19600	19600	19600	40000	34225	28900	19600	19600	40000	25	25	5	0
18	0	0	25	25	25	0	25	100	100	100	25	0	25	100	100	25	25	25	10	0
19	5	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
20	0	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

$$m(x,y)=\sqrt{(L(x+1,y)-L(x-1,y))^2+((L(x,y+1)-(L(x,y-1))^2}$$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0,0	0,0	5,0	5,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	5,0	0,0
2	0,0	0,0	0,0	0,0	5,0	0,0	0,0	0,0	0,0	0,0	5,0	0,0	0,0	0,0	0,0	5,0	0,0	0,0	0,0	0,0
3	5,0	5,0	5,0	11,2	0,0	11,2	15,8	5,0	0,0	10,0	5,0	11,2	15,8	5,0	10,0	5,0	5,0	0,0	5,0	0,0
4	0,0	0,0	10,0	0,0	5,0	11,2	18,0	15,8	5,0	7,1	11,2	11,2	18,0	15,8	7,1	5,0	7,1	11,2	10,0	5,0
5	5,0	10,0	7,1	5,0	15,8	21,2	185,1	25,5	25,0	26,9	15,8	21,2	185,1	25,5	26,9	15,8	11,2	11,2	5,0	0,0
6	5,0	10,0	5,0	10,0	11,2	216,3	130,0	221,0	135,0	7,1	5,0	216,3	130,0	221,0	135,1	15,8	14,1	11,2	15,0	5,0
7	0,0	0,0	7,1	11,2	125,9	169,2	63,2	115,0	173,6	140,8	111,0	169,2	63,2	115,0	181,2	150,7	14,1	11,2	5,0	0,0
8	0,0	0,0	5,0	0,0	115,1	138,3	53,9	50,0	139,3	135,4	110,1	138,3	53,9	50,0	144,0	140,1	11,2	11,2	5,0	0,0
9	5,0	10,0	5,0	5,0	134,4	95,0	110,0	110,0	90,6	153,1	130,9	95,0	110,0	110,0	95,5	130,9	7,1	7,1	5,0	0,0
10	0,0	5,0	7,1	95,1	159,1	153,0	198,5	192,1	121,7	114,0	115,4	153,0	165,5	192,1	92,2	169,9	95,1	7,1	10,0	0,0
11	5,0	5,0	5,0	142,1	134,2	152,6	156,2	130,0	131,5	20,0	60,8	152,6	156,2	130,0	131,5	129,7	149,2	7,1	10,0	0,0
12	5,0	10,0	90,1	196,0	116,6	70,7	64,0	136,0	140,0	144,0	162,2	70,7	64,0	136,0	90,0	60,8	195,3	95,1	5,0	0,0
13	0,0	0,0	130,1	201,6	76,2	31,6	94,3	158,1	135,4	198,1	109,8	46,1	94,3	158,1	60,8	31,6	201,2	125,0	5,0	0,0
14	5,0	10,0	11,2	230,2	211,0	22,4	92,2	80,6	135,4	184,0	71,1	58,5	92,2	80,6	70,7	111,8	222,0	10,0	0,0	0,0
15	0,0	0,0	10,0	200,2	180,3	22,4	58,3	63,2	20,0	134,6	74,3	22,4	58,3	63,2	72,8	130,4	200,1	5,0	0,0	10,0
16	0,0	5,0	7,1	190,1	261,7	182,5	191,6	133,4	130,0	143,2	196,5	182,5	191,6	133,4	143,2	233,1	195,1	7,1	10,0	0,0
17	0,0	5,0	0,0	5,0	200,1	185,1	170,0	140,1	140,0	140,4	200,0	185,1	170,0	140,1	140,4	200,1	7,1	5,0	5,0	0,0
18	0,0	0,0	11,2	11,2	7,1	0,0	7,1	10,0	10,0	10,0	7,1	0,0	7,1	10,0	10,0	5,0	5,0	11,2	10,0	0,0
19	5,0	0,0	0,0	0,0	5,0	0,0	0,0	10,0	10,0	10,0	5,0	0,0	0,0	10,0	10,0	5,0	0,0	0,0	0,0	0,0
20	0,0	0,0	0,0	0,0	5,0	0,0	0,0	10,0	10,0	10,0	5,0	0,0	0,0	10,0	10,0	5,0	0,0	0,0	0,0	0,0
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Figura 8.6. Obtención de la magnitud del gradiente para cada píxel de la región de interés.

Los valores obtenidos para la magnitud del gradiente deben ponderarse por una gaussiana que tenga una σ igual a la mitad del ancho de la ventana del descriptor. Para simplificar los cálculos y facilitar la

interpretación de los valores obtenidos en el cálculo del gradiente, en este ejercicio no realizaremos dicha ponderación.

Orientación del gradiente

A partir de las derivadas verticales y horizontales se obtiene, y se muestra en la figura 8.7, la orientación en cada píxel. Esta orientación se calculó como el arco cuya tangente es la relación entre la derivada vertical y la derivada horizontal, utilizando los valores calculados previamente mediante la fórmula expresada en la ecuación 9, que recordamos era la siguiente:

$$\theta(x,y) = \tan^{-1} \frac{L(x,y+1) - L(x,y-1))}{L(x+1,y) - L(x-1,y))}$$

Orientación en grados, entre 180 y -180, calculado a partir del arco tangente.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
2	0	0	0	0	5	0	0	0	0	0	5	0	0	0	0	5	0	0	0	0
3	5	5	-90	-117	0	-63	-72	0	0	0	0	-63	-72	0	0	0	0	0	5	0
4	0	0	0	0	-90	-27	-124	-162	-90	-45	-27	-27	-124	-162	-45	-90	-135	-27	10	5
5	5	10	-135	-90	-72	-45	-88	-101	-90	-112	-108	-45	-88	-101	-112	-108	-63	-117	5	0
6	5	10	0	0	-27	-34	-90	-142	-90	-135	-90	-34	-90	-142	-92	-162	135	63	15	5
7	0	0	45	63	7	-34	72	-90	-139	174	8	-34	72	-90	-141	174	135	63	5	0
8	0	0	0	0	2	13	68	90	159	176	3	13	68	90	160	178	-117	-63	5	0
9	5	10	0	-90	-45	0	-90	-90	174	-128	-47	0	-90	-90	174	-133	-45	-135	5	0
10	0	5	-45	-3	-46	-11	-49	-129	-171	-105	-95	-11	-65	-129	-167	-137	177	45	10	0
11	5	5	0	-39	-27	-32	40	0	171	90	-81	-32	40	0	171	-152	-140	-45	10	0
12	5	10	3	-38	-31	-98	39	107	0	-70	-56	-98	39	107	0	-99	-140	177	5	0
13	0	0	2	30	-23	-18	-148	125	-4	-47	-120	-167	-148	125	-9	-72	153	0	5	0
14	5	10	153	34	-5	63	167	-173	-4	43	129	160	167	-173	-8	-170	144	0	0	0
15	0	0	-90	-3	-3	117	149	-162	-90	59	48	117	149	-162	-16	-176	179	90	0	10
16	0	5	-45	-2	47	99	105	103	90	65	75	99	105	103	65	125	-179	-45	10	0
17	0	5	0	0	91	92	90	92	90	86	90	92	90	92	86	89	135	90	5	0
18	0	0	27	153	135	0	135	-90	-90	-90	45	0	135	-90	-90	90	90	27	10	0
19	5	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
20	0	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Orientación expresada en grados, entre 0 y 360

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
2	0	0	0	0	5	0	0	0	0	0	5	0	0	0	0	5	0	0	0	0
3	5	5	270	243	0	297	288	0	0	0	0	297	288	0	0	0	0	0	5	0
4	0	0	0	0	270	333	236	198	270	315	333	333	236	198	315	270	225	333	10	5
5	5	10	225	270	288	315	272	259	270	248	252	315	272	259	248	252	297	243	5	0
6	5	10	0	0	333	326	270	218	270	225	270	326	270	218	268	198	135	63	15	5
7	0	0	45	63	7	326	72	270	221	174	8	326	72	270	219	174	135	63	5	0
8	0	0	0	0	2	13	68	90	159	176	3	13	68	90	160	178	243	297	5	0
9	5	10	0	270	315	0	270	270	174	232	313	0	270	270	174	227	315	225	5	0
10	0	5	315	357	314	349	311	231	189	255	265	349	295	231	193	223	177	45	10	0
11	5	5	0	321	333	328	40	0	171	90	279	328	40	0	171	208	220	315	10	0
12	5	10	3	322	329	262	39	107	0	290	304	262	39	107	0	261	220	177	5	0
13	0	0	2	30	337	342	212	125	356	313	240	193	212	125	351	288	153	0	5	0
14	5	10	153	34	355	63	167	187	356	43	129	160	167	187	352	190	144	0	0	0
15	0	0	270	357	357	117	149	198	270	59	48	117	149	198	344	184	179	90	0	10
16	0	5	315	358	47	99	105	103	90	65	75	99	105	103	65	125	181	315	10	0
17	0	5	0	0	91	92	90	92	90	86	90	92	90	92	86	89	135	90	5	0
18	0	0	27	153	135	0	135	270	270	270	45	0	135	270	270	90	90	27	10	0
19	5	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
20	0	0	0	0	5	0	0	10	10	10	5	0	0	10	10	5	0	0	0	0
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Figura 8.7. Obtención de la orientación del gradiente para cada píxel de la región de interés.

b) Histograma de orientaciones para las 4x4 subregiones

Codificación de las orientaciones del gradiente

Para el cálculo del descriptor, las orientaciones del gradiente se codifican en ocho direcciones de manera que si la orientación se encuentra entre 0 y 44 grados se codifica como 1, entre 45 y 89 como 2 y así sucesivamente. La figura 8.8 presenta el resultado de aplicar dicha codificación a las orientaciones obtenidas previamente que se muestran en la figura 8.7.

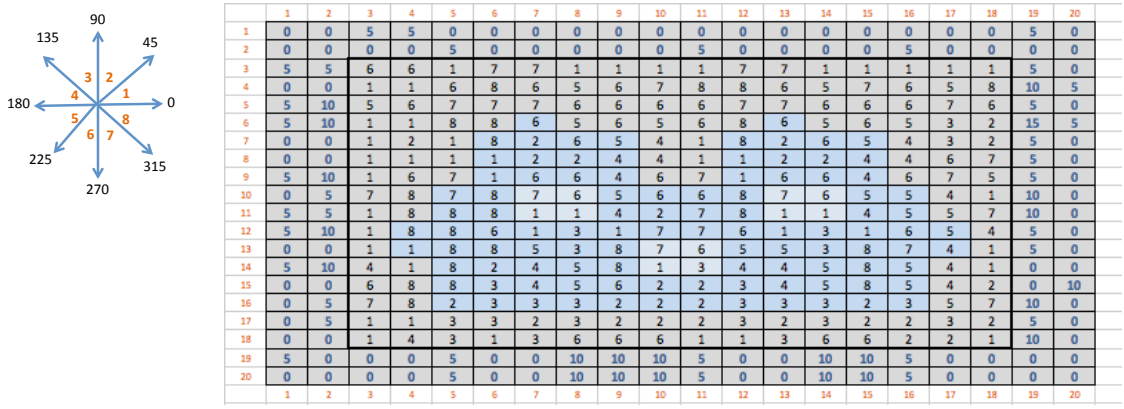


Figura 8.8. Codificación de las orientaciones del gradiente en 8 direcciones.

División de la región de interés en 4x4 subregiones

El descriptor SIFT se forma concatenando los histogramas de orientaciones obtenidos al dividir la región de interés en 4x4 subregiones. En la figura 8.9 se muestra esta subdivisión tanto para los valores de la magnitud del gradiente (a), como para su orientación (b).

(a)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0,0	0,0	5,0	5,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	5,0	0,0
2	0,0	0,0	0,0	0,0	5,0	0,0	0,0	0,0	0,0	0,0	5,0	0,0	0,0	0,0	0,0	5,0	0,0	0,0	0,0	0,0
3	5,0	5,0	5,0	11,2	0,0	11,2	15,8	5,0	0,0	10,0	5,0	11,2	15,8	5,0	10,0	5,0	5,0	0,0	5,0	0,0
4	0,0	0,0	10,0	0,0	5,0	11,2	18,0	15,8	5,0	7,1	11,2	11,2	18,0	15,8	7,1	5,0	7,1	11,2	10,0	5,0
5	5,0	10,0	7,1	5,0	15,8	21,2	185,1	25,5	25,0	26,9	15,8	21,2	185,1	25,5	26,9	15,8	11,2	11,2	5,0	0,0
6	5,0	10,0	5,0	10,0	11,2	216,3	130,0	221,0	135,0	7,1	5,0	216,3	130,0	221,0	135,1	15,8	14,1	11,2	15,0	5,0
7	0,0	0,0	7,1	11,2	125,9	169,2	63,2	115,0	173,6	140,8	111,0	169,2	63,2	115,0	181,2	150,7	14,1	11,2	5,0	0,0
8	0,0	0,0	5,0	0,0	115,1	138,3	53,9	50,0	139,3	135,4	110,1	138,3	53,9	50,0	144,0	140,1	11,2	11,2	5,0	0,0
9	5,0	10,0	5,0	5,0	134,4	95,0	110,0	110,0	90,6	153,1	130,9	95,0	110,0	110,0	95,5	130,9	7,1	7,1	5,0	0,0
10	0,0	5,0	7,1	95,1	159,1	153,0	198,5	192,1	121,7	114,0	115,4	153,0	165,5	192,1	92,2	169,9	95,1	7,1	10,0	0,0
11	5,0	5,0	5,0	142,1	134,2	152,6	156,2	130,0	131,5	20,0	60,8	152,6	156,2	130,0	131,5	129,7	149,2	7,1	10,0	0,0
12	5,0	10,0	90,1	196,0	116,6	70,7	64,0	136,0	140,0	144,0	162,2	70,7	64,0	136,0	90,0	60,8	195,3	95,1	5,0	0,0
13	0,0	0,0	130,1	201,6	76,2	31,6	94,3	158,1	135,4	198,1	109,8	46,1	94,3	158,1	60,8	31,6	201,2	125,0	5,0	0,0
14	5,0	10,0	11,2	230,2	211,0	22,4	92,2	80,6	135,4	184,0	71,1	58,5	92,2	80,6	70,7	111,8	222,0	10,0	0,0	0,0
15	0,0	0,0	10,0	200,2	180,3	22,4	58,3	63,2	20,0	134,6	74,3	22,4	58,3	63,2	72,8	130,4	200,1	5,0	0,0	10,0
16	0,0	5,0	7,1	190,1	261,7	182,5	191,6	133,4	130,0	143,2	196,5	182,5	191,6	133,4	143,2	233,1	195,1	7,1	10,0	0,0
17	0,0	5,0	0,0	5,0	200,1	185,1	170,0	140,1	140,0	140,4	200,0	185,1	170,0	140,1	140,4	200,1	7,1	5,0	5,0	0,0
18	0,0	0,0	11,2	11,2	7,1	0,0	7,1	10,0	10,0	10,0	7,1	0,0	7,1	10,0	10,0	5,0	5,0	11,2	10,0	0,0
19	5,0	0,0	0,0	0,0	5,0	0,0	0,0	10,0	10,0	10,0	5,0	0,0	0,0	10,0	10,0	5,0	0,0	0,0	0,0	0,0
20	0,0	0,0	0,0	0,0	5,0	0,0	0,0	10,0	10,0	10,0	5,0	0,0	0,0	10,0	10,0	5,0	0,0	0,0	0,0	0,0
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

(b)

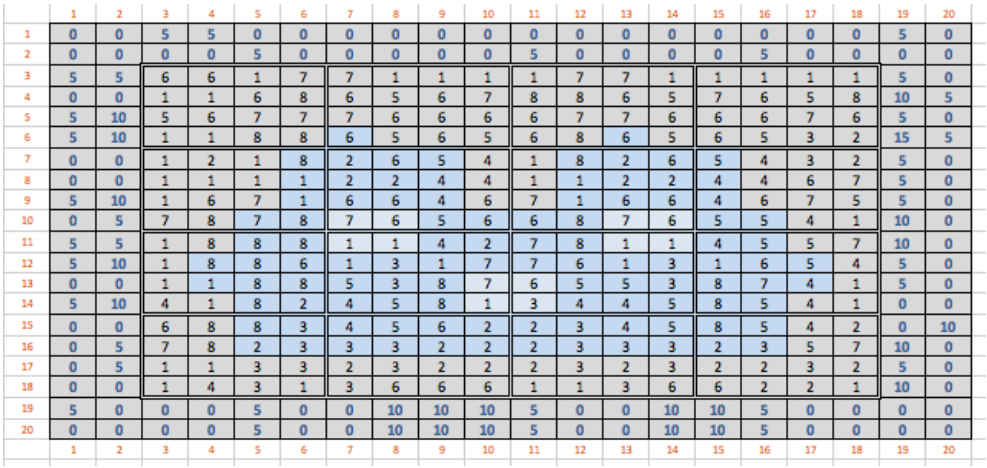


Figura 8.9. División de la región de interés en 4x4 subregiones. (a) Magnitud del gradiente y (b) Orientaciones del gradiente.

Histograma de 8 direcciones para las primeras 4 subregiones

Por claridad y para poder representar la información en el espacio disponible, se obtendrá el vector SIFT únicamente para las 4 primeras regiones de las 16 anteriores.

(a) (b) (c)

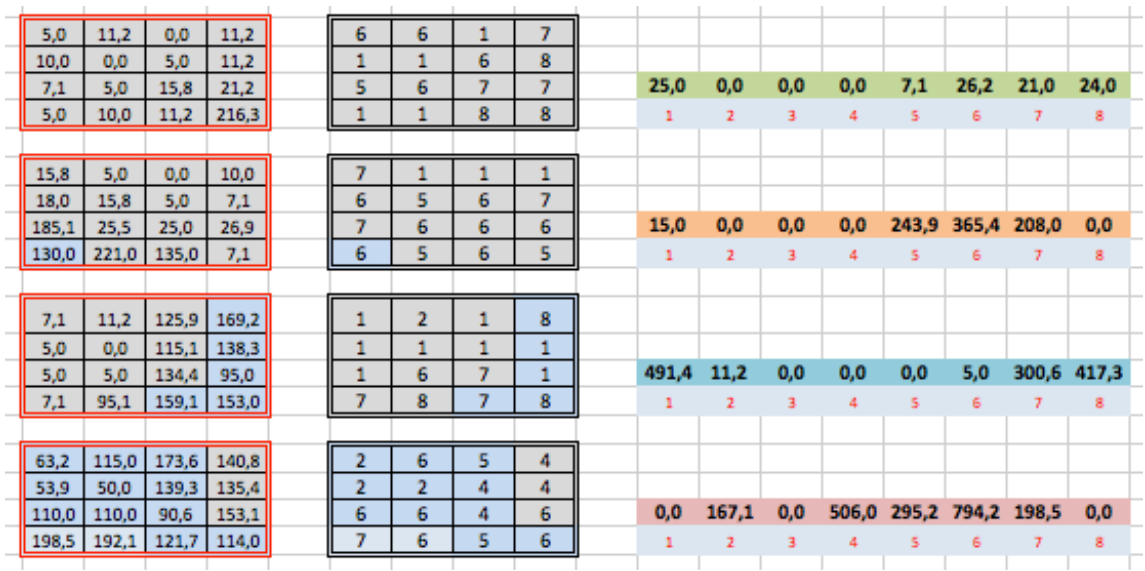


Figura 8.9. Obtención del histograma de orientaciones del gradiente para 4 subregiones. En columnas: (a) valores de la magnitud del gradiente, (b) orientaciones del gradiente, (c) histograma de orientaciones para 8 direcciones.

Como puede verse el histograma acumula para cada orientación la suma de los valores de la magnitud del gradiente de los píxeles con dicha orientación. Si nos fijamos, por ejemplo, en la primera región de la figura 8.9, puede verse que para la orientación 1, que correspondía de 0 a 44 grados, hay 1 píxel en la primera fila, 2 en la segunda y 2 en la cuarta. También puede observarse que el primer elemento del histograma (columna c de la figura 8.9) contiene la suma de los valores de dichas magnitudes que son 0,0 + 10,0 + 0,0 + 5,0 + 10, 0 = 25,0. La distribución del valor de cada gradiente

en los intervalos adyacentes del histograma mediante interpolación tri-lineal también se ha omitido en este ejercicio.

Una vez realizada dicha operación para las 16 subregiones, el vector de características se obtiene concatenando los histogramas para todas ellas. En nuestro caso, al calcularlo solo para 4 subregiones, se obtendría el vector de la figura 8.10.

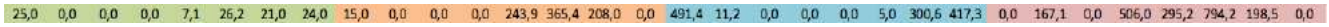


Figura 8.10. Vector de histogramas orientados de gradientes para las primeras 4 subregiones.c) Normalización del vector de características obtenido

Primera normalización a longitud unidad

Para realizar la primera normalización a longitud unidad se eleva cada elemento al cuadrado, se suman todos los elementos y se obtiene la raíz cuadrada, resultando una longitud, para el vector anterior de 1336,48.

Figura 8.11. Primera normalización del vector de histogramas orientados de gradientes para las primeras 4 subregiones.

Segunda normalización a longitud unidad

Para la realizar la segunda normalización, primero se truncan todos los valores a 0,2, como puede verse en la figura 8.11.

Figura 8.12. Segunda normalización del vector de características: valores truncados a 0,2.

Y después se normaliza nuevamente a longitud unidad. En este caso la longitud del vector es 0,6138 por lo que el vector SIFT resultante es el que se muestra en la figura 8.12.

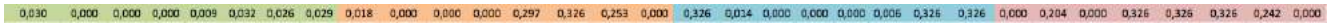


Figura 8.13. Segunda normalización de vector de características: nueva normalización a longitud unidad.

Haciendo este proceso para las 16 subregiones definidas alrededor del punto característico se obtiene el descriptor SIFT de cada keypoint.

8.3. Aplicación de SIFT al reconocimiento de objetos

Una de las principales aplicaciones de los métodos basados en características locales invariantes, como es el caso de SIFT, es el reconocimiento de objetos, y especialmente cuando hay oclusión o el fondo está lleno de objetos desordenados (*clutter*).

Cuando se utiliza este tipo de técnicas, el reconocimiento de un objeto se basa en encontrar correspondencias correctas entre el suficiente número de puntos característicos del objeto "consulta" con uno o varios de los objetos típicamente almacenados en una base de datos.

Inicialmente se realiza la correspondencia entre cada uno de los descriptores de los puntos característicos de la imagen consulta contra la base de datos que contiene todos los descriptores de los puntos característicos extraídos del conjunto de imágenes de entrenamiento. Muchas de las correspondencias iniciales serán erróneas debido a características que son ambiguas o que proceden de un fondo lleno de objetos desordenados. Para mejorar el reconocimiento, se identifican grupos de al menos tres características que sean coherentes con un objeto y su pose, asumiendo que es mucho más probable que estos grupos sean más correctos que las correspondencias obtenidas individualmente. Después, se comprueba cada grupo mediante un ajuste geométrico detallado al modelo, de manera que, en función del resultado de este ajuste, se rechaza o acepta la correspondencia entre los objetos.

8.3.1. Correspondencias de vecinos más cercanos

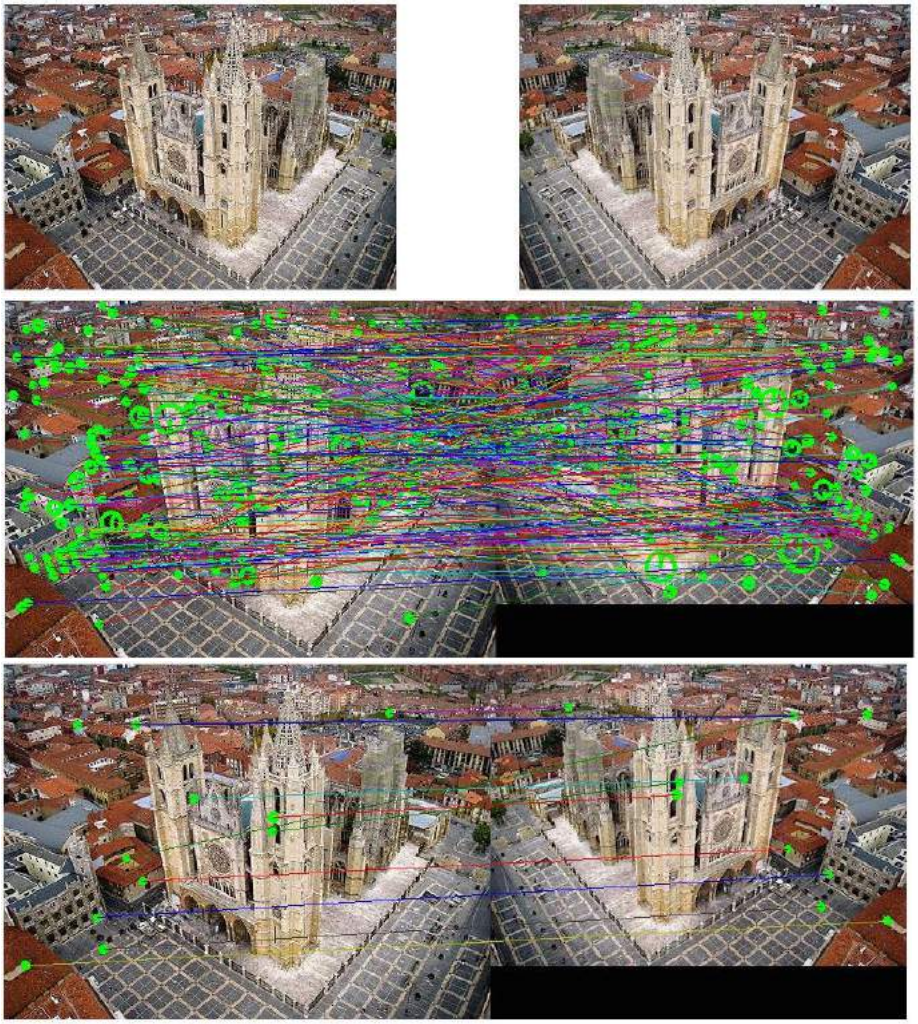
Como comentábamos antes, la mejor correspondencia candidata para cada punto característico viene dada por el vecino más cercano a su descriptor de entre todos los puntos almacenados en la base de datos que contiene los descriptores de las imágenes de entrenamiento. Para SIFT, el vecino más cercano se define como aquel punto característico con una distancia Euclídea mínima entre su vector de características invariante, que es su descriptor, y el de la consulta. Un ejemplo de estas correspondencias puede verse en la figura 8.14.

Para descartar características que no tienen una buena correspondencia Lowe (2004) evaluó la posibilidad de utilizar un umbral global llegando a la conclusión de que no es una buena medida ya que algunos descriptores son más discriminativos que otros. Por ello, propuso utilizar una medida obtenida al comparar la distancia entre el vecino más cercano y el segundo vecino más cercano. En el caso de que en el conjunto de entrenamiento haya más de una imagen del mismo objeto, se define el segundo vecino más cercano como aquel que procede de un objeto diferente al primer vecino, utilizando para ello imágenes de las que se conozca que contienen diferentes objetos. Según Lowe (2004) esta medida funciona bien porque, para que sea fiable, las correspondencias correctas necesitan tener el vecino más cercano significativamente más próximo que la más cercana de las correspondencias indirectas (ver figura 8.14).

Para la implementación realizada, Lowe (2004) rechazó todas las correspondencias en las que el ratio de distancias de la segunda sobre la primera fuera mayor de 0.8. Con eso, se eliminaron el 90% de las correspondencias erróneas, mientras que se descartaron menos del 5% de correspondencias correctas.

8.3.2. Correspondencia entre puntos mediante la aproximación del mejor-recipiente-primero

Ante la falta de algoritmos que pudieran identificar el vecino más cercano en espacios de dimensionalidad alta, que fueran más eficientes que una búsqueda exhaustiva, Lowe (2004) utilizó un algoritmo previamente publicado conjuntamente con Beis (Beis y Lowe, 1997). El algoritmo, llamado Best-Bin-First (BBF), que puede traducirse como el mejor-recipiente-primero, utiliza un ordenamiento modificado de la búsqueda para el algoritmo k-d tree, de manera que en el espacio de características los recipientes, *bins,* se buscan en el orden de la distancia más próxima a la ubicación de la consulta. En el artículo original de Lowe (2004) pueden encontrarse algunos detalles sobre su implementación y una pequeña discusión sobre por qué es apropiado su uso en este problema.



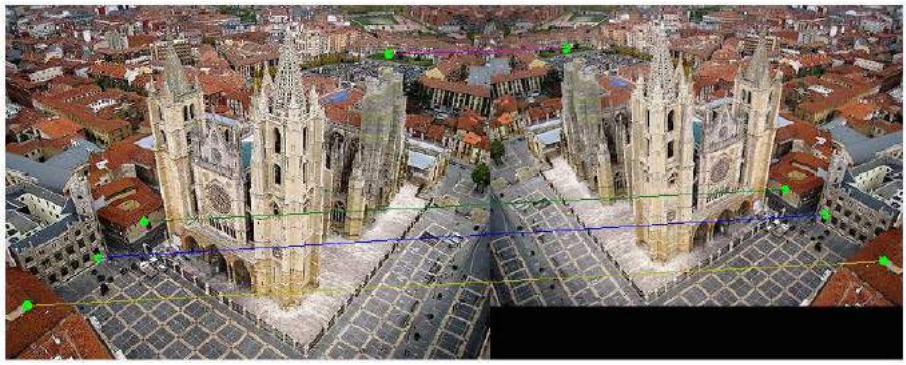


Figura 8.14. Ejemplo de dos imágenes del mismo objeto, la catedral de León, en distintas poses y las correspondencias de descriptores SIFT encontradas. En la primera fila las *imágenes originales*. En la segunda fila aparecen todas las correspondencias encontradas mediante el *primer vecino más cercano*. En la tercera fila se ha aplicado *el test del segundo vecino más cercano* a cada correspondencia. En la cuarta, se ha impuesto una *verificación geométrica* a las correspondencias.

8.3.3. Reconocimiento de objetos mediante la transformada de Hough afin

Para aplicar el descriptor SIFT al reconocimiento de objetos Lowe (2004) propuso utilizar una técnica basada en la transformada de Hough donde se aceptaba como válida una correspondencia cuando al menos tres puntos de un objeto, representados por sus correspondientes descriptores SIFT votaban a favor de ello. Pretendía hacer factible la identificación de objetos con el menor número posible de correspondencias entre características, pensando especialmente en el reconocimiento de objetos pequeños o con mucha oclusión.
La correspondencia de descriptores SIFT se ha establecido como la técnica base del estado del arte para el reconocimiento y recuperación de objetos. En el estudio realizado por Mikolajczyk y Schmid (2005), estos autores llegaron a la conclusión de que el descriptor SIFT es más robusto a deformaciones de la imagen que muchas de las otras técnicas que se estaban usando hasta ese momento, como son los filtros dirigidos, los invariantes diferenciales y complejos o los momentos invariantes, entre otros.

8.4. Extensiones de SIFT

Después de la aparición de SIFT diversos autores lo extendieron de manera que funcionara mejor en problemas para los que no había sido concebido inicialmente, como son las imágenes en color, el muestreo denso, el vídeo o la reducción de su dimensionalidad.

La primera propuesta para aplicar SIFT a imágenes en color la realizó Bosch y otros (2006) que calcularon los descriptores SIFT sobre los tres canales de la imagen tras convertirla al espacio de color HSV, obteniendo así un vector de 3x128 dimensiones. Otra propuesta similar la realizó Van de Weijer y Schmid (2006) quienes concatenaron el descriptor SIFT original con histogramas ponderados del tono o de ángulo oponente, descubriendo que mejoraba las correspondencias de puntos al evaluarlo en diferentes datasets. Otra extensión de SIFT que utiliza el color fue propuesta por Burghouts y Geusebroek (2009). Estos autores definieron un conjuntos de descriptores de imagen que se basan en invariantes de color expresados mediante un modelo de color Gaussiano. Su idea consistió en

reemplazar el gradiente de escala de grises que utiliza SIFT por diversos gradientes de color invariantes a combinaciones de niveles de intensidad local, sombras y zonas iluminadas. Llamaron a su descriptor C-colour-SIFT y comprobaron que funcionaba mejor que los métodos comentados previamente para clasificación de imágenes en categorías y para correspondencia de puntos de interés. En relación con extensiones basadas en color, comentar que uno de los trabajos más completos es el estudio de las propiedades invariantes de diferentes representaciones de color realizado por Koen y sus colegas (Van de Sande y otros, 2010). En este trabajo se estudian representaciones de color basadas en histogramas, momentos e invariantes de color juntamente con descriptores de color basados en SIFT. Su conclusión fue que el descriptor OpponentSIFT es el que mejor rendimiento produce para clasificación de categorías de objetos.

Además de las extensiones que añaden el color a SIFT han aparecido otras como son PCA-SIFT (Ke y Suktahnakar, 2004) que primero utilizan regiones de interés alrededor del keypoint mayores, de 39x39 píxeles, con el objetivo de obtener invarianza a la escala y luego, debido a la alta dimensionalidad, la reducen mediante análisis de componentes principales (PCA) a un vector de 20 dimensiones, siendo, según sus atores, más rápido y distintivo que SIFT.

Y finalmente comentar que una de las extensiones de SIFT más utilizadas ha sido denseSIFT, principalmente para clasificar imágenes en categorías. Este método consiste en calcular el descriptor SIFT sobre una rejilla densa colocada sobre la imagen, en lugar de hacerlo alrededor de los puntos de interés detectados. El motivo para hacerlo es que se considera que de esta forma se obtiene más información que la que se consigue con los mismos descriptores evaluados en los puntos característicos que se encontrarán, sin duda, más dispersos. Este planteamiento lo inició Bosch y otros (2006) y en la actualidad es el procedimiento habitual seguido en problemas relacionados con clasificación de imágenes en categorías.

8.5. Descriptores de imagen relacionados

En los últimos años han aparecido numerosas propuestas de métodos basados total o parcialmente en los conceptos subyacentes en SIFT que son: localización de puntos característicos y descripción de la región de interés, alrededor de dichos puntos, mediante algún método generalmente basado en histogramas de orientaciones del gradiente. En esta sección se comenta brevemente cuatro de los más establecidos, aunque descriptores como FERNS, DAISY, FAST, ORB, BRISK o BRIEF son otras propuestas recientes que se postulan como más rápidas, más eficientes o más apropiadas para diversos problemas.

8.5.1. Histogramas de campo receptivo

Las propuestas de descriptores de la imagen basados en el histograma se inició a principios de los años 90 cuando Swain y Ballard (Swain y Ballard, 1991) mostraron que era posible reconocer objetos si se comparaban histogramas RGB de las imágenes de dichos objetos sin tener en cuenta ningún tipo de relación espacial entre las características de la imagen. Una propuesta posterior (Shiele and Crowley, 2000) confirmó que también era posible hacerlo utilizando tanto histogramas de derivadas

parciales de primer orden, como combinaciones de las respuestas del Laplaciano y magnitudes del gradiente calculadas a múltiples escalas.

Esta propuesta ha sido generalizada aún más por Linde y Lindeberg (Linde y Lindeberg 2004; Linde y Lindeberg 2012) quienes propusieron histogramas de campo receptivo más general a partir de combinaciones de derivadas de Gaussianas o invariantes diferenciales. Posteriormente, realizaron una evaluación exhaustiva de este tipo de descriptores basados en histograma en cuanto a su rendimiento para reconocimiento y clasificación de objetos.

8.5.2. HoG (Histogram of oriented gradients)

HoG, conocido como Histograma de Gradientes Orientados, fue propuesto por Delal y Triggs (Delal y Triggs, 2005) en el CVPR'05 (Computer Vision and Pattern Recognition Conference). El método se basa en la idea de que la apariencia local y la forma de un objeto pueden describirse mediante la distribución de gradientes de intensidad o direcciones de los bordes. La técnica cuenta el número de ocurrencias de una orientación del gradiente en determinadas regiones de la imagen. El método tiene similitudes con SIFT (Lowe, 1999) y con *contextos de forma* (Belongie y otros, 2002) aunque se diferencia de estos métodos en que se calcula en una rejilla densa de celdas espaciadas uniformemente y que utiliza una normalización del contraste local para incrementar la precisión.

Inicialmente esta técnica (Delal y Triggs, 2005) fue propuesta para detectar peatones en imágenes estáticas pero su empleo se ha extendido a la detección de coches y diversos animales comunes, tanto en vídeo como en imágenes estáticas.

8.5.3. SURF (Speed up robust features)

SURF es un descriptor propuesto por Bay y otros (2006) que es similar a SIFT en cuanto que se basa en obtener puntos característicos y posteriormente describir la región alrededor de dichos puntos. Este método se caracteriza porque el espacio escala lo crea mediante wavelets de Haar, en lugar de una pirámide de diferencias de gaussianas. Además, los puntos característicos se detectan con el determinante del Hessiano y la región de interés se describe como sumas, absolutas y no, de derivadas de primer orden, en lugar de un histograma orientado de gradientes.

Si bien se considera que la capacidad de detectar y caracterizar regiones es similar o un poquito inferior a SIFT, es más rápido al basar su detección en wavelets de Haar.

8.5.4. GLOH (Gradient location and orientation histogram)

La técnica del histograma de orientaciones y ubicaciones del gradiente fue propuesta por Mikolajczyk y Schmid (2005) como una alternativa a SIFT. Se parece al método propuesto por Lowe en cuanto a que se basa en un histograma local de orientaciones del gradiente alrededor de un punto de interés. Se diferencia, de dicho método, en que:

- El histograma se calcula sobre una rejilla polar en lugar de una rejilla rectangular.
- Utiliza 16 bins para cuantizar las direcciones del grandiente en lugar de los 8 bins que utiliza SIFT.

- Utiliza análisis de componentes principales (PCA) para reducir la dimensionalidad del descriptor de la imagen.

Según sus autores, este descriptor funciona mejor, ofreciendo mayores tasas de acierto en correspondencias de puntos cuando la imagen contiene escenas estructuradas, mientras que SIFT funciona mejor cuando las escenas tienen textura.

8.6. Bibliografía

Attneave, F. (1954) "Some informational aspects of visual perception," Psychological Review, vol. 61, pp. 183–193.

Bay, H.; Tuytelaars, T. and van Gool, Luc (2006). SURF: Speeded up robust features. Proc. 9th European Conference on Computer Vision (ECCV'06) Springer Lecture Notes in Computer Science 3951: 404-417. Doi: 10.1007/11744023_32.

Beis, J.S.; Lowe, D.G., "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," Computer Vision and Pattern Recognition, 1997. Proceedings, 1997 IEEE Computer Society Conference on , vol., no., pp.1000,1006, 17-19 Jun 1997. doi: 10.1109/CVPR.1997.609451

Belongie, S. ; Malik, J. and Puzicha J. (2002). "Shape Matching and Object Recognition Using Shape Contexts". IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (24): 509–521. doi:10.1109/34.993558.

Bosch, A.; Zisserman, A. and Munoz, X. (2006). Scene classification via pLSA. Proc. 9th European Conference on Computer Vision (ECCV'06) Springer Lecture Notes in Computer Science 3954: 517~530.

Brown, M. and Lowe, D. G. "Invariant features from interest point groups," British Machine Vision Conference, BMVC 2002, Cardiff, Wales (September 2002), pp. 656-665.

Burt, Peter and Adelson, Ted (1983). The Laplacian pyramid as a compact image code. IEEE Transactions on Communications 9(4): 532–540. doi:10.1109/tcom.1983.1095851.

Burghouts, G. J. and Geusebroek, J-M (2009). Performance evaluation of local colour invariants. Computer Vision and Image Understanding 113: 48-62. doi:10.1016/j.cviu.2008.07.003.

Crowley, J. L. and Stern, R. M. (1984). Fast computation of the difference of low pass transform. IEEE Transactions on Pattern Analysis and Machine Intelligence 6(2): 212-222. doi:10.1109/tpami.1984.4767504.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. Proc. Computer Vision and Pattern Recognition (CVPR'05) (San Diego, CA): I:886-893.

Edelman, S., Intrator, N. and Poggio, T. (1997). Complex cells and object recognition. Unpublished manuscript: http://kybele.psych.cornell.edu/~edelman/Archive/nips97.pdf

Harris, C. and Stephens, M. (1988). "A combined corner and edge detector". Proceedings of the 4th Alvey Vision Conference. pp. 147–151.

Ikeuchi, K. (ed.) (2014). Computer Vision: A Reference Guide, Springer, pages 701–713. DOI: 10.1007/978-0-387-31439-6.

Ke, Y. and Sukthankar, R. (2004). PCA-SIFT: A more distinctive representation for local image descriptors. Proc. Computer Vision and Pattern Recognition (CVPR'04) (Pittsburgh, PA): II:506-513.

Koenderink, J.J. (1984). The structure of images. Biological Cybernetics, 50:363-396.
Lindeberg, T. (1994). Scale-space theory: A basic tool for analysing structures at different scales. Journal of Applied Statistics, 21(2):224-270.
Lindeberg, T. (2012) Scale Invariant Feature Transform. Scholarpedia, 7(5):10491.
Linde, O. and Lindeberg, T. (2004). Object recognition using composed receptive field histograms of higher dimensionality. Proc 17th International Conference on Pattern Recognition (ICPR'04) (Cambridge, U.K.): I:1-6. doi:10.1109/ICPR.2004.1333965.
Linde, O. and Lindeberg, T. (2012). Composed complex-cue histograms: An investigation of the information content in receptive field based image descriptors for object recognition. Computer Vision and Image Understanding 116: 538-560. doi:10.1016/j.cviu.2011.12.003.
Lowe, D. G. (1999). "Object recognition from local scale-invariant features". Proceedings of the International Conference on Computer Vision 2. pp. 1150–1157. doi:10.1109/ICCV.1999.790410.
Lowe, D. G., (2004) “Distinctive Image Features from Scale-Invariant Keypoints”, International Journal of Computer Vision, 60, 2, pp. 91-110.
Mikolajczyk, K., and Schmid, C. (2002). An affine invariant interest point detector. In European Conference on Computer Vision (ECCV), Copenhagen, Denmark, pp. 128-142.
Mikolajczyk, K and Schmid, C. (2005). A performance evaluation of local descriptors. International IEEE Transactions on Pattern Analysis and Machine Intelligence 27(19): 1615--1630. doi:10.1109/tpami.2005.188.
Schiele, B. and Crowley, J. L. (2000). Recognition without correspondence using multidimensional receptive field histograms. International Journal of Computer Vision 26(1): 31-50. doi:10.1007/bfb0015571.
Swain, M. J. and Ballard, D. H. (1991). Color indexing. International Journal of Computer Vision 7(1): 11-32. doi:10.1007/bf00130487.
SIFT Tutorial. AI Shack web page. Disponible online: http://www.aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/. (accedido 10 abril 2015)
Scale space. (2014, November 28). In Wikipedia, The Free Encyclopedia. Disponible online: http://en.wikipedia.org/w/index.php?title=Scale_space&oldid=635767690. (accedido 20 abril 2015).
Tuytelaars, T and Mikolajczyk, K (2008) Local invariant feature detectors: A survey. Foundations and Trends in Computer Graphics and Vision, 3 (3). 177 - 280. ISSN 1572-2759.
Van de Sande, K.; Gevers, Th. and Jan-Snoek, C. G. M. (2010). Evaluating color descriptors for object and scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 32(9): 1582-1596. doi:10.1109/tpami.2009.154.
Van de Weijer, J. and Schmid, C. (2006). Coloring local feature extraction. Proc. 9th European Conference on Computer Vision (ECCV'06) Springer Lecture Notes in Computer Science 3952: 334-348. doi:10.1007/11744047_26.

CAPÍTULO 9

CLASIFICACIÓN Y RECONOCIMIENTO DE PATRONES

María T. GARCÍA-ORDÁS[1], Rocío ALAIZ-RODRÍGUEZ[1], Enrique ALEGRE[1]

[1] Departamento de Ingeniería Eléctrica y de Sistemas y Automática, Universidad de León, León, España

En este capítulo se presentan las ideas básicas de la etapa de clasificación en un sistema de reconocimiento de patrones. Comienza el capítulo recordando los fundamentos del aprendizaje a partir de ejemplos para, posteriormente, hacer una revisión de las métricas y métodos más habituales de evaluación del rendimiento de un clasificador. El capítulo continúa mostrando el ciclo completo de diseño de un clasificador y finalmente, se describen, a modo de ilustración, tres modelos de aprendizaje correspondientes a los enfoques de clasificación supervisada, regresión y clasificación no supervisada.

9.1 Introducción

Los avances tecnológicos de las últimas décadas han hecho posible automatizar muchas tareas que previamente requerían una cantidad significativa de tiempo de trabajo manual repetitivo, mejorando la velocidad y reduciendo errores en esas tareas esencialmente mecánicas.

Actualmente, la tecnología nos permite disponer, almacenar y procesar gran número de datos. Igualmente, los investigadores han desarrollado nuevos modelos y algoritmos para trabajar con estos datos. Todo ello, junto con el creciente interés comercial e industrial por obtener información y conocimiento de los datos recopilados, ha permitido la automatización de tareas que no son meramente mecánicas sino que requieren inteligencia en mayor o menor grado.

Algunas de estas tareas, como el reconocimiento de rostros, no requiere esfuerzo alguno para un ser humano. Este proceso lo realizamos a diario, reconociendo a familiares o conocidos, independientemente de diversos factores que lo pueden dificultar, como son la postura, la iluminación

o el estilo de peinado, entre otros. Lo hacemos de forma inconsciente y nos resulta imposible explicar cómo lo llevamos a cabo. De ahí, la dificultad para escribir un programa para que se realice automáticamente.

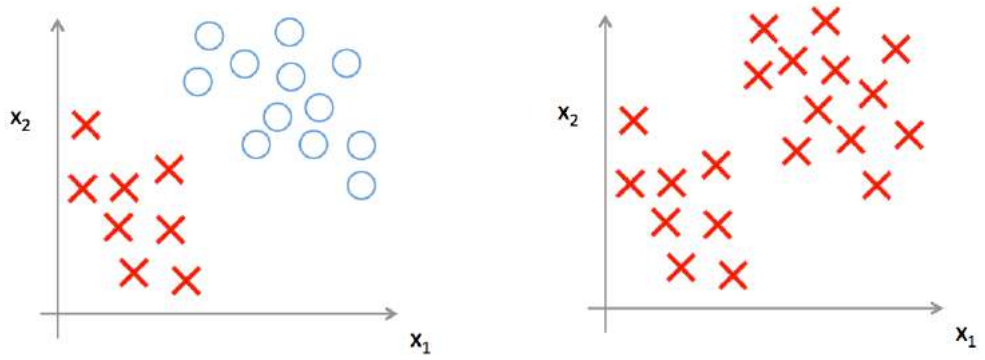
Hay otras tareas, sin embargo, que a las personas les resultan mucho más complejas e incluso difícilmente llevan a cabo con éxito. Esto se debe, en parte, al gran volumen de datos que hay que manejar y a la presencia de patrones poco evidente, no lineales muchas veces, que se deben analizar simultáneamente.

Las técnicas de aprendizaje automático y minería de datos han conseguido grandes avances en esta dirección, haciendo posible que los sistemas inteligentes sean una parte, a veces muy importante, del modelo de negocio de muchas empresas.

9.2 Fundamentos del Reconocimiento de Patrones

El reconocimiento de patrones consiste básicamente en asignar etiquetas a objetos indicando a qué clase pertenecen. Estos objetos deben ser representados por un conjunto de medidas, a las que nos referiremos como atributos o características. La tarea de reconocimiento implica necesariamente un proceso de aprendizaje a partir de un conjunto de objetos (conjunto de datos). Los fundamentos de este campo se establecieron ya en los años sesenta y setenta, con excelentes libros (Duda y Hart, 1973; Fukunaga, 1972), que dieron forma a este campo de investigación.

Hay dos grandes tipos de problemas de reconocimiento de patrones: supervisado y no supervisado. En el aprendizaje supervisado se dispone de un conjunto de datos junto con sus etiquetas, indicando a qué clase pertenecen, para cada dato o ejemplo. El objetivo que se persigue en este caso es conseguir un modelo (clasificador) que pueda etiquetar automáticamente nuevos datos que no se hayan empleado en el ajuste del modelo de clasificación. Este proceso es conocido como entrenamiento del clasificador o creación del modelo. Si las etiquetas no tienen un valor discreto, sino que toman valores continuos, hablaremos, entonces, de regresión. El reconocimiento óptico de caracteres sería un ejemplo de clasificación supervisada, mientras que la estimación de la edad de una persona a partir del rostro constituiría un problema de regresión.



(a) (b)

Figura 9.1 Ejemplo de clasificación supervisada (a) y no supervisada (b). Los datos se representan por dos características x_1 y x_2.

En las técnicas de aprendizaje no supervisado sólo se dispone de los datos sin las etiquetas de clase. Tratan de modelar los datos en sí mismos, lo que se traduce normalmente en descubrir grupos (clustering), es decir, determinar si hay ejemplos similares en el conjunto de datos y qué características hacen que estos datos sean similares dentro del grupo y diferentes del resto.

En la figura 9.1 (a) se puede ver un ejemplo de clasificación supervisada. Conocemos la clase a la que pertenecen los datos que se encuentran en el espacio de representación (clase roja y clase azul). En la figura 9.1 (b), sin embargo, no se conoce la clase a la que pertenecen los datos disponibles y buscamos técnicas que los agrupen en función de sus características.

9.2.1 Concepto de clase, características y conjunto de datos

En un problema de reconocimiento de patrones, cada muestra se identifica por un conjunto de características representada por un vector n-dimensional $\boldsymbol{x} = [x_1,...,x_n]$. Estas características pueden ser cuantitativas o cualitativas.

En el caso más extendido de aprendizaje supervisado, tendremos varias clases predefinidas, de forma que un objeto pertenece a una y sólo a una de estas clases. Cada una de las clases tiene objetos similares que son diferentes de los de las otras clases. Así, por ejemplo, en un contexto de autentificación biométrica a través de la escritura, tendríamos un problema de aprendizaje supervisado con dos clases: una determinada letra corresponde o no corresponde al usuario cuya identidad debemos verificar.

Asumiremos que hay L posibles clases, mutuamente excluyentes, etiquetadas como $u_0,...,u_{L-1}$, y organizadas en un conjunto $U_L = \{u_0,...,u_{L-1}\}$.

Dispondremos, pues, de un conjunto de muestras etiquetadas, $S = \{(\boldsymbol{x}^k, d^k), k = 1,...,N$ donde $\boldsymbol{x}^k$ es un objeto del conjunto de datos y $d^k \in U_L$ representa la etiqueta de dicha muestra. Si la clasificación fuera no supervisada no se dispondría de dichas etiquetas.

9.2.2 La clasificación

El clasificador es una función f_θ que establece la relación entre los datos de entrada $\boldsymbol{x}$ y las etiquetas d para dichos datos.

En algunos casos se trabaja con clasificadores que realizan la clasificación en dos etapas: calculan una decisión blanda y, a partir de la cual se toma la decisión dura final $\hat{d}$. Las salidas blandas vienen dadas por $y^k = f_\theta(\boldsymbol{x}^k)$ donde f_θ es una función con parámetros θ.

El algoritmo de entrenamiento ajusta los parámetros θ del clasificador de tal forma que se minimice una determinada función de error o función de coste.

9.2.3 Evaluación del clasificador

Aunque la función básica del clasificador es clara (discriminación entre dos o más clases mutuamente excluyentes), no ocurre lo mismo con la forma de evaluar las prestaciones de la clasificación realizada. Esto dificulta, por un lado, la decisión sobre cómo diseñar el clasificador (estrategia de aprendizaje, ajuste fino de los parámetros) y por otro, la comparación entre distintos clasificadores.

El resultado de la clasificación se puede resumir en una matriz de confusión con columnas y filas correspondientes a la clase pronosticada y a la real, respectivamente. Para un problema de clasificación binario (dos clases, etiquetadas como u_0, u_1) esta matriz tiene la estructura reflejada en la Figura 9.2 donde:

- TN (True Negative): Ejemplos de la clase 0 correctamente clasificados
- FN (False Negative): Ejemplos de la clase 1 clasificados incorrectamente.
- TP (True Positive): Ejemplos de la clase 1 clasificados correctamente
- FP (False Positive): Ejemplos de la clase 0 clasificados incorrectamente

		Predicción	
		u_1	u_0
Clase Real	u_1	TP	FN
	u_0	FP	TN

Figura 9.2 Matriz de confusión para un caso binario.

El número total de ejemplos positivos es $P = FN + TP$ y el número total de ejemplos negativos es $N = FP + TN$.

True Positive Rate (También llamado "hit rate", "*recall*", "Sensitivity" o "TP Rate").	TP/P	Proporción de muestras positivas que han sido correctamente clasificadas.
False Positive Rate (También llamado "False Alarm Rate" o "FP Rate").	FP/N	Proporción de muestras negativas que han sido erróneamente clasificadas como positivas.
False Negative Rate (También llamado "FN Rate").	FN/P	Proporción de muestras positivas que han sido erróneamente clasificadas como negativas (1 – TP Rate).
True Negative Rate (También llamado "Specificity" o "TN Rate").	TN/N	Proporción de muestras negativas que han sido correctamente clasificadas.
Precision (También llamado "Positive Predictive Value").	TP/(TP + FP)	Proporción de muestras clasificadas como positivas, que realmente son positivas.

F1 Score	2x Precision x Recall /(Precision + Recall)	Métrica que combina Precision y Recall.
Error Rate	(FP + FN)/(P + N)	Proporción de muestras mal clasificadas.

Figura 9.3 Algunas métricas de rendimiento para la evaluación de un clasificador.

9.2.4 Métricas de rendimiento

Con la información de la matriz de confusión se definen y emplean, diferentes métricas, en muchos casos dependiendo del campo de aplicación, para evaluar el rendimiento del clasificador. De todas ellas, la más popular ha sido la tasa de error global. Otras opciones son las tasas de error condicionadas a cada clase (conocidas en el campo del diagnóstico médico como sensibilidad y especificidad) o la utilización de tasas de riesgo. En el contexto de recuperación de información es popular la evaluación en términos de precisión y recuperación.

En las aplicaciones reales, no se consiguen clasificadores perfectos (aquel que tiene FN=0, FP=0, TN=N y TP=P). La figura 9.3 resume las métricas más empleadas para evaluarlo.

9.2.5 Técnicas de evaluación

A la hora de diseñar un clasificador disponemos de un conjunto discreto de muestras. Si utilizamos ese conjunto completo para entrenar el clasificador y posteriormente llevamos a cabo las pruebas con ese mismo conjunto de datos, tendremos una buena tasa de acierto ya que estamos probando con los mismos datos con los que hemos entrenado el clasificador. El principal problema será que habremos obtenido un modelo que probablemente no sea capaz de generalizar para datos nuevos, otros que no se hayan empleado en el entrenamiento, es decir, este modelo sólo clasificaría bien los datos que ya conoce. Este problema se conoce como sobreajuste (*overfitting*), figura 9.4.

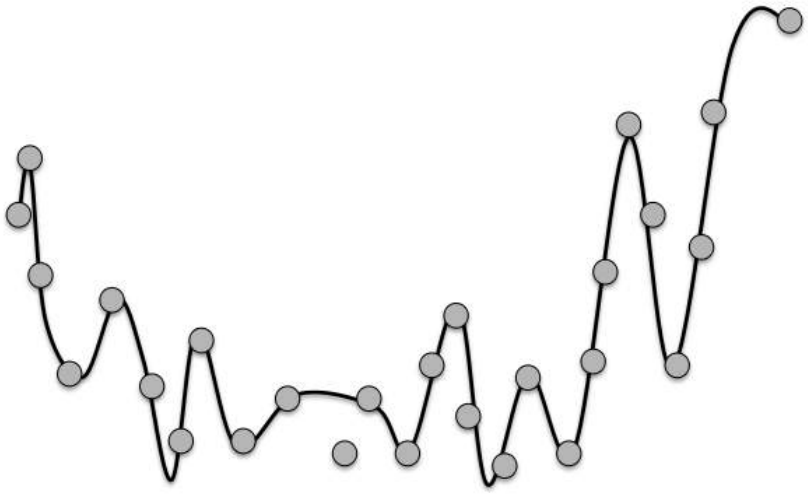


Figura 9.4 Ejemplo de *overfitting*. El modelo se ajusta a los datos pero no es capaz de generalizar para datos de entrada desconocidos.

Para que no se produzca este sobreajuste, podemos seguir diferentes estrategias: División del conjunto en dos particiones (*training* y *test*), validación cruzada con *k* particiones (*k*-fold cross validation) o Bootstrap.

9.2.5.1 División del conjunto en training y test

Cuando se divide el conjunto de datos en dos subconjuntos, uno de ellos (conjunto de training) se utiliza para entrenar, es decir, para construir el modelo, y el otro, el conjunto de prueba, para llevar a cabo la evaluación. Se suele optar por reservar 1/3 de los datos para hacer el test y los otros 2/3 para entrenar el clasificador, figura 9.5 .

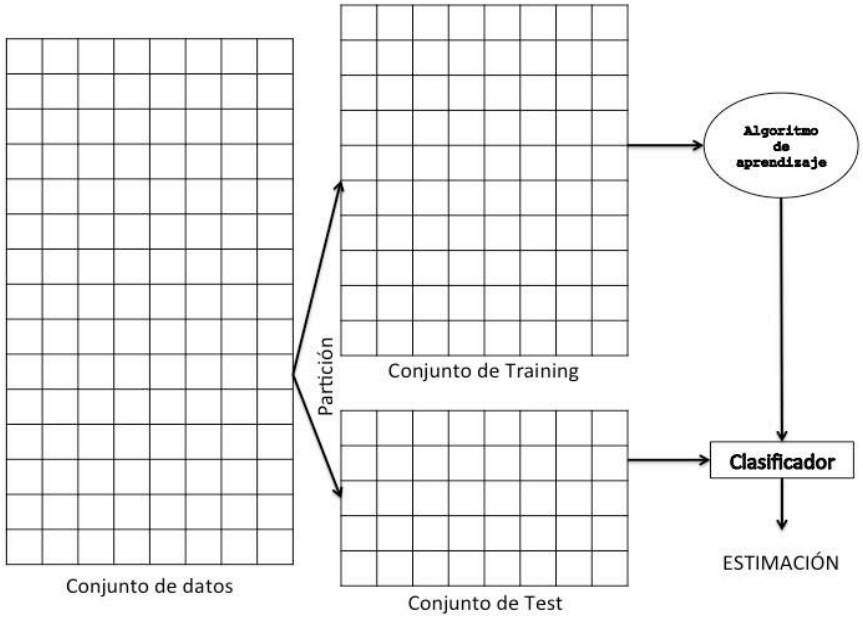


Figura 9.5 División del conjunto de datos en Training y Test.

9.2.5.2. Validación cruzada con *k* particiones

La validación cruzada con *k* particiones, llamada en inglés k-fold cross validation, es una alternativa al enfoque "División en conjuntos de *training* y *test*". El procedimiento es el siguiente: el conjunto de datos formado por N muestras, es dividido en *k* partes iguales. El valor de *k* suele ser un número pequeño, del orden de 5 o 10 y en el caso de que las partes no pudieran ser exactamente iguales, simplemente se dejaría alguna partición con un dato menos que el resto.

Una vez hechas las divisiones, se llevan a cabo *k* iteraciones. En cada una de ellas, una de las *k* partes en las que se ha dividido el conjunto de datos se usa como conjunto de test y las otras *k*-1 partes se usan como conjunto de training, figura 9.6.

Finalmente, los valores predichos para los datos de test (teniendo en cuenta las *k* iteraciones, será un total de N datos) se contrastan con las etiquetas reales de los mismos y se evalúa el modelo con las métricas seleccionadas.

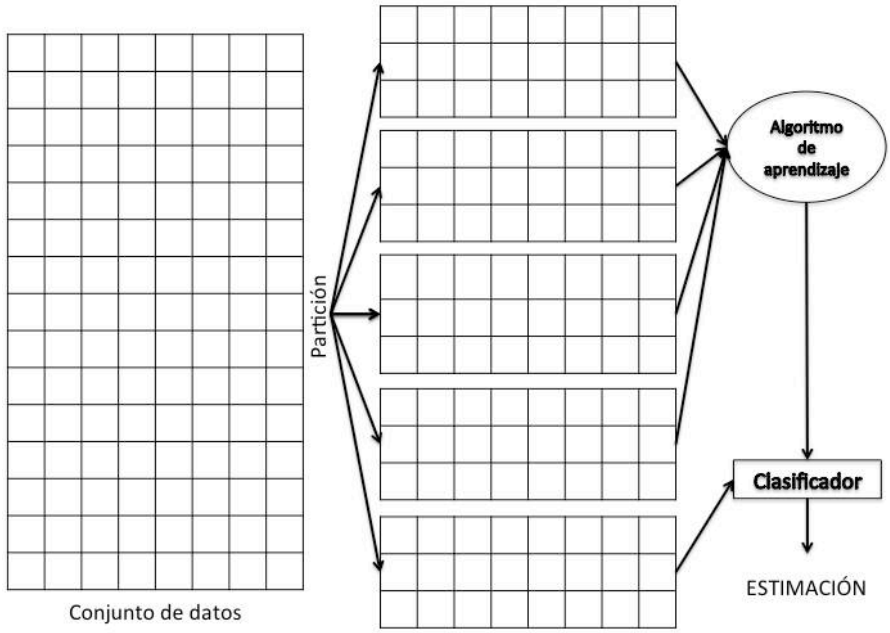


Figura 9.6. Validación cruzada k-Fold con k = 5. Ejemplo para una iteración cualquiera.

Existe un caso particular, en el que el valor del parámetro *k* coincide con el número de datos N. Se le conoce como "leave-one-out", ya que en cada una de las N iteraciones se utiliza un solo dato como conjunto de test y el resto, N-1, como conjunto de training.

9.2.5.3. Bootstrap

El conjunto de training en este caso se crea realizando N extracciones aleatorias con repetición sobre el conjunto de datos total, siendo N el número total de datos. De esta manera, tenemos un conjunto de training que tiene el mismo número de elementos que el conjunto de datos original

(algunos de ellos repetidos). El conjunto de test estará formado por todos aquellos datos que no hayan sido extraídos a la hora de crear el conjunto de training, figura 9.7.

El tamaño del conjunto de test será $0.368N$ que se obtiene simplemente calculando la probabilidad de que un ejemplo no salga en una extracción $(1-\frac{1}{N})$ y multiplicando este número por las veces que se realiza la extracción N.
Más formalmente tenemos:

$$\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^n=\lim_{n\to\infty}\left(1+\frac{1}{-n}\right)^n=\lim_{n\to\infty}\left(1+\frac{1}{-n}\right)^{n\frac{-1}{-1}}=\lim_{n\to\infty}\left(1+\frac{1}{-n}\right)^{-n^{-1}}=e^{-1}=\frac{1}{e}=0.368 \qquad (9.1)$$

Este proceso se repite un número prefijado de veces B y después se actúa como en el caso de validación cruzada, promediando las estimaciones de las métricas obtenidas para cada conjunto de test.

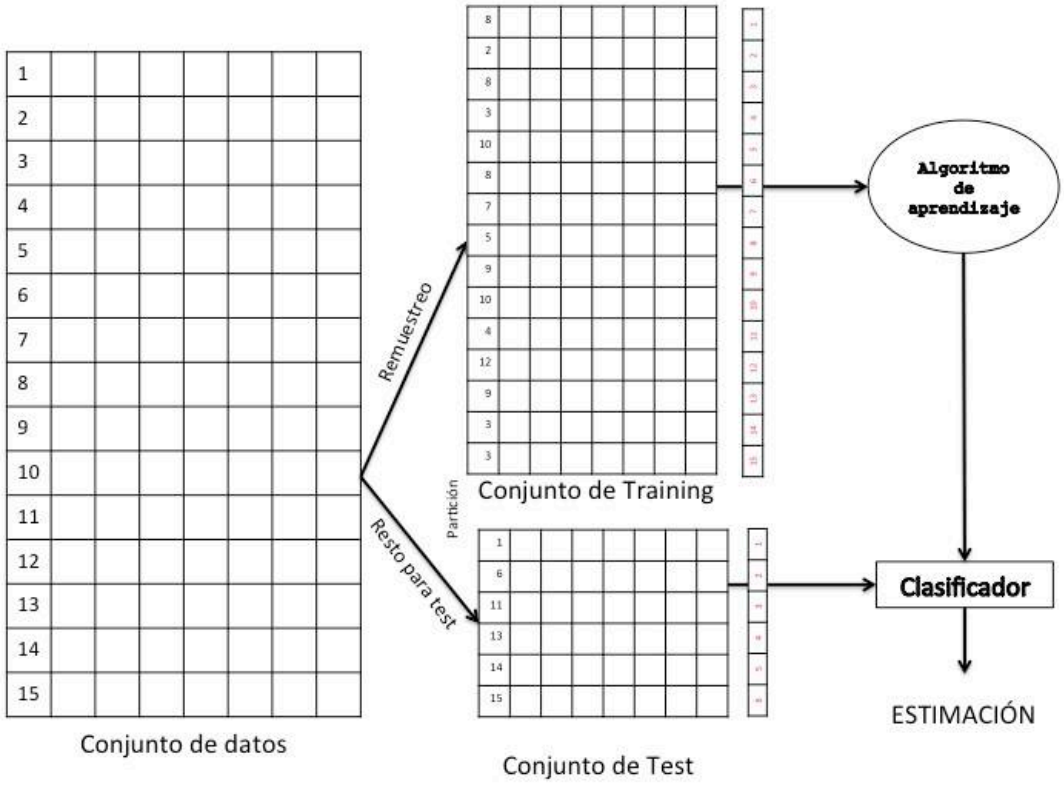


Figura 9.7 Bootstrap.

9.3 Ciclo de diseño de un clasificador

Cuando se afronta un problema de reconocimiento de patrones, hemos de considerar una serie de etapas para diseñar el clasificador.

En primer lugar, el conjunto de datos con el que se va a realizar el aprendizaje debe ser representativo de las condiciones en las que va a operar el clasificador. Si aún no disponemos de ese conjunto de datos, es necesario definir las características que puedan ser relevantes para el problema y diseñar un experimento para tomar las medidas necesarias. Si es posible, se incluirán también

características que pueden no parecer relevantes en esta etapa pero que, en combinación con otras, pudieran aportar información. Las limitaciones en la recolección de datos generalmente vienen dadas por la capacidad de financiación del proyecto, la disponibilidad de tiempo, la posibilidad de acceder a las muestras, etc.

En ocasiones, las medidas realizadas, como por ejemplo las imágenes tomadas por una cámara, deben ser procesadas para obtener, a partir de ellas, características relevantes para el sistema de reconocimiento de patrones. Esto puede incluir operaciones de filtrado, normalización, segmentación de imágenes, u obtención de descriptores de las imágenes que sean, por ejemplo, invariantes a la rotación, a la escala, etc.

Los descriptores o características (*features*) obtenidos normalmente no son todos igual de relevantes. Unos pueden serlo únicamente en relación con otros y otros pueden ser irrelevantes constituyendo sólo ruido para el sistema de reconocimiento en cuestión. De ahí que aplicar técnicas de selección de características o crear nuevas características a partir de las ya existentes, puede mejorar la calidad de la descripción.

La selección del modelo de clasificación, su entrenamiento (*training*) y la evaluación del mismo (*test*) constituye el núcleo central del desarrollo de un sistema de reconocimiento de patrones. Como se ilustra en la figura 9.8 con las líneas punteadas, el ciclo del diseño de un clasificador se puede cerrar en diferentes puntos. Podemos decidir usar el mismo modelo de clasificación con diferentes parámetros, cambiar el modelo de clasificación o bien, emplear diferentes técnicas de selección/extracción de características.

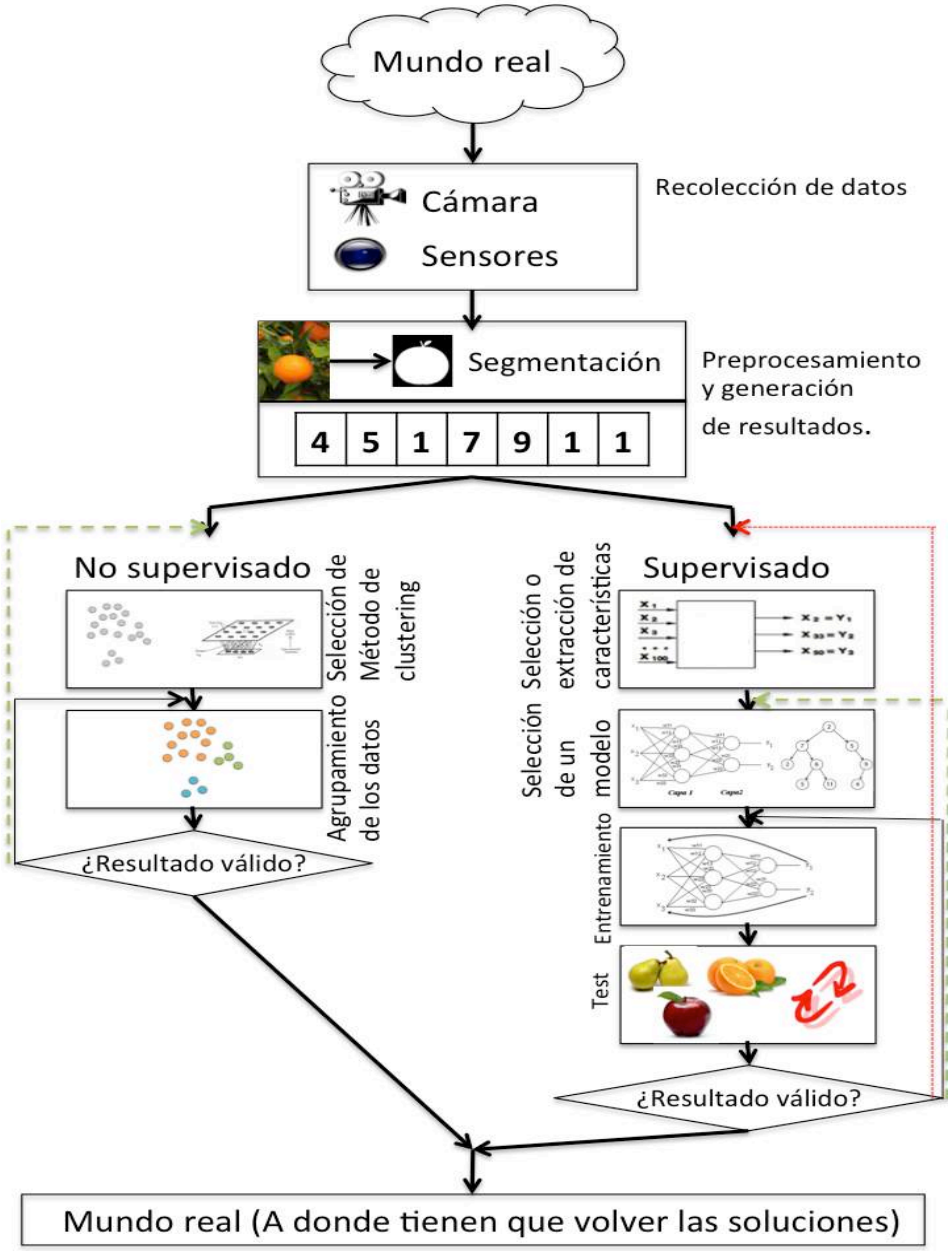


Figura 9.8 Esquema del ciclo de diseño de un clasificador.

9.4 Algoritmos de clasificación

Dado que el reconocimiento de patrones se enfrenta con problemas reales, se han desarrollado, y continúan desarrollándose, una plétora de algoritmos de aprendizaje, con sus respectivas limitaciones y fortalezas: Árboles de Decisión, Redes Neuronales, Máquinas de Vectores Soporte, Redes Bayesianas, etc. En esta sección presentamos algunos métodos básicos de clasificación, describiendo, a modo de ilustración: (a) una técnica sencilla de clasificación no supervisada (el algoritmo k-means), (b) un modelo estadístico simple de regresión lineal y (c) un modelo lineal de clasificación (regresión logística). Aspectos más avanzados, como el estudio profundo de las diferentes máquinas de clasificación, versiones avanzadas de las mismas y diferentes técnicas de optimización, quedan fuera

del alcance de este capítulo (para más información, Bishop, 2006; Duda y col. 2000; Mitchell, 1998; Pajares y de la Cruz, 2010).

9.4.1 El algoritmo K-means

K-means es un método de clasificación no supervisada muy popular para el análisis de agrupamientos o *clusters*. El término fue empleado por primera vez por MacQueen (1967) aunque el algoritmo fue propuesto por Lloyd (1957). El objetivo del algoritmo es organizar los datos en grupos (*clusters*), de tal manera que los miembros de cada grupo o *cluster* sean similares entre sí en cuanto a que sus características estén más próximas y diferentes a los de otro *cluster*.

El procedimiento consiste en dividir *N* observaciones en *k clusters*, de manera que cada observación pertenece al *cluster* más cercano. Esto da lugar a una partición del espacio de datos en celdas de Voronoi (Du y col. 2005).

Para este algoritmo, es el usuario el que decide a priori el número de grupos (*k*) en los que se quiere agrupar los datos. A continuación se explican los pasos a seguir con un ejemplo concreto (la figura 9.9 incluye un resumen del algoritmo) :

- Paso 1: Se inicializan aleatoriamente los *k* centros. En este caso, $k = 3$. Están representados en la figura 9.10 en color azul, rojo y verde.

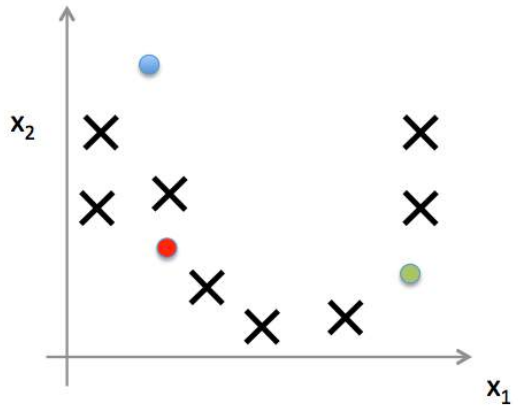


Figura 9.10 Paso 1: Inicialización aleatoria de los centros.

- Paso 2: Se asignan los datos a esos centros en función de las distancias, figura 9.11. El color de los datos representa el centro al cual ha sido asignado en función de su proximidad.

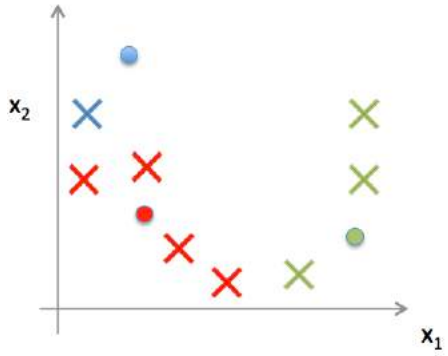


Figura 9.11 Paso 2: Asignación de puntos a centros en función de las distancias.

- Paso 3: Se reubica el centro teniendo en cuenta los puntos que se le han asignado, calculándolo como la media de todos los elementos asignados a su grupo, figura 9.12.

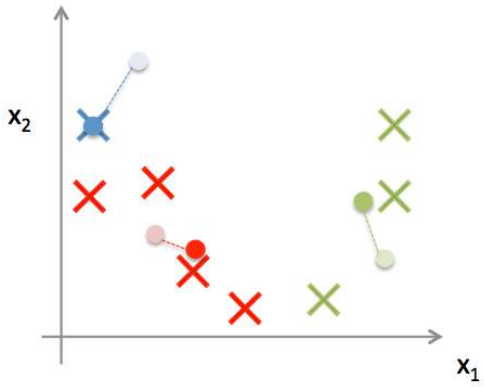


Figura 9.12 Paso 3: El centro se reubica pasando a ser el centroide de todos los puntos que tenía asignados.

- Paso 4: Se repite la asignación de puntos y la reubicación de los centros, es decir, los pasos 2 y 3 hasta que los centros no cambien de posición en dos iteraciones consecutivas.

Entrada: *k*, conjunto de N datos sin etiquetar: $\{x^1, x^2, ..., x^N\}$

1. Inicializar aleatoriamente los k centros dentro del espacio representado por los objetos.
2. Asignar cada muestra al centro más cercano.
3. Cuando todas las muestras hayan sido asignadas al centro correspondiente, recalcular las posiciones de los centros de manera que se conviertan en el centroide de las muestras que tienen asignadas dichos centros.
4. Repetir los pasos 2 y 3 hasta que los centros ya no se muevan. Esto produce una separación de los datos en *k* grupos de manera que muestras más parecidas entre sí pertenezcan al mismo grupo.

Figura 9.9 Algoritmo K-means.

Ejemplo 9.1. *Se quiere dividir un conjunto de individuos según su peso y altura en dos grupos para intentar determinar el sexo de cada uno de ellos (siendo la variable x_1 el peso y x_2 la altura del*

individuo). Utilizar KMeans con k=2 para determinar los clusters más apropiados para el siguiente conjunto de datos:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
x1	40	44	55	59	55	45	45	65	68	66	65	111	80	74	69	74	87	79	95	87	63	60	76	73	70
x2	150	148	154	162	165	148	163	160	171	179	170	195	180	172	179	181	173	168	180	192	165	158	181	177	155

Los dos centroides se han inicializado aleatoriamente a los siguientes valores:

	x1	x2
$C^0 1=$	40	150
$C^0 2=$	80	180

Solución:

Primera iteración

El primer paso consiste en calcular la distancia de cada uno de los elementos de nuestro conjunto de datos a los *clusters* para determinar a cuál de ellos pertenecen (al más cercano):

Centroide más cercano		Vector de datos x1	Vector de datos x2	Distancia euclidea a C1	Distancia euclidea a C2
1	1	**40**	**150**	0,0	50,0
1	2	44	148	4,5	48,2
1	3	55	154	15,5	36,1
1	4	59	162	22,5	27,7
1	5	55	165	21,2	29,2
1	6	45	148	5,4	47,4
1	7	45	163	13,9	38,9
2	8	65	160	26,9	25,0
2	9	68	171	35,0	15,0
2	10	66	179	38,9	14,0
2	11	65	170	32,0	18,0
2	12	111	195	84,1	34,4
2	13	**80**	**180**	50,0	0,0
2	14	74	172	40,5	10,0
2	15	69	179	41,0	11,0
2	16	74	181	46,0	6,1
2	17	87	173	52,3	9,9
2	18	79	168	43,0	12,0
2	19	95	180	62,6	15,0
2	20	87	192	63,0	13,9
2	21	63	165	27,5	22,7
1	22	60	158	21,5	29,7
2	23	76	181	47,5	4,1
2	24	73	177	42,6	7,6
2	25	70	155	30,4	26,9

A continuación, se actualizan los nuevos centroides teniendo en cuenta todos los elementos que han sido asignados a cada *cluster*. Así pues, los elementos que pertenecen al *cluster* 1 son:

	1	2	3	4	5	6	7	22
x1	**40**	44	55	59	55	45	45	60
x2	**150**	148	154	162	165	148	163	158

El nuevo centroide será la media de cada coordenada:

	x1	x2
$C^1 1=$	50,38	156

Los elementos que pertenecen al cluster 2 son:

	8	9	10	11	12	13	14	15	16	17	18	19	20	21	23	24	25
x1	65	68	66	65	111	**80**	74	69	74	87	79	95	87	63	76	73	70
x2	160	171	179	170	195	**180**	172	179	181	173	168	180	192	165	181	177	155

Por lo que el nuevo centroide $C^1 2$ será:

	x1	x2
$C^1 2=$	76,59	175,2

Como los centroides se han desplazado se continua con la siguiente iteración:

Segunda iteración

El primer paso es actualizar las distancias de cada elemento a cada centroide para determinar a que cluster pertenecen:

Centroide más cercano		Vector de datos x1	x2	Distancia euclidea a C1	Distancia euclidea a C2
1	1	**40**	**150**	12,0	44,4
1	2	44	148	10,2	42,4
1	3	55	154	5,0	30,2
1	4	59	162	10,5	22,0
1	5	55	165	10,1	23,9
1	6	45	148	9,6	41,7
1	7	45	163	8,8	33,9
1	8	65	160	15,2	19,1
2	9	68	171	23,1	9,5
2	10	66	179	27,8	11,3
2	11	65	170	20,2	12,7
2	12	111	195	72,1	39,7
2	13	**80**	**180**	38,1	5,9
2	14	74	172	28,5	4,1
2	15	69	179	29,6	8,5
2	16	74	181	34,4	6,4
2	17	87	173	40,4	10,6
2	18	79	168	31,0	7,6
2	19	95	180	50,7	19,0
2	20	87	192	51,4	19,8

1	21	63	165	15,5	17,0
1	22	60	158	9,8	23,9
2	23	76	181	35,8	5,9
2	24	73	177	30,9	4,0
1	25	70	155	19,7	21,2

A continuación, se actualizan los nuevos centroides teniendo en cuenta todos los elementos que han sido asignados a cada *cluster*. Así pues, los elementos que pertenecen al nuevo *cluster* 1 son:

	1	2	3	4	5	6	7	8	21	22	25
x1	**40**	44	55	59	55	45	45	65	63	60	70
x2	**150**	148	154	162	165	148	163	160	165	158	155

El nuevo centroide será la media de cada coordenada:

	x1	x2
$C^2 1=$	54,6	157,09

Los elementos que pertenecen al *cluster* 2 son:

	9	10	11	12	13	14	15	16	17	18	19	20	23	24
x1	68	66	65	111	80	74	69	74	87	79	95	87	76	73
x2	171	179	170	195	180	172	179	181	173	168	180	192	181	177

Por lo que el nuevo centroide $C^2 2$ será:

	x1	x2
$C^2 2=$	78,86	178,43

Como los centroides se han desplazado se continua con la tercera iteración:

Tercera Iteración

Al igual que en las iteraciones anteriores, el primer peso es actualizar las distancias y los miembros de cada cluster:

Centroide más cercano		Vector de datos x1	x2	Distancia euclidea a C1	Distancia euclidea a C2
1	1	**40**	**150**	16,3	48,1
1	2	44	148	14,0	46,3
1	3	55	154	3,1	34,1
1	4	59	162	6,6	25,8
1	5	55	165	7,9	27,4

1	6	45	148	13,2	45,5
1	7	45	163	11,3	37,2
1	8	65	160	10,8	23,1
2	9	68	171	19,3	13,2
2	10	66	179	24,7	12,9
2	11	65	170	16,6	16,2
2	12	111	195	67,9	36,2
2	13	**80**	**180**	34,2	1,9
2	14	74	172	24,4	8,1
2	15	69	179	26,2	9,9
2	16	74	181	30,8	5,5
2	17	87	173	36,1	9,8
2	18	79	168	26,7	10,4
2	19	95	180	46,4	16,2
2	20	87	192	47,6	15,8
1	21	63	165	11,5	20,8
1	22	60	158	5,4	27,8
2	23	76	181	32,1	3,8
2	24	73	177	27,1	6,0
1	25	70	155	15,5	25,0

Una vez hecho esto, se calculan los nuevos centroides. Para $C^3 1$:

	1	2	3	4	5	6	7	8	21	22	25
x1	40	44	55	59	55	45	45	65	63	60	70
x2	150	148	154	162	165	148	163	160	165	158	155

El nuevo centroide será la media de cada coordenada:

	x1	x2
$C^3 1$=	54,6	157,09

Para $C^3 2$:

	9	10	11	12	13	14	15	16	17	18	19	20	23	24
x1	68	66	65	111	**80**	74	69	74	87	79	95	87	76	73
x2	171	179	170	195	**180**	172	179	181	173	168	180	192	181	177

Por lo que el nuevo centroide $C^3 2$ será:

	x1	x2
$C^3 2$=	78,86	178,43

Como los centroides de la iteración 2 y 3 son iguales, el algoritmo se detiene y ya tenemos los clusters creados correctamente dividiendo el conjunto de muestras en función del sexo del individuo:

$C^2 1$=	54,64	157,09	$C^3 1$=	54,64	157,09
$C^2 2$=	78,86	178,43	$C^3 2$=	78,86	178,43

9.4.2 Regresión Lineal

Si disponemos de un conjunto de datos $S = \{(\boldsymbol{x}^k, y^k), k = 1,...,N\}$, donde tratamos de predecir un valor numérico continuo, $y^k \in R$, podemos emplear un modelo de regresión lineal. El objetivo que se persigue es encontrar la función h que mejor establezca la relación entre las variables independientes (descriptores) y la variable dependiente (etiqueta numérica). En el caso de la regresión lineal, la función h que buscamos se expresa como una combinación lineal de los descriptores con pesos θ y será de la forma:

$$h_\theta(\boldsymbol{x}) = \theta_0 + \theta_1 x_1 + ... + \theta_m x_m \tag{9.2}$$

El ajuste de los parámetros θ_i ha de hacerse con el conjunto de datos disponibles S, de manera que $h_\theta(\boldsymbol{x}^k)$ sea lo más próximo a y^k. Para calcular el error existente entre los datos reales y los datos estimados por nuestra función, definimos una función de coste, el error cuadrático medio:

$$J(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(h_\theta(\boldsymbol{x}) - y)^2 \tag{9.3}$$

Donde *N* se corresponde con el número de datos.

Debemos escoger los parámetros θ que minimizan la función de coste $J(\theta)$. Para ello, podemos comenzar inicializando aleatoriamente los parámetros y después, emplear un algoritmo de búsqueda, como puede ser el algoritmo de descenso de gradiente, que sucesivamente cambie el valor de θ de forma que el valor $J(\theta)$ sea cada vez menor y converja al valor de los parámetros que minimiza la función de coste.

El algoritmo de descenso por gradiente comienza un con valor arbitrario para los parámetros, que se actualizan de la forma

$$\theta_j = \theta_j - \alpha \frac{d}{d\theta_j} J(\theta) \tag{9.4}$$

donde α es la tasa de aprendizaje, que indica cuánto cambian los parámetros y el signo menos indica que nos movemos en la dirección opuesta al gradiente (ya que la dirección del gradiente es la de máximo incremento de la función). Si elegimos un valor de α muy pequeño, la función tardará mucho en converger. En cambio, si escogemos un valor muy grande puede oscilar y no llegar a converger en el mínimo global esperado.

Ilustraremos este proceso, con un problema de regresión con dos parámetros donde la función de regresión lineal toma la forma $h_\theta(x) = \theta_0 + \theta_1 x$.

La figura 9.16 muestra un ejemplo de la función de coste $J(\theta)$ donde podemos observar que es una función convexa con un mínimo global.

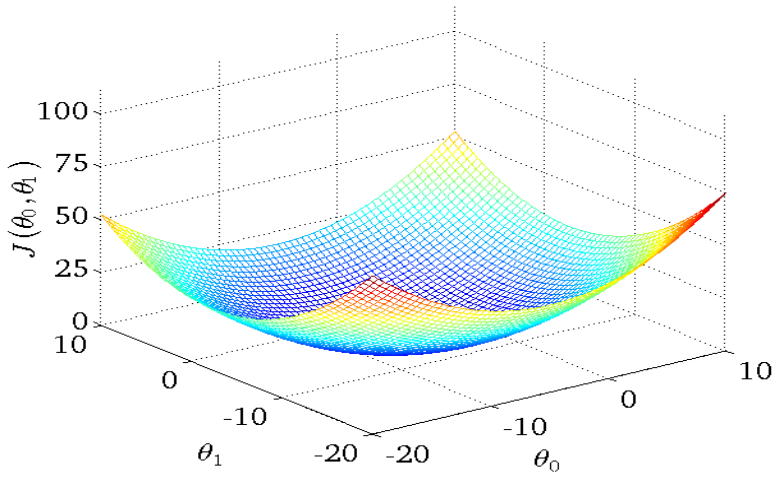


Figura 9.16 Representación de la función de coste para un problema de regresión lineal con dos parámetros.

Particularizando la ecuación 9.4 para un caso con dos parámetros, nos queda

$$\frac{d}{d\theta_j}J(\theta_0,\theta_1)=\frac{d}{d\theta_j}\frac{1}{2N}\sum_{i=1}^{N}(h_\theta(x)-y)^2=\frac{d}{d\theta_j}\frac{1}{2N}\sum_{i=1}^{N}(\theta_0+\theta_1x_1-y)^2 \tag{9.5}$$

Si derivamos ahora con relación a ambos parámetros, tenemos:
Para θ_0 :

$$\frac{d}{d\theta_0}J(\theta_0,\theta_1)=\frac{1}{N}\sum_{i=1}^{N}(h_\theta(x)-y) \tag{9.6}$$

Para θ_1 :

$$\frac{d}{d\theta_0}J(\theta_0,\theta_1)=\frac{1}{N}\sum_{i=1}^{N}(h_\theta(x)-y)x \tag{9.7}$$

Por tanto, el algoritmo de descenso de gradiente aplicado al problema de regresión lineal, se resume como:

$$\theta_0=\theta_0-\alpha\frac{1}{N}\sum_{i=1}^{N}(h_\theta(x)-y) \tag{9.8}$$
$$\theta_j=\theta_j-\alpha\frac{1}{N}\sum_{i=1}^{N}(h_\theta(x)-y)x$$

La figura 9.17 muestra cómo sería la actualización del parámetro θ_1 con el algoritmo de descenso de gradiente.

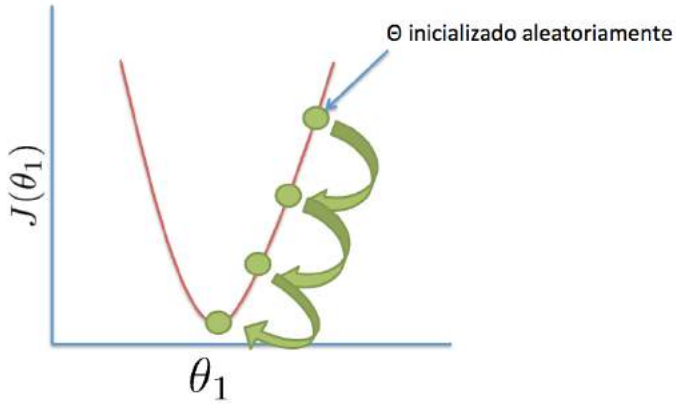


Figura 9.17 Descenso por gradiente.

9.4.3 Regresión Logística

Cualquier técnica de regresión puede adaptarse fácilmente para abordar problemas de clasificación supervisada (con un número discreto de clases). El truco consiste en llevar a cabo la regresión para cada clase asignando el valor uno para los ejemplos que pertenecen a la clase de interés y cero para el resto.

En este tipo de clasificación queremos que nuestra etiqueta de salida sea 0 o 1, por tanto, podemos acotar la función de regresión lineal entre esos valores. Para ello, vamos a emplear la función sigmoidal, que viene dada por:

$$g(z) = \frac{1}{1+e^{-z}} \tag{9.9}$$

y se representa de la forma mostrada en la figura 9.19:

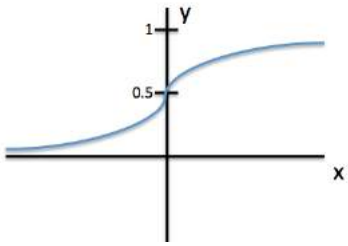


Figura 9.19 Ejemplo de funcionamiento del ajuste de los parámetros.

De esta manera, nuestra hipótesis quedaría acotada:

$$h_\theta = g(\theta_0 + \theta_1 x_1 + ... + \theta_m x_m) \tag{9.10}$$

Podemos decir que $h_\theta(\boldsymbol{x}^k)$ representa la probabilidad de que la salida sea uno, dado el dato $\boldsymbol{x}^k$, y parametrizado con θ.

No podemos utilizar la misma función de coste que hemos estado utilizando para los problemas de regresión, porque al haber acotado nuestra función, tendríamos una representación similar a la de la gráfica mostrada en la figura 9.20 izquierda.

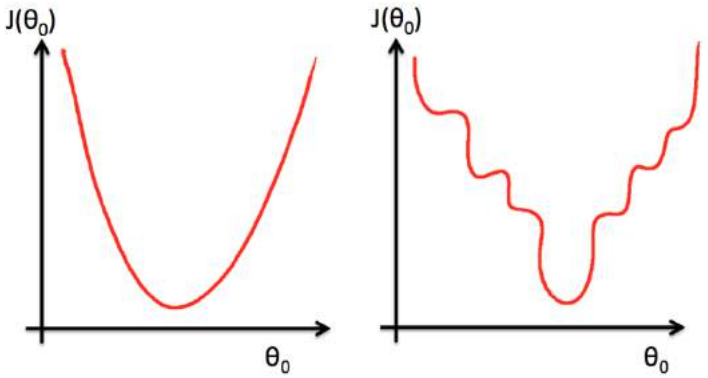


Figura 9.20 Diferencia entre la función coste de regresión lineal (izquierda) y regresión logística (derecha).

Si utilizáramos la misma función coste para ambos métodos, en el caso de la regresión logística, es probable que no consiguiéramos llegar a encontrar el mínimo global, por la existencia de muchos mínimos locales.

La función coste que emplearemos para regresión logística es la siguiente:

$$J(\theta) = -\frac{1}{N}[\sum_{i=1}^{N} d \log h_\theta(\boldsymbol{x}) + (1-d)\log(1-h_\theta(\boldsymbol{x}))] \tag{9.11}$$

La figura 9.21 muestra la contribución a la función de coste para: un ejemplo $\boldsymbol{x}^i$ que pertenece a la clase 1 (izquierda) y un ejemplo que pertenece a la clase 0 (derecha). Cuando la etiqueta es $d = 1$, el segundo sumando de la ecuación (9.11) queda anulado y si nuestra salida de la función se aproxima a uno (que es la etiqueta real), el error tiende a cero. En cambio, si se aproxima a cero, el error tiende a infinito.

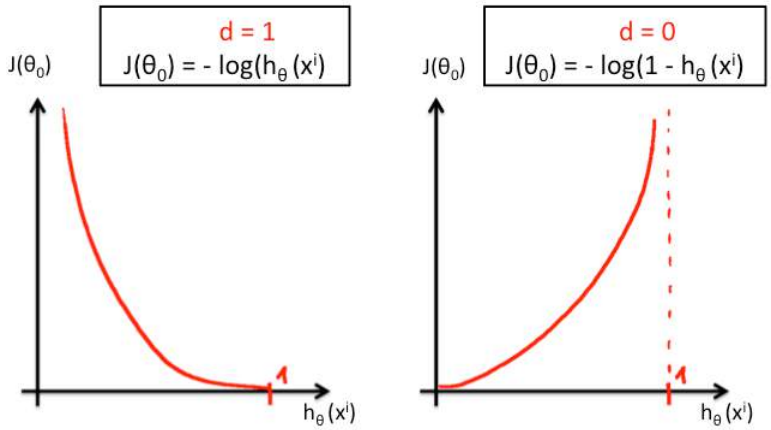


Figura 9.21 Función coste de la regresión logística.

Una vez ajustado el modelo de clasificación, éste nos devolverá, para los datos de *test*, un valor en el intervalo de cero a uno que indicará la probabilidad de pertenencia a la clase uno. Se aplicará, finalmente, un umbral para tomar la decisión final de pertenencia a la clase.

9.5 Bibliografía

Duda, R. O.; Hart, P.E. (1973) Pattern classification and scene analysis. John Wiley & Sons.

Fukunaga, K. (1972) Introduction to Statistical Pattern Recognition. Academic Press, Inc.

Bishop, C.M. (2006) Pattern Recognition and Machine Learning. Springer.

Duda, R. O.; Hart, P. E.; Stork, D. G. (2000) Pattern Classification, John Wiley & Sons.

Mitchell, T. (1998) Machine Learning, MIT Press.

Pajares, G. y de la Cruz, J.M. (Eds.) (2010). Aprendizaje Automático: un enfoque práctico. RA-MA, Madrid.

Du, Qiang; Wang, Desheng (2005), The Optimal Centroidal Voronoi Tessellations and the Gersho's Conjecture in the Three-Dimensional Space, *Computers and Mathematics with Applications (49)*: 1355–1373

Fix, E.; J.L. Hodges (1989) An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation: Commentary on Fix and Hodges. *International Statistical Review / Revue Internationale de Statistique 57* (3): 233–238

Viñuela Isasi, P.; Galván León, I.M. (2003). Redes de neuronas artificiales. Un enfoque práctico. Prentice Hall.

Lloyd, S. P. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory 28* 129–137.

MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability 1*. University of California Press. pp. 281–297.

Ng, A., Curso de "Machine Learning". Coursera. <https://www.coursera.org/learn/machine-learning/> [Consulta: abril 2015]

Yan. X; Gang Su. X. (2009). Linear Regression Analysis. *Theory and Computing.* World Scientific Pub Co.

CAPÍTULO 10

CLASIFICACIÓN DE IMÁGENES CON BAG OF VISUAL WORDS

Víctor GONZÁLEZ-CASTRO[1], Enrique ALEGRE[2], Eduardo FIDALGO[2]

[1] Department of Neuroimaging Sciences, Centre for Clinical Brain Sciences, University of Edinburgh, Edinburgh, United Kingdom
[2] Departamento de Ingeniería Eléctrica y de Sistemas y Automática, Universidad de León, León, España

La clasificación de imágenes es un proceso mediante el cual un ordenador es capaz de decidir qué contenidos están presentes en una imagen, esto es a qué clase pertenece o qué objetos contiene. En los últimos años el modelo Bag of Visual Words (BoVW) se ha convertido en una de las soluciones más utilizadas para realizar esta tarea. El término *visual word* (palabra visual, o simplemente "palabra") hace referencia a una pequeña parte de una imagen. El BoVW consta de varias etapas: un muestreo de puntos característicos (*keypoints*) de la imagen, la descripción de los mismos, la creación de un diccionario de palabras visuales mediante un proceso de agrupamiento, la representación de las imágenes a nivel global utilizando este diccionario y, finalmente, una clasificación de estas representaciones para decidir la clase a la que pertenece. En este capítulo se explicará el modelo BoVW de clasificación de imágenes, detallando estas etapas.

10.1. Introducción

La clasificación de imágenes consiste en determinar automáticamente los contenidos que aparecen en una imagen o, dicho de otro modo, asignar una o varias etiquetas a una imagen de entrada de entre un conjunto fijo de categorías. Este es uno de los problemas principales en Visión por Computador y, de hecho, muchas tareas de esta disciplina (segmentación, detección de objetos, detección de rostros) se pueden reducir a un problema de clasificación de imágenes. Por ejemplo, supóngase un sistema de clasificación de imágenes en 4 clases, *{gato, perro, cobaya, hamster}*, figura 10.1. Hay que tener en cuenta que, para un ordenador, una imagen se representa como una matriz numérica de tres

dimensiones (en el caso de imágenes en color) o una (en caso de imágenes en escala de grises), con valores que representan la intensidad, mayor o menor, de color o nivel de gris. La tarea del sistema de clasificación es obtener a partir de este gran conjunto de datos numéricos una etiqueta de categoría.

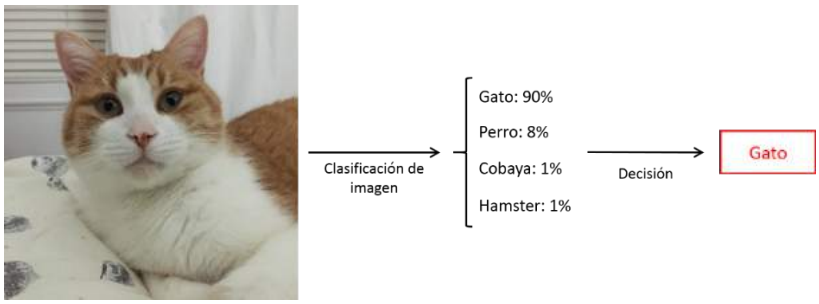
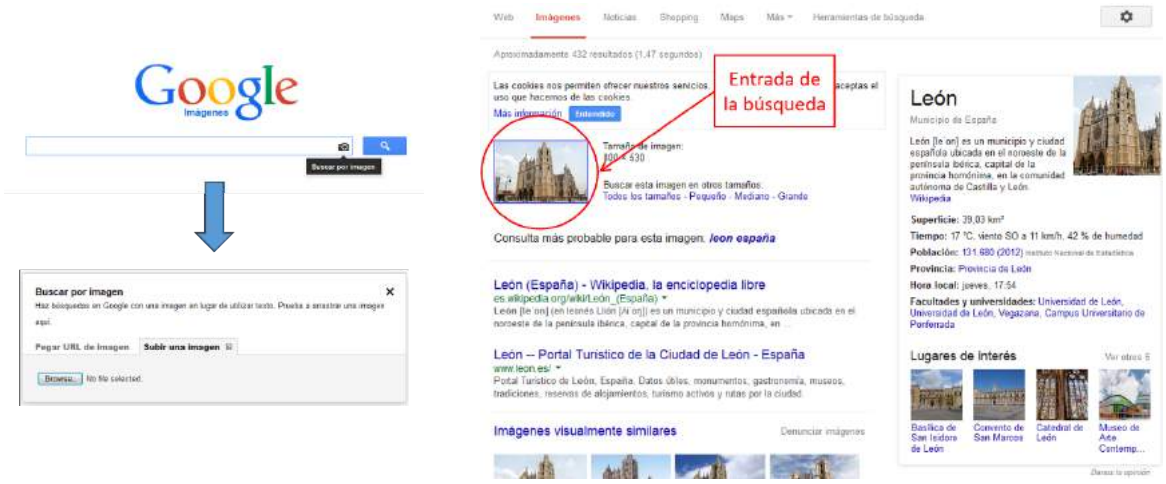


Figura 10.1. Ejemplo de sistema de clasificación de imágenes en las categorías *{gato, perro, cobaya, hamster}*.

La clasificación de imágenes se encuentra presente en muchas aplicaciones que utilizamos en la vida diaria. Facebook ha implementado detectores de personas y rostros para sus aplicaciones, y Google ha desarrollado clasificadores de imágenes más sofisticados para sus Google Glasses, o para su buscador, cuando se selecciona la opción de búsqueda por imagen, figura 10.2. La clasificación de imágenes también está presente en otras aplicaciones en la industria como son la teledetección (Sheng y col., 2010), la seguridad (Jing y col., 2013), la clasificación de documentos (Augereau y col., 2014), etc.



(a) *(b)*

Figura 10.2. Aplicación "buscar por imagen" de google imágenes (*a*) Interfaz para subir una imagen. (*b*) Resultados de la búsqueda.

A pesar de que reconocer una imagen es trivial para un ser humano, hay que considerar los retos que presenta para un sistema automático de Visión por Computador para el que, recordemos, una imagen no es más que un conjunto de números. Esa dificultad se basa principalmente en la presencia de uno o varios de los siguientes aspectos:

- **Cambio en el punto de vista:** Un mismo objeto puede estar orientado de muchas maneras diferentes respecto a la cámara.
- **Variación en la escala:** Los objetos de una misma clase a menudo presentan variaciones en su tamaño (por ejemplo, la altura de una persona puede ser muy variable).
- **Cambios en la pose:** Muchos objetos no son rígidos, y puede aparecer en posiciones diferentes.
- **Oclusiones:** El objeto de interés puede estar parcialmente oculto tras otro objeto en una imagen.
- **Condiciones en la iluminación:** Los diferencias de la iluminación sobre una imagen pueden ocasionar grandes cambios de intensidad a nivel de píxel.
- **Variación intra-clase:** Las clases *objetivo* pueden contener objetos diferentes entre sí. Por ejemplo, la clase *silla* puede contener sillas muy diversas,
- **Similaridad inter-clase:** Clases *objetivo* diferentes pueden tener ciertos parecidos entre sí que "confundan" al sistema de clasificación. Por ejemplo, un *lobo* puede ser similar a ciertos *perros*.

10.1.1. Modelo Bag of Visual Words

Bag of Visual Words (BoVW) (Szelinski, 2011), basado en el modelo Bag of Words utilizado en procesamiento del lenguaje natural, es un modelo de clasificación de imágenes propuesto inicialmente por Sivic y Zisserman (Sivic y Zisserman, 2003), que representa una imagen en función de la frecuencia de una serie de elementos visuales. Se esperaría que, por ejemplo, una imagen de un edificio contenga más ventanas que una imagen de un árbol. Estos elementos visuales se llamarán palabras visuales. El conjunto de palabras visuales constituye el diccionario.

Un sistema de clasificación de imágenes que utilice BoVW contiene los siguientes elementos:

1. **Conjunto de imágenes o *dataset***. Para crear un modelo de BoVW es necesario disponer de conjuntos de imágenes, etiquetados y normalmente grandes, también llamados *datasets*. Están formados por cientos, miles o incluso millones de imágenes de las que normalmente se conoce su contenido. Un ejemplo de *dataset* ampliamente utilizado en la investigación es *Flowers* (Nilsback y Zisserman, 2008), formado por 1360 imágenes que contienen 17 tipos de flores, estando cada tipo representado por 80 imágenes.

2. **Muestreo de puntos característicos o *keypoints*:** En cada imagen del *dataset*, es necesario seleccionar una serie de características que permitan representarla. Para ello es necesario seleccionar las posiciones en las que se extraerán dichas características. Esos puntos clave se suelen obtener principalmente mediante el cálculo de puntos característicos (*keypoints*), un muestreo denso o un muestreo aleatorio.

3. **Descripción de regiones alrededor de puntos característicos:** Los puntos clave seleccionados se caracterizan mediante algún descriptor que típicamente recoge la información de cómo se distribuyen los niveles de grises de los píxeles que le rodean. Algunos ejemplos de descriptores típicamente utilizados son SIFT (Lowe, 2004), SURF (Bay y otros, 2008), HoG (Dalal y Triggs, 2005), etc.

4. **Creación de un conjunto de entrenamiento y pruebas:** Es necesario dividir el *dataset* en dos conjuntos disjuntos, uno de entrenamiento y otro de pruebas o *tests*. Habitualmente el conjunto de entrenamiento suele ser mayor al de test, con objeto de dar al sistema más robustez, pero no es imprescindible. Para el anterior *dataset* (Flowers) es posible hacer un conjunto de entrenamiento con el 75% de las imágenes (1020) y un conjunto de pruebas con el 25% restante (340). Ninguna imagen del conjunto de pruebas ha de estar incluida en el conjunto de entrenamiento y viceversa.

5. **Formación del diccionario:** Los descriptores extraídos en el punto 3 se usan para construir un diccionario, o conjunto de palabras que representen a los puntos clave del punto 2. Estos conjuntos de palabras representativas se obtienen agrupando los vectores de características obtenidos para cada punto clave mediante alguna técnica de *clustering* (K Means, Gaussian Mixture Model, etc.). Para la construcción del diccionario se usan única y exclusivamente los descriptores pertenecientes a las imágenes del conjunto de entrenamiento.

6. **Representación de imágenes:** Una vez que el diccionario está construido, cada imagen del *dataset* se representa mediante un histograma que contiene con qué frecuencia aparece en esa imagen cada una de las palabras visuales del diccionario. Para ello se mide la distancia de cada descriptor extraído de una imagen con todas las palabras visuales del diccionario: dicho descriptor quedará representado por la palabra visual cuya distancia sea menor. Repetido este proceso para todos los descriptores, cada imagen del *dataset* quedará representada por un vector cuyo tamaño será el mismo que el número de palabras visuales del diccionario. Cada elemento del vector indicará la frecuencia de esa palabra dentro de la imagen correspondiente.

7. **Aprendizaje y reconocimiento:** Usando las imágenes del conjunto de entrenamiento, se creará un modelo usando las imágenes codificadas con la frecuencia de cada palabra visual. La precisión de este clasificador se evalúa típicamente utilizando las imágenes del conjunto de pruebas.

En las figuras siguientes se ilustra este muestreo y formación del diccionario (figura 10.3), representación (figura 10.4) y el reconocimiento de imágenes (figura 10.5), respectivamente.

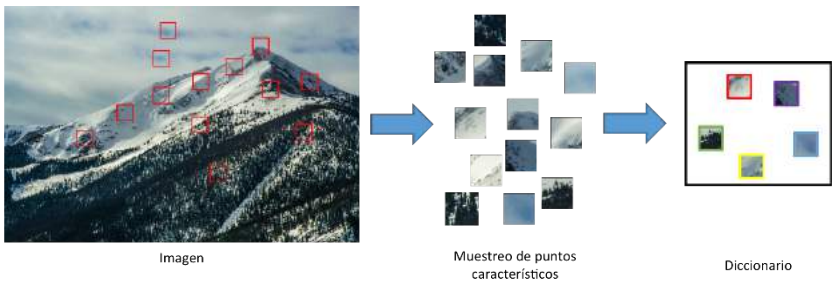


Figura 10.3. Ilustración del muestreo de características y formación del diccionario.

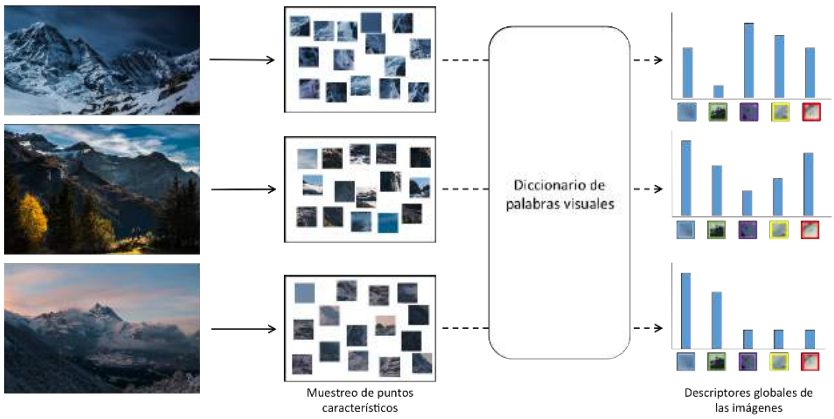


Figura 10.4. Diagrama de la etapa de representación de imágenes.

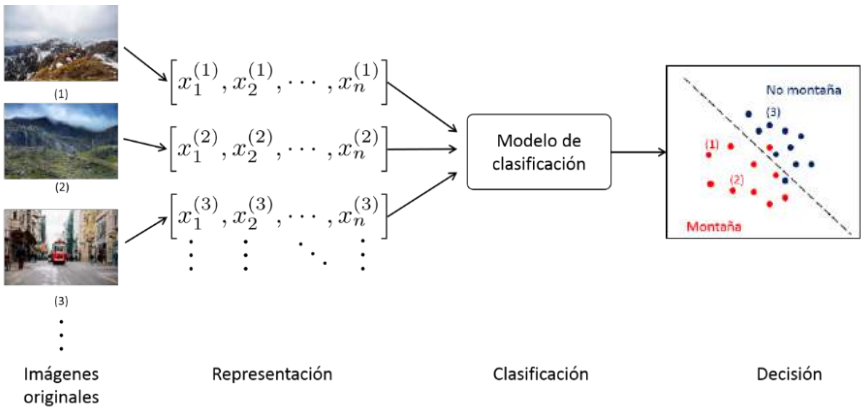


Figura 10.5. Diagrama de la etapa de reconocimiento de imágenes.

10.2. Descripción de imágenes utilizando SIFT

SIFT (Scale Invariant Feature Transform) es un método propuesto por Lowe (Lowe, 1999; Lowe, 2004) que permite obtener puntos característicos de la imagen y que posteriormente describe utilizando

un conjunto de histogramas de gradiente locales obtenidos a partir de la región que rodea a cada dicho punto característico. Los métodos que como SIFT utilizan este enfoque, obtener puntos invariantes y describir la región que los rodea, son muy robustos ante solapamientos parciales del objeto, variaciones de escala, translaciones y cambios en la iluminación. Por ese motivo estos métodos han sido muy utilizados en los últimos años en un gran número de aplicaciones que requieren describir y reconocer objetos o incluso clasificar imágenes, como es el problema que nos ocupa en este capítulo. Si bien, desde que SIFT fue publicado por primera vez (Lowe, 1999) han surgido muchos otros métodos similares que utilizan la misma estrategia basada en obtener características locales invariantes de la imagen, SIFT sigue siendo a día de hoy el referente y también, muy probablemente, el más utilizado.

El método propuesto por Lowe (Lowe, 1999) consta de dos partes claramente diferenciadas: la obtención de *puntos característicos* y la *descripción* de la región alrededor de cada punto de interés. Como se ha comentado anteriormente los dos primeros pasos en la creación de un modelo basado en Bag of Visual Words son el muestreo de puntos característicos y la descripción de cada uno de estos puntos. Si bien la obtención de la posición de los puntos puede realizarse de diferentes formas, en el contexto de la clasificación de imágenes, se asume que el *muestreo denso* es el procedimiento a seguir. Por ello, en esta sección explicaremos el método propuesto por Lowe (Lowe, 1999) para la obtención de puntos característicos, posteriormente indicaremos cómo se realiza un muestreo denso y finalmente explicaremos cómo se obtiene un descriptor para una ubicación determinada, siguiendo el método propuesto por Lowe (Lowe, 1999; Lowe, 2004).

10.2.1. Obtención de puntos característicos

Uno de los motivos por los que SIFT es muy utilizado es porque llevó un paso más allá la detección y localización de puntos característicos en el espacio y en la escala. Si bien el concepto de pirámides de paso banda de diferencia de Gaussianas había sido propuesto por Burt (Burt y Adelson, 1983) y por Crowley (Crowley y Stern, 1984), Harris (Harris y Stephens, 1988) ya había realizado su propuesta para detectar esquinas en base a las relaciones entre la traza y el determinante del Hessiano, y Lindeberg (Lindeberg, 1994) ya había demostrado que el único kernel posible para el espacio-escala es la función gaussiana, Lowe combinó todas estas ideas para proponer una nueva forma de obtener puntos característicos invariantes a la escala.

Los pasos necesarios a seguir para calcular dichos puntos son los siguientes:

1. **Detección de extremos** en el espacio-escala

   Se buscan puntos de interés en toda la imagen y en todas las escalas consideradas utilizando un espacio-escala formado por diferencias de Gaussianas.

2. **Localización precisa** de puntos característicos

Se localizan de forma más precisa los extremos anteriores, ajustando al punto obtenido una función cuadrática 3D. Posteriormente, se eliminan *keypoints* que están próximos a los bordes o tienen bajo contraste.

3. Asignación de la **orientación**

   A cada punto característico se le asigna una o varias *orientaciones* en función de las *direcciones del gradiente local*. Esta orientación, conjuntamente con la ubicación y la escala calculadas anteriormente, permiten que el descriptor sea invariante a estas tres transformaciones.

10.2.1.1 El espacio escala

Para detectar puntos característicos tanto en el espacio como en la escala, Lowe propuso crear un espacio escala mediante diferencia de Gaussianas. En contraste con otros métodos como el de Lindeberg (Lindeberg, 1994) que utilizaba el Laplaciano de la Gaussiana normalizado a la escala, o el mismo Harris (Harris y Stephens, 1988), que utilizaba un criterio basado en la relación entre el determinante y la traza del Hessiano, Lowe propone utilizar DoG, diferencia de Gaussianas, para ubicar puntos característicos, tanto en el espacio como en la escala. Aunque DoG no sea el mejor detector desde el punto de vista de la repetibilidad, como probó Mikolajczyk (Mikolajczyk y Schmid, 2002) sí es muy eficiente, más que el LoG, es una buena aproximación al Laplaciano de Gaussiana normalizado a la escala y es muy estable en los puntos detectados.

Por ello, Lowe (Lowe, 1999) propuso crear el espacio escala para SIFT mediante 4 octavas y 5 escalas cada octava con las siguientes características:

- Cada octava está formada por 5 imágenes, cada una procedente de una Gaussiana con una σ mayor que la anterior al ser multiplicada por un factor k. Se dice que cada una de esas imágenes se encuentra a una escala diferente, determinada por el valor de su σ.
- Cada octava se diferencia de la anterior en el tamaño de la imagen ya que la segunda octava se obtiene reduciendo el tamaño de las imágenes en la octava anterior por un factor de dos. En concreto, se obtiene eliminando una de cada dos filas y columnas. De igual manera, para obtener la tercera octava se realiza el mismo procedimiento para reducir el tamaño de las imágenes de la segunda. Y así sucesivamente.

10.2.1.2 Localización precisa de los puntos de interés

En el artículo publicado por Lowe (Lowe, 2004) se añadió al método presentado originalmente cinco años antes una propuesta que permite ajustar una función cuadrática 3D a los puntos de una muestra local, de manera que se puede determinar la localización interpolada del máximo. Esta

propuesta, basada en el método publicado por Brown (Brown y Lowe, 2002) proporciona a SIFT una mejora en la búsqueda de correspondencias y en la estabilidad al eliminar puntos de bajo contraste.

En ocasiones alguno de los puntos extremos detectados cae a lo largo de un borde lo que hace que sean puntos poco deseados al ser menos estables. Para eliminarlos, SIFT utiliza la relación entre los valores propios del Hessiano y, para evitar calcular explícitamente sus valores, toma prestada de Harris (Harris y Stephens, 1988) la aproximación que permite obtener la relación entre dichos valores propios a partir del cálculo de la traza y del determinante del Hessiano. De esa manera, Lowe propone evaluar la siguiente expresión:

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}, \tag{10.1}$$

donde r es la relación entre los dos valores propios y que en SIFT se fija como $r = 10$. Aquellos puntos clave con una relación entre las curvaturas principales mayor de 10, son eliminados.

10.2.1.3 Asignación de la orientación

Para conseguir que el descriptor, que se calculará posteriormente alrededor del punto característico, sea invariante a la rotación, es necesario en primer lugar asignar una orientación principal a cada *keypoint* y después tenerla en cuenta en el cálculo del descriptor.

Con el fin de que el cálculo de la orientación sea invariante a la escala, se selecciona la imagen Gaussiana suavizada que tiene la escala más próxima a la del punto característico que se está estudiando. Para cada píxel de dicha imagen, se calcula su magnitud, $m(x, y)$, y orientación, $\theta(x, y)$, utilizando diferencias entre los píxeles vecinos, de la siguiente manera:

$$\begin{aligned} m(x,y) &= \sqrt{(L(x+1,y)-L(x-1,y))^2 + ((L(x,y+1)-(L(x,y-1))^2}, \\ \theta(x,y) &= \tan^{-1}\frac{L(x,y+1)-L(x,y-1))}{L(x+1,y)-L(x-1,y))} \end{aligned} \tag{10.2}$$

Tomando la región que rodea al punto característico, se calcula un histograma de 36 elementos, donde cada elemento cubre un sector circular, consecutivo al anterior, de 10º. En cada una de las 36 agrupaciones del histograma se añade la magnitud del gradiente correspondiente a esa orientación previamente ponderado por una gaussiana circular, centrada en el punto característico, y cuya σ es 1,5 veces el tamaño de la escala a la que se localizó dicho punto.

Un detalle adicional del método es que a cada punto de interés se le asignan una o varias orientaciones, de manera que aproximadamente un 15% de los puntos obtenidos tendrán múltiples orientaciones. Estos puntos, con más de una orientación asignada, contribuyen muy positivamente a que la búsqueda de correspondencias sea estable. Para realizar esta asignación, sobre el histograma anterior se detecta el pico más alto y todos aquellos que tengan una magnitud superior al 80% de dicho pico. Se crea un punto característico para cada uno de los picos anteriores, con la orientación indicada por su posición en el histograma.

El último paso consiste en ajustar una parábola a los 3 valores del histograma que están más próximos a cada pico elegido de manera que se interpola la posición del pico de una forma más precisa.

10.2.2. Muestreo denso de puntos de interés

Como se ha comentado al principio de esta sección, el muestreo denso es el método normalmente empleado cuando los descriptores obtenidos con SIFT se utilizan para clasificar imágenes. Este muestreo se caracteriza por que no se buscan puntos de interés sino que se calcula un descriptor SIFT, en una ventana de tamaño *s*, para cada *n* píxeles de la imagen. Si se siguiera la propuesta de Lowe para SIFT, el tamaño de la ventana, *s*, sería un entorno de 16x16 píxeles alrededor del punto de interés.

La idea subyacente en el muestreo denso es calcular un descriptor SIFT para cada *n* píxeles de la imagen donde, si *n* fuera igual a 1, se estaría calculando dicho descriptor para cada uno de los píxeles de la imagen. Dependiendo de la aplicación, este *n*, normalmente llamado el *step*, suele tener valores que van de 3 a 7, de forma que se calcularía el descriptor SIFT cada 3 o cada 7 píxeles de la imagen. Los valores del *step* suelen mantenerse bajos.

Los detalles de cómo se realiza este muestreo denso dependen de la implementación que se utilice. La librería VLFeat, (Vedaldi y Fulkerson, 2008) ofrece una función, vl_dsift que permite realizar este cálculo y de la que comentamos a continuación alguna de sus características, para ilustrar mejor alguna de las posibilidades que suelen considerarse cuando se muestrea una imagen de esta manera.

En la implementación de VLFeat, para obtener el muestreo denso, no se calcula un espacio escala gaussiano de la imagen. Esto es así porque se asume que dicho cálculo es necesario para la detección de puntos de interés y el muestreo denso se basa en elegir un gran número de puntos igualmente espaciados, que cubran toda la imagen, y que permitan describir su contenido. De todas formas, asumiendo que este muestreo denso puede realizarse a diferentes escalas, vl_dsift sugiere que se suavice previamente la imagen sobre la que se va a realizar el muestreo, al nivel deseado de escala, utilizando otra de las funciones de la librería.

La función vl_dsift llama *step* a la distancia en píxeles entre cada descriptor extraído, siendo su valor por defecto 1, y *size* al tamaño de la ventana sobre la que se calcula el descriptor. Otras opciones de esta función es que permite indicar la región sobre la que se calcula el muestreo mediante la opción *bounds,* o que se puede elegir entre utilizar una función Gaussiana sobre la ventana de cálculo, al igual que SIFT, o hacerlo de forma rápida, mediante la opción *fast,* y realizar el cálculo de forma plana, elemento a elemento, sin realizar la ponderación.

Como apunte final, indicar que cuando se utiliza muestreo denso, en contraste con lo que se hace en SIFT basado en puntos característicos, suele recomendarse suavizar poco la imagen original, para potenciar que los gradientes aparezcan resaltados.

10.2.3. Cálculo del descriptor SIFT

Una vez finalizado el proceso de detección de puntos característicos, o bien seleccionadas las posiciones de la imagen alrededor de las cuales se desea obtener un descriptor, se procede al cálculo del vector de características, típicamente de 128 elementos, basado en la concatenación de histogramas orientados de gradientes. Los puntos de interés son en realidad una tupla de cuatro elementos, *(x, y, σ, θ)*, donde (x,y) corresponde con las coordenadas del punto, σ con la escala a la que se ha obtenido y θ la orientación principal que tiene dicho punto, calculada según se explicó en la sección 10.2.1.3 Asignación de la orientación.

De forma resumida el cálculo del descriptor, para cada punto, consta de los siguientes pasos:

4. Se obtiene la *magnitud y la orientación del gradiente* en una ventana de, típicamente, 16x16 píxeles centrada en el punto característico.

5. Se calcula un *histograma de orientaciones del gradiente de 8 bins para cada una de las 16 regiones* que se obtienen al dividir la ventana inicial en regiones menores de 4x4 píxeles. Cada *bin* corresponde con uno de los 8 sectores circulares de 45 grados en los que se divide la circunferencia completa. El histograma contendrá las magnitudes acumuladas de los píxeles cuya orientación se encuentre dentro del sector asignado a cada *bin*.

6. Se *normaliza el vector de 128 elementos* que se obtiene al concatenar los histogramas de 8 valores para cada uno de las 16 divisiones que se han realizado de la región. Se realiza una doble normalización a longitud unidad.

A continuación se explican algunos detalles adicionales de cada uno de estos pasos.

10.2.3.1 Obtención de la magnitud y orientación del gradiente

Dada una ventana de 16 x 16 píxeles, alrededor del punto característico se obtiene la orientación del gradiente, para cada píxel, teniendo en cuenta los siguientes detalles:

- Se obtiene la magnitud y la orientación del gradiente para cada píxel, utilizando la información de sus vecinos horizontal y vertical. Se sigue el mismo procedimiento de cálculo que se realizó para obtener los gradientes en la etapa de cálculo de la orientación del punto característico según se explicó en la sección 10.2.1.3 Asignación de la orientación.
- El cálculo se realiza sobre la imagen del espacio escala con el nivel de suavizado Gaussiano más próximo a la escala del punto.

- Se realiza con invarianza a la orientación, rotando las coordenadas del descriptor y las orientaciones del gradiente en función de la orientación del punto característico.
- Se ponderan todos los valores de la magnitud del gradiente de la ventana a calcular por una gaussiana con una σ igual a la mitad del ancho de la ventana del descriptor.

10.2.3.2 Histograma de orientaciones del gradiente

Una vez se dispone de la orientación y la magnitud del gradiente para cada píxel se procede a calcular un histograma de las orientaciones de dicho gradiente en subregiones de 4x4 píxeles. Asumiendo que la ventana que se quiere describir tiene 16 x 16 píxeles, se divide ésta en 16 regiones menores, de 4 x 4 píxeles cada una. Para cada una de esas regiones se hace lo siguiente:

3. Se obtiene un *histograma de 8 direcciones para cada subregión*. Para ello se dividen los 360 grados de la circunferencia en 8 sectores circulares de 45 grados cada uno. Las orientaciones que caigan en el primer sector, de 0 a 45 grados, corresponderán con el primer *bin* del histograma, y así sucesivamente. Cada *bin* del histograma tendrá un valor correspondiente a la suma de las magnitudes de todos los píxeles cuya orientación corresponda con su sector circular.

4. Se *distribuye el valor de cada gradiente en los bins adyacentes* del histograma, mediante una interpolación tri-lineal, de forma que se evitan los efectos de frontera. Finalmente, cada una de las entradas a un *bin* del histograma se multiplica por un peso de $1 - d$ para cada dimensión, donde d es la distancia de la muestra al valor central del *bin* medida en unidades del espaciado de los *bins* del histograma.

Una vez obtenido el histograma orientado de gradientes para cada una de las 16 subregiones en las que se había dividido la ventana, se concatenan todos los histogramas formando un vector de características de 16 x 8 = 128 elementos.

10.2.3.3 Normalización del vector de características

El paso final consiste en normalizar el vector obtenido para mejorar la robustez del anterior descriptor frente a ciertos cambios de iluminación, producidos por la reflexión de la luz en superficies 3D o por efectos debidos a la saturación de la cámara.

SIFT realiza una doble normalización a longitud unidad de la siguiente manera:

- En primer lugar se realiza una primera normalización a longitud, dividiendo el vector por su norma.
- Después se umbraliza cada uno de los elementos del vector anterior, utilizando 0.2 como valor de corte, de manera que todos aquellos elementos del vector que tengan un valor

superior al de corte, se dejan con 0.2. Finalmente se vuelve a realizar una segunda normalización a longitud unidad, obteniendo de esta manera el vector definitivo.

Con todo lo anterior, el vector normalizado de 128 elementos conteniendo los histogramas de las orientaciones de los gradientes, queda calculado.

10.3. Generación del diccionario

La generación del diccionario consiste en obtener, a partir de N vectores de características (descriptores) extraídos de un conjunto de imágenes de entrenamiento, K palabras visuales mediante alguna técnica de agrupamiento (*clustering*) de datos, figura 10.6. Es importante resaltar que el diccionario no se puede generar usando descriptores de imágenes del grupo de pruebas.

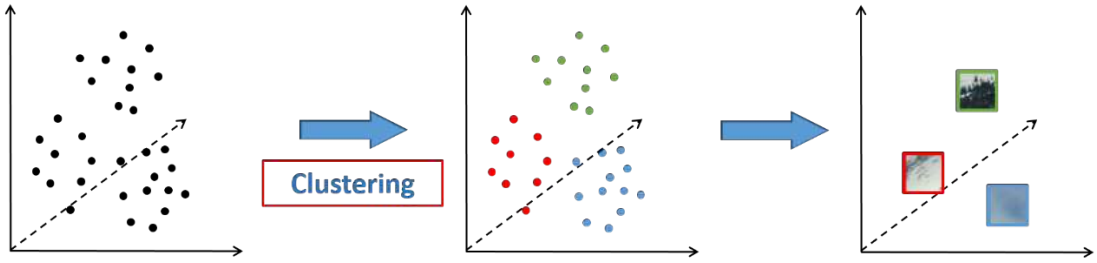


Figura 10.6. Diagrama de obtención de $K = 3$ palabras a partir de un conjunto de descriptores.

Los algoritmos de *clustering* agrupan un conjunto de elementos, representados por vectores de n características de acuerdo con un criterio, generalmente de distancia o similitud en el espacio de características. Idealmente, los agrupamientos deben ser tales que: (a) sean compactos (sus elementos son cercanos en el espacio de características) y (b) los distintos grupos estén lo más lejos posible entre ellos.

Existen varios tipos de algoritmos de *clustering*. Algunos de los más típicos son:

- ***Clustering* basado en centroides:** En estos algoritmos, cada agrupamiento está representado por un vector de características único, que no tiene que pertenecer al conjunto de datos original. Algunos ejemplos de este tipo de *clustering* son k-means, k-medians, k-medoids, fuzzy k-means, etc.

- ***Clustering* jerárquico:** Cuanto más relacionados entre sí están dos elementos, más cercanos se encuentran en el espacio de características. Siguiendo esa idea, este tipo de algoritmos "conectan" elementos para formar agrupamientos basados en sus distancias, de forma que un grupo se puede describir mediante la distancia necesaria para conectar sus elementos más lejanos. A menudo los agrupamientos obtenidos con este tipo de algoritmos se representan mediante dendrogramas. Un nuevo elemento se añade al grupo utilizando un criterio de unión o *linkage* que contempla la distancia del elemento a añadir al centroide del grupo, a su elemento más cercano, más alejado, etc. En función de ello se habla de diferentes métodos

de unión, o *linkage*, como son *single-linkage-clustering*, *complete linkage clustering*, UPGMA, etc.

- ***Clustering* basado en distribuciones:** Los algoritmos de agrupamiento de este tipo se basan en la consideración de que los elementos de un mismo grupo pertenecen a la misma distribución de probabilidad. Un método muy popular dentro de este tipo de algoritmos es el Gaussian Mixture Model (usando el algoritmo Expectation-Maximization, o EM), que trata de agrupar los datos asumiendo diferentes distribuciones normales, y variando los parámetros iterativamente hasta conseguir la convergencia.

- ***Clustering* basado en densidad:** En este tipo de algoritmos los grupos se definen como áreas de densidad más alta (en el espacio de características) que el resto de datos. Los datos que no pertenecen a estas regiones con alta densidad se consideran ruido o puntos "frontera" entre distintos grupos. Ejemplos de algoritmos de *clustering* basado en densidad son DBSCAN, OPTICS, o Mean-Shift.

En este capítulo sólo se explicará en detalle el algoritmo k-means, propuesto originalmente en (Hartigan, 1979). La razón es que este algoritmo representa cada agrupamiento de descriptores (palabra visual) mediante un vector de características (el prototipo, o centroide, del *cluster*), siendo este tipo de entrada la más adecuada para el proceso de clasificación posteriormente explicado.

10.3.1. Algoritmo de *clustering* K-means

Supongamos que tenemos un conjunto de datos $\{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_N}\}$ que consiste en N observaciones de una variable D-dimensional $\mathbf{x}$. El objetivo es dividir los datos en un número de *clusters* K, que vamos a suponer dado. Como ya se indicó anteriormente, se considera que los elementos de un *cluster* se sitúan a distancias pequeñas entre ellos en comparación con las distancias a los puntos que no pertenecen a ese *cluster*. Formalizaremos esta noción introduciendo un conjunto de vectores D-dimensionales $\boldsymbol{\mu}_\mathbf{k}$, con $k = 1, 2, \ldots, K$, de forma que $\boldsymbol{\mu}_\mathbf{i}$ es un prototipo asociado al *cluster* i-ésimo. El objetivo, por lo tanto, es calcular (a) los vectores $\{\boldsymbol{\mu}_\mathbf{k}\}$ y (b) una asignación de los datos a dichos vectores de manera que la suma de los cuadrados de las distancias entre cada punto $\mathbf{x_n}$ y el representante $\boldsymbol{\mu}_\mathbf{i}$ del *cluster* al que se ha asignado sea mínima. Esto se puede llevar a cabo mediante un proceso iterativo en el que en cada repetición se realicen dos pasos: Se comienza asignando unos valores iniciales a los vectores $\{\boldsymbol{\mu}_\mathbf{k}\}$. A continuación, en una primera fase, se asigna cada punto $\mathbf{x_n}$ al *cluster* cuyo prototipo $\boldsymbol{\mu}_\mathbf{i}$ sea más cercano. En una segunda fase se recalcula cada prototipo $\boldsymbol{\mu}_\mathbf{i}$ como la media de los puntos $\mathbf{x_n}$ asignados al *cluster* correspondiente.

Ambas fases se repiten hasta que las asignaciones no cambien, o bien hasta que se llegue a un número máximo de iteraciones. La convergencia de este algoritmo está garantizada debido a que en cada iteración se reduce la suma de las distancias entre los puntos y los representantes del *cluster* al

que pertenecen o, en el peor caso, esta suma no cambia. Sin embargo, el algoritmo puede converger a un mínimo local en vez de a un mínimo global.

La figura 10.7 ilustra las etapas del algoritmo K-means:

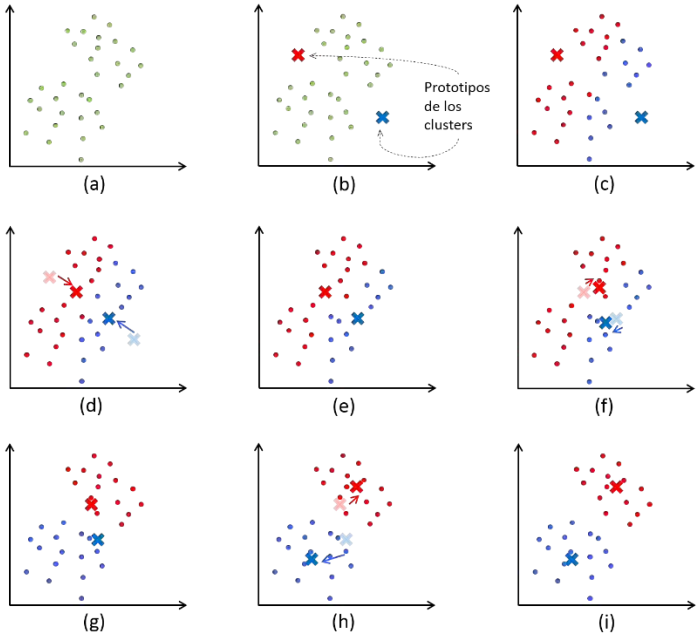


Figura 10.7. Ilustración del algoritmo K-means. (a) Los puntos verdes denotan los datos en un espacio Euclídeo bidimensional. (b) Inicialización de los prototipos μ_1 y μ_2 (cruces roja y azul). (c) Asignación inicial de cada punto a uno de los *clusters*. Cada punto se asigna al *cluster* rojo o azul en función de la cercanía de los prototipos correspondientes. (d) Actualización de los prototipos como la media de los nuevos puntos asignados a cada *cluster*. (e) - (i) Sucesivos pasos hasta la convergencia final del algoritmo.

Nótese que en el ejemplo anterior se han inicializado puntos no óptimos deliberadamente. En la práctica un método mejor de inicialización será escoger como prototipos $\{\boldsymbol{\mu}_k\}$ un subconjunto de K puntos de entre el conjunto de datos original. De hecho, la elección aleatoria de puntos podría dar lugar a que no se asignase ningún punto a un prototipo determinado, como se puede ver en la figura 10.8.

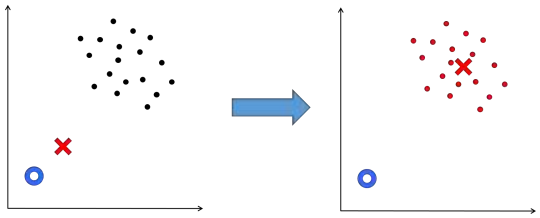


Figura 10.8. Ilustración de inicialización incorrecta de prototipos: ninguno de los puntos del conjunto de datos se asignará al *cluster* representado en rojo.

Esta es una breve introducción al método K-means. Una explicación más detallada puede encontrarse en la sección 9.4.1 del capítulo 9, “Clasificación y Reconocimiento de patrones”.

10.4. Representación de imágenes

Una vez creado el diccionario de palabras visuales mediante alguna técnica de *clustering* tal como K-means es el momento de representar la imagen. En el modelo Bag of Visual Words de clasificación de imágenes, representar una imagen consiste en describirla de manera global mediante un histograma de ocurrencias de palabras visuales en el diccionario. Dicho de otro modo, para cada descriptor de la imagen es necesario buscar la palabra del diccionario más similar. Una vez encontradas dichas palabras para todos los descriptores de una imagen, ésta se construye mediante el histograma de frecuencias de las mismas. Para determinar cuan parecidas son dos palabras se utilizará una medida de distancia entre sus descriptores (por ejemplo SIFT). Este proceso se ilustra en la figura 10.9.

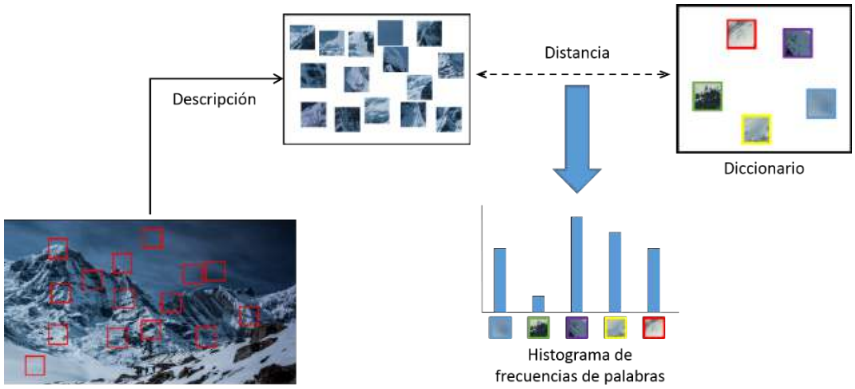


Figura 10.9. Ilustración del proceso de representación de imágenes.

En concreto, para cada uno de los n puntos característicos muestreados de la imagen se calculará la distancia entre su descriptor y los vectores que representan cada una de las K palabras del diccionario obtenido en el proceso de construcción del mismo (esto es el *clustering* de las palabras). Formalizando esta idea, sean $\mathbf{x}_i$ el vector que describe el i-ésimo descriptor de la imagen y μ_j el vector correspondiente a la j-ésima palabra del diccionario. Tras calcular la distancia entre ellos, $d_{ij} = \| \mathbf{x_i} - \boldsymbol{\mu}_\mathbf{j} \|_2$, $\forall j \in 1,\ldots K$ [1], se puede formar el vector p, con $p(i) = k$, donde k es el índice de la palabra $\boldsymbol{\mu}_\mathbf{k}$ que hace que $d_{ik} \leq d_{ij}$, $\forall j = 1,\ldots K$. Finalmente, el descriptor global (esto es la representación) de la imagen consiste en el histograma de frecuencias de p.

El resultado es un conjunto de vectores que contienen los histogramas de frecuencias de las K palabras visuales del diccionario. Cada vector está asociado a una imagen del *dataset*.

10.5. Clasificación

La última etapa del sistema es la de clasificación. Esta etapa consiste en decidir la clase a la que pertenece una imagen cualquiera a partir de sus descriptores. En el proceso de creación del modelo

[1] En este caso se muestra la distancia Euclídea, pero podría utilizarse cualquier otra medida de distancia (Manhattan, Mahalanobis, etc.)

sólo han participado las imágenes del conjunto de entrenamiento. Con ellas se ha creado el diccionario que ha servido para describir, a través del histograma de frecuencias de las palabras visuales, las imágenes del conjunto de prueba. Si se quiere evaluar el modelo creado, se realizará un proceso de test con las imágenes separadas previamente para ese fin. Esta evaluación da una idea de cómo se comportará este modelo a la hora de clasificar una imagen nueva, perteneciente a una clase desconocida. La evaluación, expresada típicamente mediante alguna medida basada en la precisión (precisión) o la rememoración (recall), como puede ser el F-Score, indica si el modelo creado generalizará bien. Como hemos indicado, cuando se utiliza Bag of Visual Words cada imagen se describe mediante su histograma de frecuencias de palabras visuales, siendo ese el vector de características utilizado.

En general, como se ha explicado en el capítulo 9, existen dos enfoques diferentes para crear un modelo:

1. **Aprendizaje no supervisado:** Los algoritmos de aprendizaje no supervisado tratan de ajustar un modelo de clasificación a partir de un conjunto de datos no etiquetados (de los que no se conocen las clases a las que pertenecen) $\{\mathbf{x_1},\ldots\mathbf{x_N}\}$. Para realizar este ajuste, el algoritmo debe encontrar de forma autónoma las características, regularidades y correlaciones de los datos, de forma que pueda separar las categorías similares entre sí. Algunos ejemplos de este tipo de aprendizaje son los algoritmos de *clustering* mencionados en la sección 10.3., *Self-Organising Maps* (SOM) o *Adaptive Resonance Theory* (ART).

2. **Aprendizaje supervisado:** Los algoritmos de aprendizaje supervisado ajustan el modelo, o función, de clasificación mediante un conjunto de datos cuya clase es conocida, llamado *conjunto de entrenamiento*. Formalmente este conjunto se representa como $\{(\mathbf{x_1},\mathbf{y_1}),\ldots(\mathbf{x_N},\mathbf{y_N})\}$, donde $\mathbf{x_i}$ es el *i*-ésimo vector del conjunto de entrenamiento mientras que $\mathbf{y_i}$ es la clase a la que pertenece dicho vector. Una vez ajustado el modelo, la función de clasificación es capaz de asignar la clase $\mathbf{y_k}$ a la que pertenecen nuevos elementos $\mathbf{x_k}$. El conjunto de elementos de clase desconocida se llama **conjunto de test**. Algunos ejemplos de algoritmos de aprendizaje supervisado son las redes neuronales artificiales, la regresión logística, los *k* vecinos más cercanos (k-Nearest Neighbours o kNN), las máquinas de vector de soporte (Support Vector Machines), etc.

Este capítulo explicará brevemente qué es y cómo se usa el aprendizaje supervisado mediante máquinas de vector de soporte (SVM) en la clasificación de imágenes.

10.5.1. Máquinas de Vectores Soporte (Support Vector Machines)

Las máquinas de vectores soporte (SVM por sus siglas en inglés *Support Vector Machines*) (Vapnik, 1995) constituyen una de las técnicas de clasificación supervisada más potentes, y un estándar en el estado del arte actualmente. Se trata de un clasificador biclase (esto es permite clasificar

en dos clases), aunque la mayoría de implementaciones del mismo, (como, por ejemplo, LIBSVM, LIBLINEAR) soportan clasificación multi-clase.

Al igual que cualquier clasificador supervisado biclase, el objetivo de un SVM es inferir una frontera de decisión (lineal o no) en el espacio de características de modo que las observaciones posteriores se clasifiquen automáticamente en uno de los dos grupos definidos por dicha frontera (también conocida como *hiperplano*). La particularidad de SVM es que trata de generar dicho hiperplano de modo que maximice su separación con cada uno de los grupos, figura 10.10. Por ello se dice que este clasificador es un clasificador de margen máximo.

Si consideramos elementos en un espacio D-dimensional, i.e. $\circ^{D}$, el hiperplano está determinado por un vector $\boldsymbol{\theta} = (\theta_0, \theta_1, ..., \theta_D)$, de forma que su ecuación es:

$$\theta_0 + \sum_{i=1}^{D} \theta_i X_i = 0 \tag{10.3}$$

Así, dado un elemento $\mathbf{x} = (x_1, \ldots, x_D) \in \circ^{D}$, un clasificador asigna $\mathbf{x}$ a una clase o a otra en función de si $\theta_0 + \sum_{i=1}^{D} \theta_i x_i > 0$ o $\theta_0 + \sum_{i=1}^{D} \theta_i x_i < 0$. Sin embargo, en el caso de SVM esta asignación se realiza en función de si $\theta_0 + \sum_{i=1}^{D} \theta_i x_i > \gamma$ o $\theta_0 + \sum_{i=1}^{D} \theta_i x_i < \gamma$ con $\gamma > 0$.

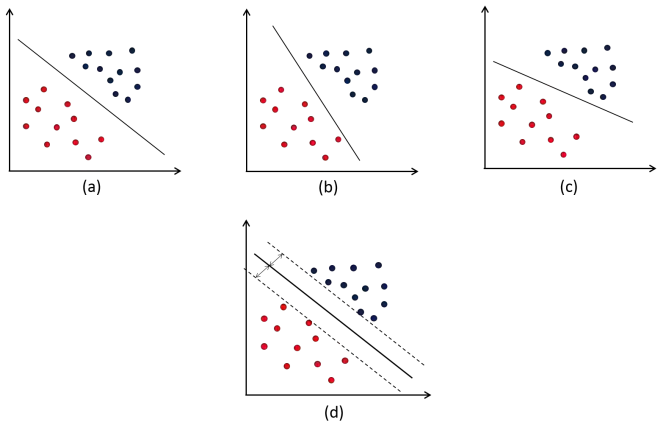


Figura 10.10. Ejemplos de hiperplanos que separan dos clases diferentes en un espacio bidimensional. (a)-(c) Diferentes hiperplanos. (d) Hiperplano de margen máximo.

Para inferir el hiperplano durante la fase de entrenamiento es necesario minimizar la función de coste:

$$J(\theta) = C \sum_{i=1}^{N} \left[\mathbf{y_i} cost_1(\boldsymbol{\theta}^T \mathbf{x_i}) + (1 - \mathbf{y_i}) cost_0(\boldsymbol{\theta}^T \mathbf{x_i}) \right] + \frac{1}{2} \sum_{j=1}^{D} \theta_j^2, \tag{10.4}$$

donde $cost_0(z)$ y $cost_1(z)$ son funciones de coste en función de si $\mathbf{y}_i = 1$ o $\mathbf{y}_i = 0$, respectivamente que tienen un aspecto similar al mostrado en la figura 10.11.

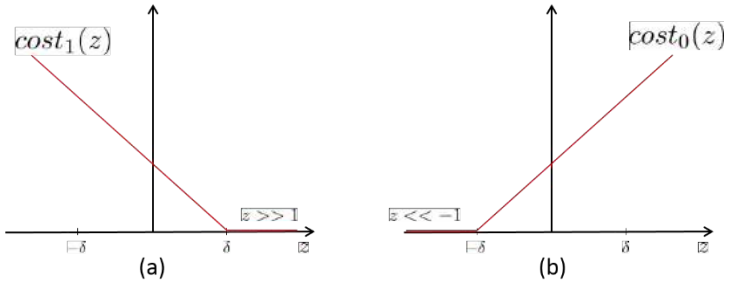


Figura 10.11. Ilustración de las funciones $cost_0$ y $cost_1$.

Otra de las particularidades de las SVM es su capacidad de obtener fronteras de decisión no lineales, figura 10.12 aumentando la dimensión del espacio de características mediante el uso de *kernels*. Por ejemplo, mapeando un conjunto de D características $\{X_1,\ldots,X_D\}$ en, por ejemplo, uno de $2D$ características $\{X_1,X_1^2,\ldots,X_D,X_D^2\}$. Esto es especialmente útil cuando las clases de los elementos que se van a clasificar **no son linealmente separables en el espacio original**. Este mapeado se lleva a cabo con las llamadas funciones *kernel* o, simplemente, kernels, $k(\mathbf{x},\mathbf{y})$, que se diseñan de forma que el valor que devuelven sea más pequeño cuanto más alejado esté $\mathbf{y}$ de $\mathbf{x}$. En la práctica, tanto los elementos $\mathbf{x}$ como $\mathbf{y}$ serán los elementos del conjunto de entrenamiento. En definitiva, los vectores que definen el hiperplano se calculan como una combinación lineal de imágenes del kernel k con parámetros α_i:

$$\sum_i \alpha_i k(\mathbf{x_i},\mathbf{x}) = cte \tag{10.5}$$

Algunos de los *kernels* más comunes son el lineal, ecuación (10.6), polinomial, ecuación (10.7), gaussiano, también conocido como *radial basis function*,ecuación (10.8) o sigmoidal, ecuación (10.9).

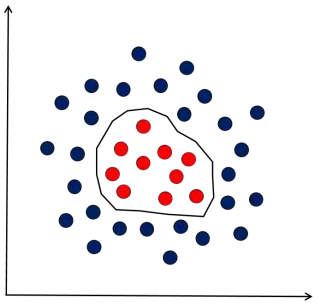


Figura 10.12. Ejemplo de frontera de decisión no lineal.

$$k(\mathbf{x_i},\mathbf{x_j}) = \mathbf{x_i}\cdot\mathbf{x_j} \tag{10.6}$$

$$k(\mathbf{x_i},\mathbf{x_j}) = (\mathbf{x_i}\cdot\mathbf{x_j})^d \tag{10.7}$$

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|), \ \gamma > 0^2 \quad (10.8)$$

$$k(\mathbf{x_i}, \mathbf{x_j}) = \tanh(\kappa \mathbf{x_i} \cdot \mathbf{x_j} + c), \ \kappa > 0, \ c < 0 \quad (10.9)$$

Se remite al lector interesado en más detalles sobre SVM al capítulo 5 de (Pajares y de la Cruz, 2010).

10.6. Bibliografía

Augereau, O.; Journet, N.; Vialard, A.; Domenger, J.-P. (2014) Improving Classification of an Industrial Document Image Database by Combining Visual and Textual Features. Proceedings del 11th IAPR International Workshop on Document Analysis Systems (DAS), Tours, Francia, 7-10 Abril 2014; pp. 314–318.

Bay, H.; Ess, A.; Tuytelaars, T. and Van Gool, L. (2008) Speeded-Up Robust Features (SURF). Computer Vision and Image Understanding 110(3): 346-359. doi:10.1016/j.cviu.2007.09.014.

Bishop, C.M. (2006) *Pattern Recognition and Machine Learning*, 1ª ed.; Springer Verlag: Nueva York, USA, pp. 423–428.

Brown, Matthew and Lowe, David G. "Invariant features from interest point groups," British Machine Vision Conference, BMVC 2002, Cardiff, Wales (September 2002), pp. 656-665

Burt, Peter and Adelson, Ted (1983). The Laplacian pyramid as a compact image code. IEEE Transactions on Communications 9(4): 532–540. doi:10.1109/tcom.1983.1095851.

Crowley, James L. and Stern, Richard M. (1984). Fast computation of the difference of low pass transform. IEEE Transactions on Pattern Analysis and Machine Intelligence 6(2): 212-222. doi:10.1109/tpami.1984.4767504.

CS231n Convolutional Neural Networks for Visual Recognition: Image Classification, data-driven approach, k-nearest neighbor. Disponible online: http://cs231n.github.io/classification/

CS231n Convolutional Neural Networks for Visual Recognition: Linear classification: SVM/Softmax. Disponible online: http://cs231n.github.io/linear-classify/

Dalal, N.; Triggs, B. (2005) "Histograms of oriented gradients for human detection," Computer Vision and Pattern Recognition CVPR 2005. IEEE Computer Society Conference on , vol.1, no., pp.886,893 vol. 1, 25-25. doi: 10.1109/CVPR.2005.177

Duda, R.O.; Hart P.E.; Stork, D.G. (2000) *Pattern Classification*, 2ª ed.; John Wiley & sons: Canada, pp. 259–268.

Fei-Fei, L.; Fergus R.; Torralba A. Recognizing and Learning Object Categories, CVPR 2007 short course. Disponible online: http://people.csail.mit.edu/torralba/shortCourseRLOC/index.html

Harris, C. and Stephens, M. (1988). "A combined corner and edge detector". Proceedings of the 4th Alvey Vision Conference. pp. 147–151.

Hartigan, J.A.; Wong, M.A. (1979) Algorithm AS 136: A K-Means Clustering Algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics), 28(1), 100–108.

[2] En el kernel gaussiano en ocasiones se utiliza el valor $\gamma = 1/2\sigma^2$

Jing, Z.; Lei, S.; Li, Z.; Zhenwei, L.; Yuncong, Y. (2013) An approach of bag-of-words based on visual attention model for pornographic images recognition in compressed domain. *Neurocomputing*, 110, 145–152.

Lindeberg, T. 1994. Scale-space theory: A basic tool for analysing structures at different scales. Journal of Applied Statistics, 21(2):224-270.

Lowe, David G. (1999). "Object recognition from local scale-invariant features". Proceedings of the International Conference on Computer Vision 2. pp. 1150–1157. doi:10.1109/ICCV.1999.790410.

Lowe, D.G. (2004) Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, 60(2), 91–110.

Mikolajczyk, K., and Schmid, C. 2002. An affine invariant interest point detector. In European Conference on Computer Vision (ECCV), Copenhagen, Denmark, pp. 128-142.

Nilsback, M-E. and Zisserman, A. 2006. A Visual Vocabulary for Flower Classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition

Pajares, G. y de la Cruz, J.M. (Eds.) (2010). Aprendizaje Automático: un enfoque práctico. RA-MA, Madrid.

Sheng, X; Tao, F.; Deren L.; Shiwei W. (2010) Object Classification of Aerial Images with Bag-of-Visual Words. *Geoscience and Remote Sensing Letters, IEEE*, 7(2), 266–370

Szelinski, R. (2011) *Computer Vision: Algorithms and Applications*, 1ª ed.; Springer Verlag: Nueva York, USA, pp. 658–729.

Sivic, J. and Zisserman A. (2003) Video Google: A text retrieval approach to object matching in videos. In Proceedings of the International Conference in Computer Vision (ICCV), volume 2, pages 1470–1477.

Vapnik, V. (1995) *The Nature of Statistical Learning Theory*, 2ª ed.; Springer Verlag: Nueva York, USA, pp. 123–170.

Vedaldi, A and Fulkerson, B. (2008). "VLFeat: An Open and Portable Library of Computer Vision Algorithms". Disponible on-line: http://www.vlfeat.org/

CAPÍTULO 11

REPRESENTACIÓN DE LA INFORMACIÓN 3D

Antonio ADÁN[1], Pilar MERCHÁN[2], Santiago SALAMANCA[2]

[1] Escuela Superior de Informática. Universidad de Castilla La Mancha. Ciudad Real. España
[2] Escuela de Ingenierías Industriales. Universidad de Extremadura, Badajoz. España

La representación 3D proporciona la base de varias disciplinas parcialmente ligadas y solapadas como son las de Visión por Computador, Gráficos por Computador y Diseño por Computador. En esta confluencia de disciplinas existen ciertos modelos de representación 3D comunes en mayor o menor medida a todos ellas. El objetivo de este capítulo se centra en presentar cuáles de estos modelos son utilizados en el campo de la Visión por Computador como base o como complemento para resolver problemas de mayor nivel de abstracción; desde la segmentación de nubes de puntos hasta la comprensión de los objetos de una escena compleja que ha sido adquirida con un sensor 3D. Se realiza una sucinta descripción de los mismos, estableciendo su fundamento y enfatizando su aplicabilidad en el área de la Visión por Computador.

11.1. Información 3D

El tipo de información que los sensores 3D capturan de la escena depende en gran medida del tipo de sensor utilizado. Las principales formas de proporcionar dicha información son las siguientes:

Nube de puntos: Se proporcionan las coordenadas tridimensionales de puntos del espacio normalmente en el sistema de referencia del propio sensor. Éste es el formato que devuelven usualmente los escáneres láser por tiempo de vuelo y cambio de fase, que actúan en grandes entornos.

Imagen de profundidad: La información en una imagen donde cada píxel contiene la distancia al centro óptico de la cámara utilizada por el sensor. En este caso también se obtiene la imagen color o en escala de grises de la escena captada. Formato habitual proporcionado por cámaras RGB-D.

Malla poligonal: Es un caso particular de nubes de puntos pero estructurada en polígonos (normalmente triangulares). Es un formato que puede ser encontrado en algunos sistemas de triangulación activa por láser.

Imagen de rango: Consiste en tres imágenes de profundidad según los tres ejes cartesianos del sistema de referencia del sensor. Podemos encontrar este tipo de información en sistemas de proyección por luz estructurada.

Imagen estereoscópica: Se proporcionan dos imágenes en color, cada una correspondiente a un sensor de imagen. Se utiliza normalmente para aplicaciones de visualización en tres dimensiones. Este formato es devuelto por los sistemas estereoscópicos.

Aparte del tipo de información que es devuelto por el propio sensor 3D, es importante conocer los principales formatos digitales ligados a etapas posteriores del procesamiento de datos 3D.

11.1.1. Formatos digitales

La información capturada por los sensores, explicada en la sección anterior, es almacenada en ficheros que, de una forma u otra, representan la información capturada. A lo largo de los años se han definido infinidad de formatos digitales, algunos con más éxito que otros. Una clasificación sencilla que podríamos utilizar para agrupar estos formatos es la siguiente:

a) Formatos que almacenan nubes de puntos 3D: La característica común entre todos ellos es que almacenan, exclusivamente, información relativa a las nubes de puntos 3D de los objetos, obviando cualquier información topológica.

b) Formatos que almacenan información de superficie de los objetos 3D: En este caso, además de la información asociada a los puntos 3D, también pueden almacenar información superficial, que dependiendo del tipo de fichero, puede representarse de forma no paramétrica y/o paramétrica. Los formatos CAD (*Computer-Aided Design*) se englobarían dentro de esta categoría.

c) Formatos que almacenan información extendida: Dentro de esta categoría se encuentran aquellos formatos que, además de la información 3D, permiten almacenar otro tipo de información, como por ejemplo, vídeos, audio, etc. Es el tipo de formatos que se emplea para los sistemas de realidad virtual o aumentada.

A continuación se describen con más detalle algunos de los formatos más usuales empleados hoy en día.

11.1.1.1 Formatos que almacenan nubes de puntos 3D

Los formatos más habituales son *xyz*, *pts*, *asc* o *3d*, que son variantes de ficheros de tipo *csv* (valores separados por coma). En el caso más básico, son ficheros de texto de forma que en cada fila se almacenan las coordenadas 3D, según un sistema de referencia cartesiano (x, y, z), de los puntos. Estas coordenadas están separadas entre sí por una coma, aunque también se pueden usar espacios o tabulaciones o punto y coma.Otras variantes almacenan, tras las coordenadas, información de textura, reflectividad o cualquier otra información que pueda estar relacionada con los puntos 3D.

Una variante de los formatos anteriores es el *pcd.* Es el formato nativo de la librería de programación gratuita Point Cloud (PCL, *pointclouds.org*), dedicada al tratamiento de nubes de puntos e imágenes 3D. En este caso, el fichero comienza con una cabecera en la que se identifican y declaran distintas propiedades de los datos que se almacenarán. Esta cabecera se encuentra codificada en ASCII

(*American Standard Code for Information Interchange*). Alguna información que se puede incluir en esta cabecera es la versión del tipo de fichero *pcd*, los nombres que cada campo del fichero puede tener, el tamaño en bytes de cada uno de los campos, los tipos de datos que los definen, si se van a definir texturas o normales para los vértices, etc. Tras la cabecera, el resto del fichero, que puede escribirse en formato de texto o binario, está asociado a la definición de los datos indicados en la cabecera.

11.1.1.2. Formatos que almacenan información de superficie de los objetos 3D

Dentro de esta categoría se encuentran infinidad de tipos. Los más usuales son los siguientes:

Formato stl (Hon Wah, 1999), que es un tipo de fichero CAD cuyo nombre es el acrónimo de STereoLitography, fue creado por la empresa 3D Systems (*www.3dsystems.com*) para su uso en la industria de prototipado rápido y sistemas de fabricación asistida por ordenador de máquinas de fabricación.
Los ficheros pueden estar en formato texto o binario. Sólo almacenan información geométrica, por lo que no es un formato útil cuando necesitamos conocer la textura de los objetos.
En el caso de los ficheros de texto, siempre se comienza con una cabecera que es:

solid *nombre*

donde *nombre* es opcional. La forma de terminar el fichero es con la palabra clave endsolid
Entre una y otra palabra clave se definen las distintas facetas de la malla de la siguiente forma

facet normal n_x n_y n_z
outer loop
vertex v_{1x} v_{1y} v_{1z}
vertex v_{2x} v_{2y} v_{2z}
vertex v_{3x} v_{3y} v_{3z}
endloop
endfacet

donde (n_x n_y n_z) son las coordenadas del vector unitario normal a la faceta y cuyo sentido es hacia el exterior del objeto. Con (n_x n_y n_z) se definen los distintos vértices de las facetas, normalmente en sentido contrario a las agujas del reloj visto desde el exterior del objeto. La definición de las coordenadas del vector normal puede ser (0, 0, 0), indicando, en ese caso al software encargado de leer el fichero, que las normales no se han especificado.

Formato ply, o *Formato de Triángulos de Stanford* (Bourque, 2009), debido a que fue creado en esta universidad, es un tipo de formato en el que se amplía, con respecto al formato stl, la información que se puede almacenar, convirtiéndolo en un método de almacenamiento mucho más flexible. Al igual que el formato anterior, los ficheros pueden ser de tipo texto o binario.

La estructura típica de un fichero *ply* es la siguiente:

Cabecera

Lista de vértices
Lista de caras o facetas
(Listas de otros elementos)

La cabecera es un conjunto de líneas en las que se describe cómo va a ser el resto del fichero. La información que se puede incluir en esta cabecera es muy extensa y permite que los datos se puedan presentar de muchas maneras, lo que hace que éste sea uno de los formatos más flexibles. La lista de vértices y la de caras especifica estos elementos de la malla. Por último, se pueden incluir otros elementos de la malla como, por ejemplo, la textura.

Formato obj (Murray y Van Ryper, 2005), es un formato digital creado por la empresa Wavefront Technologies (*www.wvfront.com*). Es un tipo de fichero que ofrece una muy alta flexibilidad, permitiendo el almacenamiento, no sólo de mallas poligonales, sino también de superficies y curvas paramétricas. Es muy utilizado dentro del ámbito de las gráficas por computador, por lo que se puede también incluir información para la visualización y el *renderizado* de los objetos 3D que almacene.

Al igual que en los formatos anteriores, los ficheros *obj* pueden ser binarios o de texto. El fichero empieza normalmente con una cabecera que emplea el carácter de almohadilla (#) como comienzo de las líneas. Tras la cabecera se incluyen en cada línea los elementos del objeto (vértices, caras, normales, texturas), que para ser reconocidos van precedidos de una palabra. Así, por ejemplo, si en una línea se escribe

$$v\ v_x\ v_y\ v_z\ r\ g\ b$$

se está indicando con la palabra clave v que lo que en esa línea se está definiendo es un vértice, que sus coordenadas cartesianas son (v_x, v_y, v_z) y que su textura es (r, g, b). El número de palabras claves es extensísimo, lo que le da la flexibilidad a este formato de datos 3D, y para cada palabra clave existe un conjunto de valores obligatorios y opcionales.

11.1.1.3. Formatos que almacenan información extendida

En esta categoría el formato más relevante es el *x3d*. Es un tipo de formato que va más allá de los objetivos de este capítulo, por lo que sólo comentaremos brevemente algunas características básicas. Para un estudio profundo sobre este formato se puede consultar el libro de Brutzman y Daly (2007).

Es, además de un formato estándar ISO, una arquitectura de ejecución para representar y comunicar escenas y objetos 3D usando XML. Proporciona soporte para gráficas 3D, datos CAD, animaciones, vídeo y audio especializado, interacción de usuarios, geolocalización, etc. El formato VRML, que ha sido desde el año 1995 el formato más usado para representación de escenas 3D en web, ha sido integrado en *x3d* desde el año 2001.

11.2. Modelos de representación no paramétricos

Los modelos no paramétricos, en contraposición con los paramétricos, son aquellos que no necesitan parámetros para su definición. Actualmente son los modelos más empleados dentro del

ámbito de la Visión por Computador debido a que son los datos más cercanos a los que proporcionan los sensores 3D y a que son más sencillos de tratar computacionalmente.

11.2.1. Representación de nubes de puntos

El tipo de dato más sencillo que se puede representar es la nube de puntos, que se trata de un conjunto de puntos definidos por sus coordenadas, normalmente, cartesianas *x*, *y*, *z*.

Las formas de representar las nubes de datos son dos. En primer lugar, las nubes de puntos desorganizados, que es una mera lista de todos los vértices que componen la nube de puntos. No contiene ningún tipo de información topológica. En segundo lugar, las nubes de puntos estructuradas, donde los puntos se especifican con un orden determinado, usualmente, el definido por una matriz. En este caso sí existe relación topológica entre los puntos. Es la forma de representación más común en los escáneres 3D. En la figura 11.1 se representa una nube de puntos desorganizada y otra estructurada

(a) (b)

Figura 11.1. *a)* Nube de puntos estructurada. *b)* Nube de puntos desorganizada

Las imágenes de rango son un tipo de nubes de puntos estructurados. En este caso, en vez de almacenarse las coordenadas *x*, *y*, *z*, se almacenan las distancias a los puntos de la superficie de un objeto desde un sistema de referencia conocido. Las imágenes de rango son denominadas de diferente forma según el contexto de trabajo: mapa de rango, mapa de profundidad, imagen de profundidad, imagen 3D, imagen 2 ½ D, lista de puntos *xyz*, mapas digitales de terreno (DTM), etc. En la figura 11.2 se muestra un ejemplo de una imagen de rango representada en forma de imagen de nivel de gris, correspondiente a los niveles de profundidad y en formato de mapa de alturas.

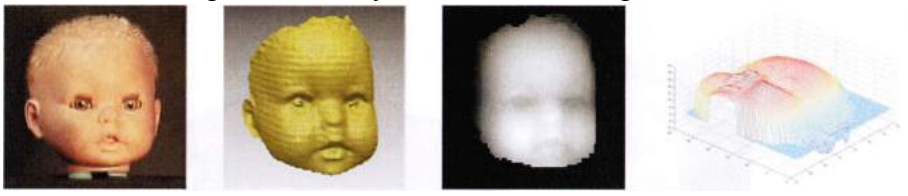


Figura 11.2. Dos representaciones de una vista parcial de un objeto *a)* Fotografía del objeto representado; *b)* Malla poligonal; *c)* Representación de la imagen de rango en formato de imagen de nivel de gris; *d)* Representación de la imagen de rango en formato de mapa de alturas.

11.2.2. Representación de superficies: mallas poligonales.

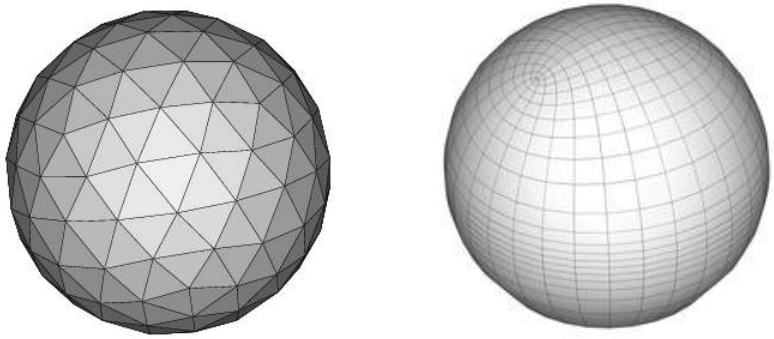
El tipo de modelo no paramétricos de superficies es la malla poligonal. Se define matemáticamente (Salamanca, 2005) como el par

$$\mathcal{O} = \langle \mathcal{P}, \mathcal{V} \rangle \tag{11.1}$$

donde $\mathcal{V} = \{v_1, \dots, v_n\}$ es el conjunto de vértices o nodos 3D, siendo $v_i = \{x_i, y_i, z_i\}$, y $\mathcal{P} = \{p_1, \dots, p_m\}$ una lista de elementos que relacionan los vértices entre sí. Estas relaciones suelen ser las aristas que unen a un par de vértices y/o los polígonos o *patches* formados por éstos.

Se pone de manifiesto, tras esta definición, que las mallas poligonales tienen asociadas dos tipos de propiedades: geométricas y topológicas. Las primeras están relacionadas con las coordenadas de los vértices, es decir, con su posición en el espacio. Las segundas están referidas a la relación existente entre los vértices o nodos de la malla.

Teniendo en cuenta la información que almacene $\mathcal{P}$, se puede hacer varias clasificaciones de mallas.



(a) (b)

Figura 11.3. Representación de una esfera mediante *a)* una malla no estructura (existen nodos con 5 o 6 vecinos) y *b)* malla estructurada.

Si lo que se almacena en $\mathcal{P}$ son las aristas que unen los distintos vértices, se considerará la relación de vecindad entre ellos. Teniendo en cuenta esta relación, las mallas se pueden clasificar como:

- *Mallas no estructuradas*, que son aquellas donde el número de vecinos de cada nodo es variable.
- *Mallas estructuradas*, donde el número de vecinos de cada nodo es constante.

Las segundas presentan ciertas ventajas frente a las primeras, ya que son más simples, requieren menos memoria para su almacenamiento y permiten un control más directo sobre la forma de los *patches*. Además, existe un gran número de algoritmos desarrollados para este tipo de mallas y el cálculo de las propiedades de los nodos es más sencillo. La gran desventaja es la poca flexibilidad que presentan cuando tiene que representar superficies complejas, ya que suelen presentar suavizados en las zonas de alta curvatura.

La decisión de elegir un tipo u otro de malla dependerá de la aplicación a la que irá destinada el modelo de malla. Normalmente, para modelos CAD, cuyo objetivo es representar los objetos con precisión, se emplean mallas no estructuras. En la figura 11.3 se muestra un ejemplo de representación de una malla no estructurada y estructurada.

Otra clasificación que se puede realizar basándonos en las propiedades topológicas de la malla es en función del tipo de polígono o *patch* que se emplee. En este caso, cada elemento de $\mathcal{P}$ almacenará el conjunto de índices que definen los polígonos de la malla. En principio, los polígonos pueden ser muy diversos, pero los más habituales son los polígonos triangulares, cuadrangulares y hexagonales. De entre los tres, los más usados son los *patches* triangulares, ya que constituyen el tipo de polígono más sencillo y permiten calcular los vectores normales a la superficie, necesarios en muchos algoritmos, entre otros, en el de la simulación del proceso de reflexión de la luz (primordial en aplicaciones de gráficas por computador).

Por último, comentar que en los casos en los que la malla se define como una colección desordenada de polígonos, sin información de las posiciones relativas de unos con respecto a otros, se denomina *sopa de polígonos*. A esta estructura, en algunos motores de cálculo de físicas (colisiones, dinámicas, etc.), se les suele llamar *trimesh* y, en estos casos, es una de las representaciones más útiles.

11.2.2.1. Mallas triangulares

De entre todos los tipos de mallas existentes, las más importantes son las mallas triangulares que, como se dijo en la sección anterior, son aquellas en las que los polígonos que se emplean para representar la información superficial es el triángulo.

Formalmente, una malla triangular $\mathcal{M}$ consiste en un componente geométrico y otro topológico, en el que la superficie viene representada por un conjunto de vértices

$$\mathcal{V} = \{v_1, \dots, v_n\}, v_i \in \mathbb{R}^3 \tag{11.2}$$

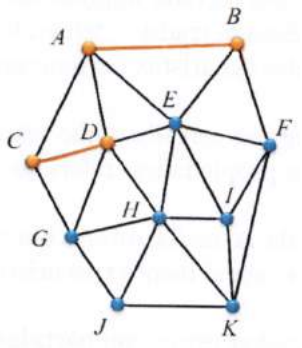
y un conjunto de caras triangulares que los conectan

$$\mathcal{T} = \{t_1, \dots, t_n\}, t_i \in \mathcal{V}\times\mathcal{V}\times\mathcal{V} \tag{11.3}$$

donde en cada triángulo se especifican los vértices que lo forman. La conectividad de la malla también puede venir definida en términos de las aristas

$$\mathcal{A} = \{a_1, \dots, a_n\}, a_i \in \mathcal{V}\times\mathcal{V} \tag{11.4}$$

La superficie está formada, por tanto, por una red de triángulos cuyos vértices son una nube de puntos cuya conectividad viene definida por la 2-tupla $\{\mathcal{T}, \mathcal{A}\}$. Se puede decir que la malla triangular es un grafo $\mathcal{G}(\mathcal{V}, \mathcal{A})$ en el que se establece una relación de conectividad entre los vértices a través de las aristas, tal como puede observarse en la figura 11.4.



$\mathcal{G}(\mathcal{V}, \mathcal{A})$ ≡ Grafo malla triangular
$\mathcal{V}$ ≡ vértices=$\{A, B, C, \dots, K\}$
$\mathcal{A}$ ≡ arístas=$\{(A\ B), (AE), (CD), \dots\}$
$\mathcal{T}$ ≡ triángulos=$\{(ABE), (DHG), \dots\}$

Figura 11.4. Entidades que componen la malla triangular e interconexión entre ellas.

Una característica topológica que deben verificar las mallas triangulares es que deben ser variedades bidimensionales (en inglés, *2-manifold*). Ésto, en una superficie discreta, o lineal continua definida a tramos, como es la malla triangular, se traduce en que no deben existir puntos singulares donde la superficie intersecte consigo misma o se abra en varias hojas, figura 11.5.

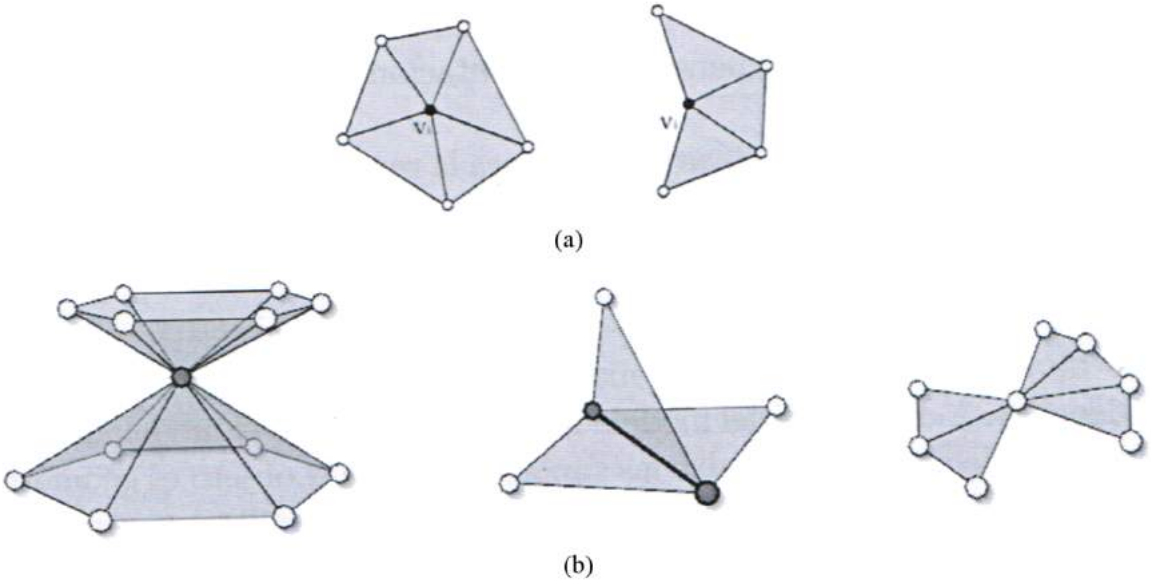


Figura 11.5. Variedades *(a)* bidimensionales y *(b)* no bidimensionales.

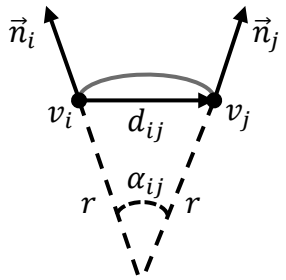


Figura 11.6. Sección, *s*, a lo largo de una superficie de curvatura constante.

Si una malla es una variedad no bidimensional, muchos algoritmos que trabajan con ella fallan, ya que la relación de vecindad no queda bien definida en el entorno de las zonas que ocasionan que la malla no sea una variedad no bidimensional.

Por otro lado, si una malla es cerrada, quiere decir que existe continuidad en toda la red de triángulos que modelan la superficie. Esto se traduce en que todas las aristas de la malla pertenecen exclusivamente a dos triángulos.

Otro concepto que se emplea para caracterizar este tipo de mallas es el del grado de un vértice, $grad(v_i)$, que se define como el número de aristas que inciden sobre dicho vértice. De esta forma, en el caso de mallas triangulares cerradas y que sean una variedad bidimensional, siempre se cumplirá que $grad(v_i) \geq 3$. Además, si la malla es estructurada, $grad(v_i)$ es constante en $\mathcal{G}(\mathcal{V}, \mathcal{A})$.

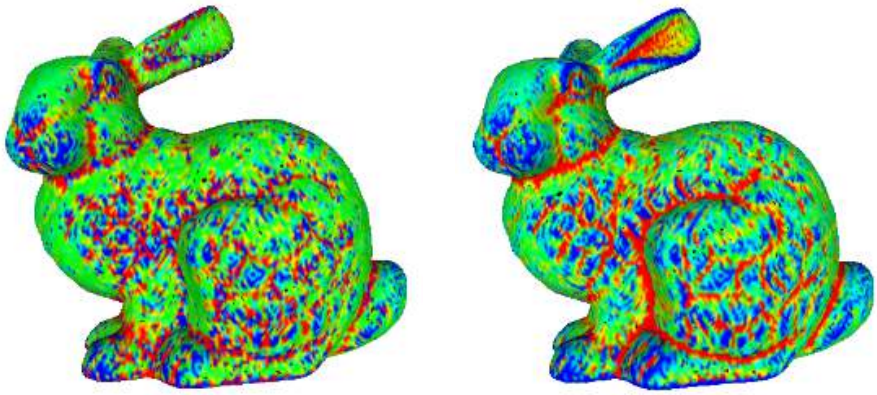
11.2.2.2. Cálculo de curvaturas a partir de una malla triangular

Además de las propiedades que se han comentado en apartados anteriores, las mallas triangulares permiten determinar propiedades de la superficie muy importantes. Quizás las más importantes son las medidas de curvatura

Si se asume que la malla triangular se aproxima a una superficie suave y la resolución de la malla es lo suficientemente alta como para anular las variaciones en la curvatura de la superficie entre vértices adyacentes, la superficie puede ser aproximada con más precisión con mallas de aristas circulares. En este modelo simplificado, la curvatura es muy sencilla de calcular.

En la figura 11.6 se muestra una sección, s, a través de una zona de superficie de curvatura constante que se encuentra entre dos vértices de la malla, v_i y v_j. La curvatura c_{ij} de s se calcula como

$$c_{ij} = \pm\frac{1}{r} \approx \pm\frac{\alpha_{ij}}{d_{ij}} \approx \cos^{-1}(\vec{n}_i \cdot \vec{n}_j) \tag{11.5}$$



(a) (b)

Figura 11.7. Representación, mediante código de color, de la *(a)* curvatura gaussiana y *(b)* curvatura media.

El signo positivo se refiere al caso de superficies cóncavas y el signo negativo a superficies convexas. Los vectores $\vec{n}_i$ y $\vec{n}_j$ son los vectores normales a la superficie en los vértices v_i y v_j respectivamente.

Por tanto, el cálculo de la curvatura a partir de una malla triangular es sencillo siempre que se conozcan los vectores normales asociados a los vértices. Éstos, a su vez, son muy fáciles de determinar a partir de los vectores normales de los triángulos a los que pertenezca. Así, si el vértice v_i forma parte, por ejemplo, de los triángulos t_k, t_l, t_m y t_n, cuyos vectores normales son $\vec{n}_k$, $\vec{n}_l$, $\vec{n}_m$ y $\vec{n}_n$, su vector normal, $\vec{n}_i$, se calculará como el valor medio de cada una de las coordenadas de los vectores normales de los triángulos y se dividirá por su módulo para hacerlo unitario.

Las curvaturas principales $\mathcal{K}_1(i)$ y $\mathcal{K}_2(i)$ de v_i son los valores extremos de c_{ij} con respecto a todos sus vecinos:

$$\mathcal{K}_1(i) \approx \min_{j}(c_{ij}) \tag{11.6}$$

$$\mathcal{K}_2(i) \approx \max_{j}(c_{ij}) \tag{11.7}$$

A partir de las curvaturas principales, se puede calcular la curvatura gaussiana, que es el producto de las dos curvaturas principales, o la curvatura media, que es el valor medio de $\mathcal{K}_1(i)$ y $\mathcal{K}_2(i)$. En la figura 11.7 se representa con un código de color la curvatura gaussiana y media de una superficie de un objeto de forma libre representada mediante una malla triangular.

11.2.3. Fases de creación del modelo

Independientemente del tipo de sensor utilizado para obtener la información tridimensional, hasta el más simple de los objetos necesita varias tomas parciales para adquirir la información de toda su superficie. En el proceso de construcción del modelo final (Campbell y Flynn, 2001) han de completarse las fases de registro; triangulación y fusión de las vistas parciales, dependiendo del tipo de dato de entrada con el que trabajemos; y procesado de la malla única obtenida.

11.2.3.1. Registro

Esta primera etapa consiste en alinear todas las vistas parciales, ya vengan dadas en forma de nube de puntos o como mallas, en un sistema de referencia común.

En ocasiones se puede encontrar un registro inicial utilizando una mesa de rotación para escanear el objeto. Ésta consiste básicamente en un plato giratorio accionado por un motor que produce una rotación del objeto a intervalos de grados determinados, para así ir obteniendo vistas del objeto espaciadas uniformemente a lo largo del giro de 360º. Esta solución simple limita enormemente el tamaño y la complejidad geométrica de los objetos a escanear, y precisa de una variación manual de la orientación del objeto con respecto al escáner para obtener las zonas de la superficie que no son captadas al girarlo.

La mayoría de los sistemas parten de un alineamiento interactivo en el que el operador humano selecciona dos tomas contiguas que se superponen parcialmente, y debe identificar al menos tres pares de puntos correspondientes en dichas vistas para obtener una trasformación rígida que alinea las tomas parciales en una primera aproximación. A continuación, este registro inicial debe ser refinado. Las mejores soluciones al registro fino las proporcionan aproximaciones basadas en el algoritmo ICP (*Iterative Closest Point*) (Rusinkiewicz y Levoy, 2001) .

El algoritmo ICP consta de dos pasos: en el primer paso, se identifican pares de puntos correspondientes candidatos en la zona de solapamiento de dos escaneos. Posteriormente, un procedimiento de optimización calcula una transformación rígida que reduce la distancia (en el sentido de mínimos cuadrados) entre los dos conjuntos de puntos. El proceso itera hasta que se satisface un criterio de convergencia. La idea general es que en cada iteración la distancia entre las dos exploraciones se reduce, lo que permite una mejor identificación de pares coincidentes verdaderos, y por lo tanto una mayor probabilidad de una mejor alineación en la próxima iteración. Se ha demostrado que el proceso converge a un mínimo local, y si está bien implementado, en unos pocos pasos. Sin embargo, el algoritmo puede no converger a un mínimo global, dependiendo de la configuración inicial. Las variaciones del algoritmo difieren en cómo se identifican los pares de puntos candidatos, en cuáles son los que se usan en el cálculo de la transformación rígida, y en el tipo de procedimiento de optimización utilizado.

Una vez que se tiene los pares de puntos correspondientes $P = \{p_1, \dots, p_n\}$, $Q = \{q_1, \dots, q_n\}$, el siguiente problema consiste en encontrar una matriz de rotación *R* y un vector de traslación *T* tales que la suma al cuadrado de las distancias entre los pares de puntos correspondientes, es decir, la cantidad *e* sea minimizada:

$$e = \sum_{i=1}^{n} \|p_i - (Rq_i + T)\|^2 \tag{11.8}$$

Hay que tener en cuenta que cuando este registro de vistas parciales por parejas se utiliza secuencialmente para obtener un modelo, los errores se pueden ir acumulando, y el registro global distar mucho de ser óptimo.

Cuando se dispone de imágenes con la textura de los objetos registrada con la información geométrica, el registro de éstas puede realizarse en la fase inicial del proceso de registro de las vistas parciales. Es particularmente ventajoso si las imágenes tienen una resolución espacial mayor que la nube de puntos o la imagen de rango, y/o el objeto tienen características de textura en áreas con pocas características geométricas. La información de textura también puede usarse en el refinamiento de la alineación inicial.

11.2.3.2. Triangulación y fusión

Para la mayoría de aplicaciones es necesario combinar las exploraciones múltiples que han sido registradas en una superficie única (malla poligonal) no redundante. En la fase de triangulación y en la de fusión es donde se realiza esta tarea, y la realización de una u otra depende del tipo de datos con el que se esté trabajando. Si los datos que se han registrado son nubes de puntos, habrá que realizar una triangulación, es decir, habrá que definir una malla que se aproxime a los puntos 3D. Si los datos de las distintas vistas parciales que se han registrado son mallas, se hará una fusión de ellas.

De entre los métodos de triangulación cabe destacar el trabajo de Hoppe *y col.* (1992) que desarrollaron un procedimiento de aproximación de una malla a un conjunto de datos 3D desorganizados usando una técnica volumétrica. En concreto, se define una función de distancia que se evalúa para los vóxeles cuya unión crea un volumen que encierra los datos. El conjunto de valores que hacen que esta distancia sea 0, es la que define la superficie. En cada uno de estos vóxeles se determina un plano que se aproxima a la superficie en ese punto y tras ello se termina definiendo los triángulos de la malla. En la figura 11.8 se muestra un ejemplo de triangulación siguiendo este método.

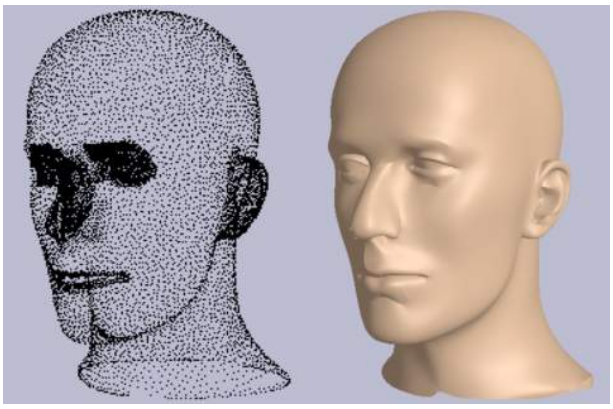


Figura 11.8. Triangulación de datos desorganizados siguiendo el método de Hoppe.

Otros procedimientos de triangulación son los basados en los métodos de Delaunay, como los α-*shapes*. Cabe por último reseñar dentro de los métodos de triangulación para datos 3D, los modelos deformables, en los que se trabaja con una malla inicial a la que se le aplica un proceso de deformación para aproximarla a los datos 3D. Esta deformación se suele realizar mediante la minimización de una función de energía. Estas técnicas son similares a los *snakes* o contornos activos empleados con imágenes de intensidad.

En el caso de que se dispongan de mallas de las distintas vistas parciales, obtenidas por algún método de triangulación o de los propios datos del sensor, habrá que proceder a realizar la fusión de las distintas mallas. Los métodos más importantes son los denominados de cremallera (o zippering en su terminología inglesa). Éstos se suelen dividir en tres etapas:

1. Eliminación de superficies redundantes, en la que se van descartando los triángulos de la frontera de una de las mallas de la vista parcial que se encuentra a una distancia menor al umbral de los triángulos de otra vista parcial con la que se solape.
2. Unión de las mallas mediante la definición de nuevos vértices y aristas en los triángulos de las fronteras obtenidas tras el paso anterior.
3. Eliminación de los triángulos despreciables, en los que se desechan los triángulos cuyos tamaños son despreciables con respecto al tamaño medio del modelo.

Redefinida la topología siguiendo los pasos anteriores, se suele aplicar algún algoritmo que redefina la posición de los vértices para conseguir una mejor aproximación a la superficie. Estos métodos producen buenos resultados, pero en las zonas donde se "cosen" las mallas suelen aparecer triángulos que en muchas situaciones no siguen el patrón del resto de la malla. En la figura 11.9 se muestra un ejemplo de una malla obtenida por una técnica de cremallera y un detalle de una zona de cosido entre las mallas.

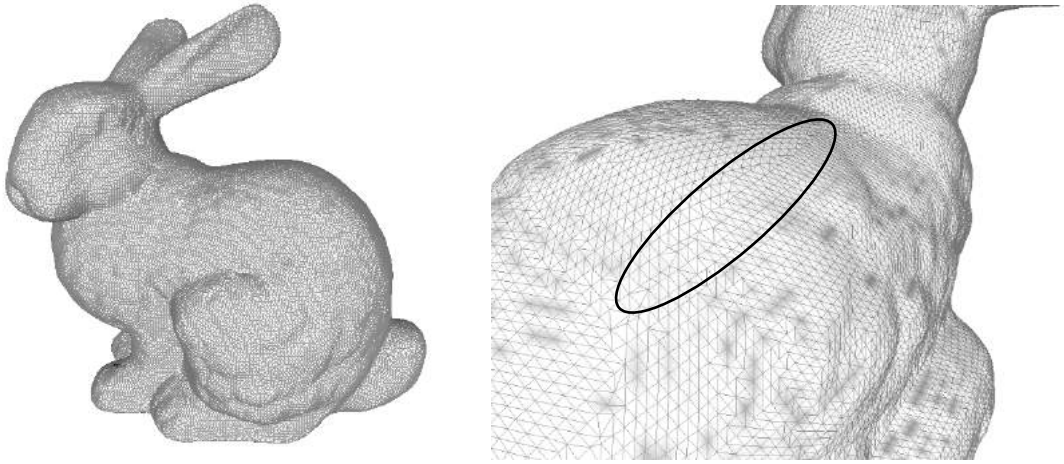


Figura 11.9. Obtención de una malla completa mediante la integración de distintas vistas parciales mediante la técnica de "cremallera" (izquierda) y detalle de la zona de cosido de las mallas (derecha).

11.2.3.3. Procesado

Las operaciones de procesado son necesarias para adecuar el modelo resultante de la fase de fusión. Estas operaciones suelen agruparse en lo que se conoce como tareas de reparación de mallas (Pérez, 2011). En general, un algoritmo de reparación de mallas toma como entrada una malla *M* que representa una superficie y produce una versión modificada, *M'*, de la misma, en la que ciertos defectos han sido eliminados.

Hay que tener en cuenta que para que sea de utilidad en la práctica, un modelo poligonal tiene que satisfacer determinados criterios según la aplicación a la que vaya a ser destinado. Los problemas más comunes que necesitan ser tratados pueden agruparse en tres grandes grupos: los relacionados con la conectividad local, aquellos que afectan a la topología global, y los que se refieren a la geometría. Los métodos propuestos para resolverlos pueden separase en métodos que se basan en aproximaciones *locales*, y modifican la malla sólo en la vecindad del defecto individual, mientras que el resto de la superficie permanece completamente inalterada, y métodos que emplean una estrategia global, que suelen basarse en un remallado completo que permite alcanzar más fácilmente robustez en la malla final. Como regla general, mallas con muchos detalles que presentan defectos aislados deben ser reparadas utilizando aproximaciones locales para preservar el mayor número de detalles posible; por el contrario, en mallas con muchos defectos o inconsistentes es conveniente utilizar aproximaciones globales. Las aproximaciones globales generalmente son altamente robustas mientras que las locales son menos invasivas.

Algunas de las operaciones más comunes que suelen acometerse en esta fase son los siguientes:

- **Eliminación de errores**:
  - *Eliminación de caras repetidas*: tras el proceso de triangulación, en ocasiones pueden aparecer caras repetidas en la estructura de triángulos de la malla. Estas caras pueden tener la misma orientación o normal, o bien ser opuestas. En cualquier caso, es un error que hay que corregir.
  - *Eliminación de triángulos no adyacentes que intersectan*: son triángulos que aparecen por una mala construcción de la malla en etapas de integración o como resultado de alguna operación

de procesado. Este defecto puede corregirse bien mediante el corte de los triángulos por la línea de intersección, con lo que se generan nuevos triángulos y se eliminan las partes redundantes, o bien eliminando uno de los triángulos o ambos.

- *Eliminación de triángulos cruzados*: un triángulo cruzado es el que tiene dos aristas compartidas con dos triángulos. Su existencia puede causar errores en los cálculos geométricos, por lo que es necesario corregir este problema. Se puede proceder a la subdivisión de los triángulos cruzados y la aplicación de una función de suavizado, o bien eliminarlos y generar nuevos triángulos si es necesario.
- *Eliminación de triángulos redundantes*: en ocasiones, debido a errores producidos en las fases de registro e integración o fusión, pueden aparecer en la malla caras redundantes. La corrección de este error se realiza mediante la división de la cara redundante, o bien mediante su eliminación.
- *Eliminación de triángulos que no producen una variedad bidimensional*: se detectan buscando aquellas aristas que son compartidas por más de tres triángulos. El proceso de corrección se lleva a cabo mediante la eliminación de una de las caras o triángulos
- *Eliminación de triángulos con normales erróneas*: en el proceso de construcción de la malla es posible que existan errores en la secuenciación de las caras en la lista $\mathcal{T}$, lo que produce que la normal de ciertas caras sea opuesta a las normales de las caras de su entorno. Para detectar este tipo de errores se ha de definir la dirección de referencia hacia fuera y hacia dentro en diversas zonas de la malla, y estudiar las normales de todas las caras. El proceso de corrección consistirá en cambiar la secuencia de vértices de las caras que no se ajustan a la dirección de referencia hacia fuera.

- **Suavizado**

El proceso de suavizado se aplica tanto de forma local como de forma global con el fin de corregir los siguientes errores:

- La aparición de picos en la superficie.
- Las desviaciones que en ocasiones pueden producirse en las zonas de solapamiento de las vistas parciales durante el proceso de registro, que se traducen en la aparición de líneas o contornos en la malla integrada, correspondiente a esas zonas de solapamiento.
- El ruido que aparece, debido principalmente a errores en el proceso de adquisición. La aparición de este ruido tiene mucho que ver con el tipo de superficie del objeto que se desea adquirir y la iluminación utilizada durante el proceso de adquisición. Los escáneres 3D basados en el uso de láser presentan ciertas limitaciones en la adquisición de superficies brillantes. Este problema se acrecienta si se utiliza una fuente de luz potente que incida sobre la superficie. El resultado es la aparición de ruido en la superficie adquirida. Para eliminarlo es necesario aplicar un suavizado con la intensidad correcta en cada caso, ya que con una intensidad baja podría no eliminarse completamente, y con una demasiado elevada, aunque si se consiguiera eliminar el ruido, sería a costa de una pérdida de características de la superficie.

- **Remallado**

Una vez finalizada la integración de las vistas parciales se obtiene una malla que, normalmente, no es regular, es decir, el tamaño de sus caras es variable. Esta característica se acentúa tras la aplicación de algunas de las operaciones de procesado descritas anteriormente.

La operación de remallado se aplica para aumentar la regularidad de la malla, sin modificar sustancialmente el número de triángulos que la forman. También puede requerirse que el tamaño de cara no sea el mismo en toda la malla y, por tanto, la regularidad se distribuya por zonas, en función del nivel de detalle requerido. Por tanto, pueden distinguirse dos tipos de remallado:

- *Remallado uniforme*: se aplica para obtener una malla regular, donde el tamaño de sus caras es aproximadamente constante.
- *Remallado adaptativo*: se aplica para obtener una malla regular por zonas, con un tamaño de cara menor cuanto mayor sea el nivel de detalle a representar.

- **Relleno de huecos**

Existen dos hechos que hacen que aparezcan huecos en una malla. El primero es que en el proceso de registro y/o integración se haya producido alguna pérdida o redefinición incorrecta de la malla, por ejemplo, algún triángulo que falte por definir. Este tipo de hueco no supone un problema, y suelen rellenarse de forma directa sin tener en cuenta la información geométrica que lo rodea. El segundo hecho que propicia la aparición de un hueco es la falta de información geométrica debido a oclusiones o sombras en los objetos. En este caso los huecos suelen ser de mayor tamaño que los anteriores. Además, cuando se trabaja con objetos de forma libre con cierto nivel de detalle, pueden encontrase huecos de formas muy complejas, y su rellenado se convierte en un problema de gran dificultad. Este grado de dificultad será mayor cuanto mayor sea la curvatura, la resolución, la profundidad de determinadas cavidades, etc.

El problema del rellenado de huecos suele dividirse en dos subproblemas:

1. La identificación de los huecos.
2. La creación de la geometría que rellene el hueco existente.

La identificación de los huecos depende del tipo de representación con que se esté trabajando. Para el caso de mallas poligonales triangulares, la identificación es relativamente sencilla. Dado que la malla está formada por triángulos que comparten cada una de sus aristas sólo con un triángulo vecino, se deduce que los triángulos que estén en el borde disponen de una arista no compartida. Partiendo de esta condición se realiza una búsqueda en la estructura de triángulos de la malla y se determinan los caminos o ciclos cerrados que forman estas aristas no compartidas, y que darán lugar a los bordes del hueco.

En cuanto a la generación de la geometría para rellenar el hueco, no es un problema trivial, dada la complejidad y diversidad de huecos que pueden presentarse, y no existe ningún método que pueda rellenar de forma automática cualquier tipo de hueco que se presente. Dependiendo de cada situación particular, debe elegirse el método de rellenado de huecos más apropiado.

11.3. Modelos de representación paramétricos

El término paramétrico es muy utilizado en el campo tecnológico y científico. En realidad un parámetro en el sentido matemático no es más que una variable que está inserta en una expresión o en una función matemática. La función depende de parámetros que pueden tomar un conjunto finito o no de valores. En este sentido, los modelos de representación 3D paramétricos están definidos de manera que dependen de parámetros de ciertas funciones o estructuras geométricas que definen el espacio (Mortenson, 2006).

Usualmente los modelos paramétricos se asocian a los modelos utilizados en diseño industrial, diseño gráfico y arquitectura (Autodesk, MicroStation, SolidWorks, Rhinoceros, etc). Sin embargo, en el campo de visión por computador 3D, la escena debe ser a menudo representada en un formato parametrizado. La generación de dichos formatos es una tarea a veces llamada de ingeniería inversa (por ejemplo, para piezas industriales u otros objetos), donde se parte de una nube de puntos desestructurada y se acaba construyendo un modelo estructurado y matemáticamente definido.

11.3.1. Modelos B-rep

Cuando la percepción del objeto es dependiente de su superficie, una representación de superficie es la representación natural. Este caso ocurre con frecuencia en los sistemas de visión por computador, donde el elemento sensorial capta información de la superficie visible del objeto. Los datos de la superficie de la escena son preprocesados, segmentados y ajustados a formas superficiales, que pueden ser formas poligonales (triángulos, hexágonos, etc.), cuádricas o superficies más complejas.

Los modelos de representación B-rep provienen del inglés “Boundary representation”, que significa representación de frontera. El modelo B-rep se basa en la descripción algebraica de los sólidos, asumiendo que están delimitados por un conjunto de caras, que pertenecen a superficies orientables y cerradas. La orientación implica que es posible distinguir la cara exterior respecto de la interior del sólido. Por su parte, la orientación suele estar indicada por el vector normal a la superficie. El modelo inalámbrico es una simplificación del modelo B-rep que contiene información solamente de aristas y nodos de una malla, no teniendo información de caras ni de normales.

Es un modelo muy popular que está relacionado con el dibujo tradicional y cuya principal ventaja es poder construir sólidos difíciles con primitivas básicas. Por el contrario, requiere mucho espacio de almacenamiento y tiempo de proceso. El objeto puede ser descrito por caras, aristas y vértices. Si el objeto posee caras curvas, éstas son aproximadas por polígonos.

En el modelo B-rep se distingue entre geometría, definida por las primitivas del modelo, y topología, que expresa la relación entre las primitivas existentes. La definición de primitivas y elementos topológicos en un modelo B-rep es el siguiente:

- Vértice (*Vertex*): punto en el espacio.
- Arista (*Edge*): curva finita limitada por dos vértices
- *Loop*: secuencia ordenada consecutiva de vértices y aristas
- Cara (*Face*): región finita limitada por uno o más loops.
- Agujero que no atraviesa (*not through hole*): depresión en un objeto

- Agujero que atraviesa (*though hole o handle*): el número de agujeros de ese tipo se llama *genus*.
- Cuerpo (*Body o Shell*): conjunto de caras que limitan un volumen cerrado continuo.

La representación B-rep más simple se realiza con una lista de caras poligonales, cada una de ellas caracterizada por una lista de coordenadas de los vértices, como se muestra en la figura 11.10. Para objetos curvos se siguen las mismas leyes, considerando aristas y superficies curvas. En este caso la representación es aproximada mediante un faceteado de la superficie.

Los modelos B-rep son utilizados en tareas de reconocimiento y posicionamiento de formas 3D de tipo poliédrico o para modelado de interiores/exteriores de edificios. Asimismo son los modelos utilizados en aplicaciones de arquitectura y construcción. También son muy útiles en aplicaciones de visión por computador para implementar algoritmos de análisis y comprensión de escenas.

Otros modelos más avanzados de B-rep pueden estar representados mediante una estructura de grafo relacional más complejo donde se introducen características del objeto y de sus partes. Así, cada nodo representa una característica o parte (primitiva) de objeto donde además puede incluirse propiedades de la misma (tamaño, forma, área, tipo, color, etc.), mientras que los arcos representan relaciones entre características (distancia entre centroides, medida de adyacencia, medida de similitud, etc.). Las primitivas ya no son polígonos sino superficies implícitas. La figura 11.10 b) muestra un ejemplo de grafo relacional de una taza, donde se han definido tres características de superficie del objeto A, B, C, etiquetadas con tres propiedades: tipo de superficie, color y área. Las relaciones tomadas son: distancia entre centroides y dos relaciones de adyacencia. Estos modelos pueden contener caras curvas que pueden estar representadas por funciones matemáticas.

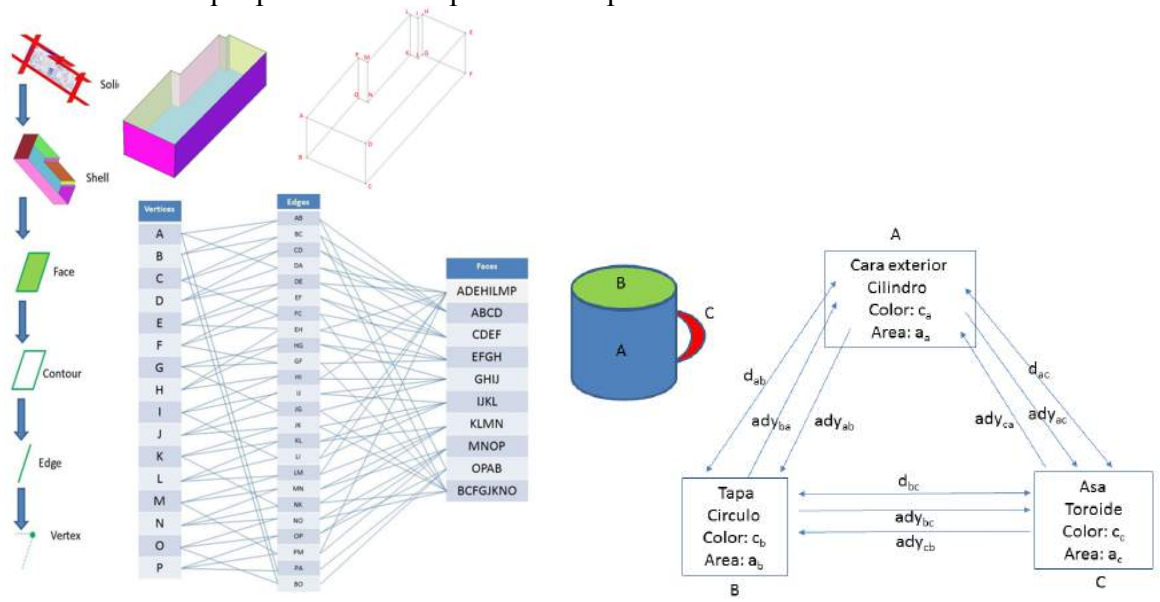


Figura 11.10. ***a)*** **Ejemplos de modelos B-rep poligonal de una habitación.** ***b)*** **Representaciones de grafo en modelos B-rep avanzados.**

11.3.2. Modelos de curvas y superficies implícitas

Los modelos de mallas y los modelos B-rep, podrían ser llamados modelos poligonales, ya que están construidos con polígonos. Existen otros modelos que utilizan ecuaciones o funciones matemáticas para representar curvas o superficies en los objetos. Esta sección se centra en modelos creados mediante superficies implícitas y explícitas.

Cuando una superficie de un objeto está dada de manera que un conjunto de puntos (x, y, z) satisfacen la ecuación $F(x, y, z) = 0$, se dice que la superficie está representada en forma implícita. Los inconvenientes genéricos de superficies implícitas son que el número de formas que modelan es limitado y que tienen problemas para ser convertidos a una representación poligonal. Ejemplos de superficies implícitas elementales son el cono, el cilindro o la esfera. Existen otros tipos se superficies implícitas que pueden representar por sí solas las superficies de una amplia variedad de formas tridimensionales.

11.3.2.1. Cuádricas

Las superficies cuádricas son primitivas matemáticas que responden a la ecuación (11.9). Dependiendo de los valores de los coeficientes *a, b, c, d, e, f ,g ,h ,j y k*, se generan las distintas cuádricas

$$ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k = 0 \qquad (11.9)$$

Los modelos de cuádricas son comunes en paquetes de modelado y diseño debido a la sencillez de uso y a su capacidad de extensión para representar sólidos de forma simple. El problema principal es que las cuadricas sólo pueden representar un conjunto limitado de las formas: elipsoides, paraboloides, hiperboloides, cilindros y conos. Algunas de estas formas sólo se encuentran en el entorno de la arquitectura. Por lo tanto, no sirven para modelar elementos naturales aunque, al menos, evitan el efecto de *aliasing*.

Figura 11.11. Representación de cuádricas y objetos modelados con cuádricas.

11.3.2.2. Supercuádricas

Las supercuádricas son un conjunto de formas paramétricas nacidas como una extensión de los elipsoides. Sus expresiones generales implícita y paramétrica son:

$$S(x,y,z)=\left[\left[\left(\frac{x}{a_1}\right)^{\frac{2}{\varepsilon_1}}+\left(\frac{y}{a_2}\right)^{\frac{2}{\varepsilon_2}}\right]^{\frac{\varepsilon_1}{\varepsilon_2}}+\left(\frac{z}{a_3}\right)^{\frac{2}{\varepsilon_2}}\right]^{\varepsilon_1}=0 \tag{11.10}$$

$$S(\mu,\omega)=\begin{bmatrix}x(\mu,\omega)\\ y(\mu,\omega)\\ z(\mu,\omega)\end{bmatrix}=\begin{bmatrix}a_1\cos^{\varepsilon_1}(\mu)\cos^{\varepsilon_2}(\omega)\\ a_2\cos^{\varepsilon_1}(\mu)\sin^{\varepsilon_2}(\omega)\\ a_3\sin^{\varepsilon_1}(\mu)\end{bmatrix},\ -\frac{\pi}{2}\le\mu\le\frac{\pi}{2},-\pi\le\omega\le\pi \tag{11.11}$$

Donde ω es el ángulo entre el eje X y la proyección del vector S en el plano XY, y μ es el ángulo entre el vector S y su proyección en el plano XY. Los valores a_1, a_2 y a_3 definen el tamaño de la supercuádrica en la coordenadas x, y, z. Por último, los valores ε_1 y ε_2 producen diversos efectos del elipsoide.

Las supercuádricas pueden adoptar funcionalidades de deformación de sólidos (adelgazamiento, torsión) y llegar a modelar una gran variedad de formas, por lo que se utilizan para modelar objetos deformables. Se usan para representar diferentes clases geométricas de objetos denominados *geones*. Sin embargo, los modelos generados son insuficientes como para modelar detalles locales diferenciadores. Por ejemplo, en el modelo de un rostro las supercuádricasno pueden proporcionar detalles de la boca o los ojos. Esto restringe la amplitud de aplicaciones en tareas de reconocimiento de formas en 3D.

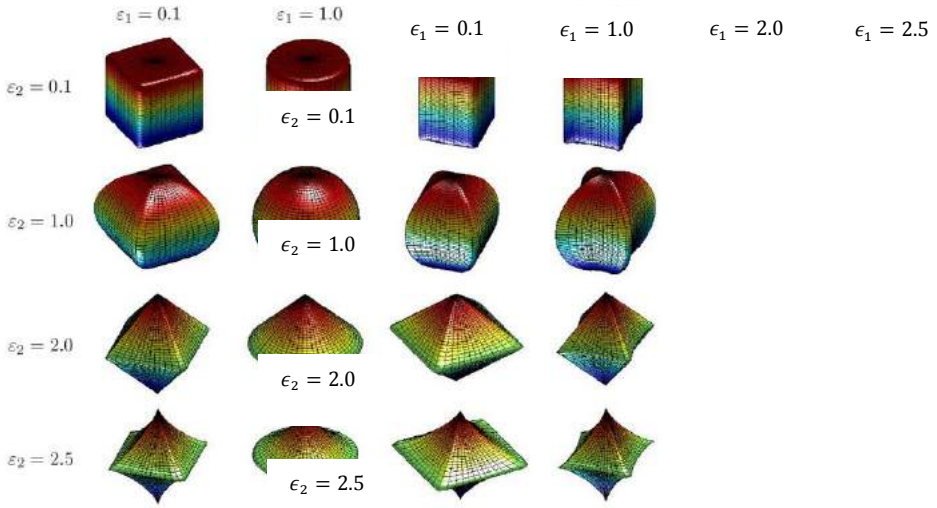


Figura 11.12. Ejemplo de supercuádricas

11.3.2.3. Isosuperficies

Las isosuperficies son también llamadas superficies equipotenciales. Son útiles para la representación de formas suaves, formas orgánicas y moleculares. En ellas se definen campos de valor

escalar en donde a cada punto del espacio le corresponde un atributo, por ejemplo campos de temperatura, campos de potencial eléctrico, campos de color, etc. Así, la superficie del objeto queda definida por el conjunto de puntos con un determinado valor del campo generando, son las llamadas superficies equipotenciales. Se debe fijar un valor umbral del campo T y entonces se obtiene una isosuperficie asociada a dicho valor umbral. Así una isosuperficie $F(x,y,z)$ asociada a un valor umbral T es una superficie en la que todos sus puntos cumplen $F(x,y,z)=T$

El potencial de campo de un *blob* en un punto del espacio *(x,y,z)* es:

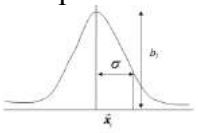
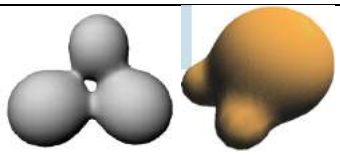
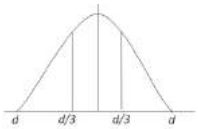
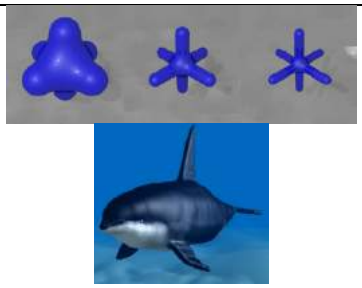
$$F(x,y,z)=\left(\sum_{i=1}^{N} b_i e^{-a_i f_i(x,y,z)}\right)-T \qquad (11.12)$$

En la expresión anterior a_i y b_i representan respectivamente la caída y la fuerza de campo de la i-ésima primitiva, T es el umbral del campo y N es el número de primitivas.

La construcción del modelo se realiza a partir de un conjunto de primitivas en el espacio que generan el campo de fuerza. Cambiando los elementos generadores del campo se genera la isosuperficie y normalmente se trabaja con campos esféricos o elipsoidales. El resultado son objetos globulares con formas abultadas, denominados *blobs.*

Como puede observarse en la ecuación 11.12, el exponente de la ecuación es una función $f(x,y,z)$. Dicha función viene definida por el modelo que estemos utilizando: Campos Gaussianos, Campos Poligonales a trozos (*Metaballs*) y Soft Objects. Según el modelo utilizado, la forma resultante tendrá un aspecto diferente. La figura 11.13 muestra las expresiones de las funciones primitivas $f(x,y,z)$ y algunos objetos generados con isosuperficies. Para el caso de los Metaballs, tomando como referencia el segmento que va del centro del modelo a un punto de estudio (x,y,z), r es la distancia del centro del modelo a la intersección y d la distancia de la intersección al punto de estudio

En general, estos modelos son muy adecuados para modelar objetos orgánicos y son muy compactos. Tienen el inconveniente de ser ineficaces en objetos con aristas o con una poligonalización muy compleja.

Campo G 	$f_i(\mathbf{x},b_i,r_i)=b_i e^{-\frac{1}{r^2}(\mathbf{x}-\mathbf{x}_i)^2}$ x_i :posición, b_i : fuerza r_i: desviación típica	
Metaballs 	$f(r)=\begin{cases} b(1-3\frac{r^2}{d^2}), & 0<r\le\frac{d}{3} \\ \frac{3}{2}b\left(1-\frac{r}{d}\right)^2, & \frac{d}{3}<r\le d \\ 0, & r>d \end{cases}$	

Soft Objects	$f(r)=\begin{cases}1-\frac{22r^2}{9d^2}+\frac{17r^4}{9d^4}-\frac{4r^6}{9d^6}, & 0<r\le d\\ 0 & r>d\end{cases}$	

Figura 11.13. Isosuperficies. Funciones de campo y modelos de objetos

11.3.3. Modelos de curvas y superficies explícitas

Las superficies explícitas son también conocidas genéricamente como superficies paramétricas. En las superficies explícitas se puede despejar una de las variables en función de las otras dos, por ejemplo, $z=f(x,y)$. Para realizar la parametrización de una superficie que esté dada de manera explícita se puede utilizar la forma genérica:

$$g(u,v)=\begin{bmatrix}u & v & f(u,v)\end{bmatrix}^T=\begin{bmatrix}x & y & z\end{bmatrix}^T,\quad a\le u\le b,\quad c\le v\le d \tag{11.13}$$

Las superficies explícitas facilitan el modelado de formas libres controlando localmente las deformaciones. En este caso, el paso a modelos poligonales es fácil y consistente. Además, proporcionan herramientas de cálculos de tangencias y curvatura de la superficie. Otra característica interesante es que ofrecen buenas prestaciones en cuanto a almacenamiento.

La superficies explícitas pueden ser creadas desde dos puntos de vista. Si son entendidas como superficies de interpolación, se crean a través de un conjunto de puntos de control por los que se quiere que pase la superficie explícita. Por otro lado, si no se exige que se pase por los puntos que controlan la forma de la superficie, se dice que se tiene una superficie de aproximación.

En el campo de visión por computador las técnicas de aproximación son especialmente útiles para obtener modelos paramétricos de superficies que han sido obtenidas a través de sensores tridimensionales realizando procesos de ingeniería inversa. Estos modelos pueden ser generados por herramientas CAD para crear prototipos o piezas físicas del objeto.

Las superficies de aproximación son una extensión a dos parámetros de las curvas de aproximación conocidas como splines, curvas de Bézier, B-splines y NURBS. Definiremos de forma sucinta el significado de cada una de éstas.

Como se sabe, la aproximación básica de una curva se puede realizar con funciones polinómicas, de modo que una curva de *n* puntos sería aproximada por un polinomio de grado *n.* Ya que esta opción presentaría problemas si el número de puntos fuese alto, existe otra posibilidad de definir una curva como combinación de trozos de curvas más simples. Así, una buena solución es componer una curva con *n*-1 trozos de grado 3, donde *n* es el número de puntos por los que pasa la curva. Este tipo de curvas se conocen como *splines*. La ecuación matemática de un spline es:

$$C_j(u)=\sum_{i=0}^{3}a_{ij}u^i \tag{11.14}$$

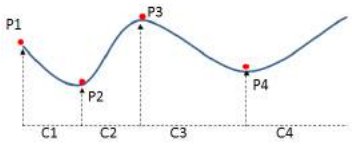
donde el parámetro *u* se evalúa en el intervalo [0,1] y j se refiere al *j*-ésimo trozo de grado 3.

Las curvas de Bézier son una variación de las anteriores. Solamente pasan por los puntos de control extremos y se generan a través de los llamados polinomios de Berstein. Tienen el problema de que los polinomios de Berstien están sujetos al número específico de puntos de control, lo que resta capacidad de aproximación en casos de formas complejas. Esta dificultad la salvan los llamados B-splines. En los B-splines, se realiza una unión del espacio paramétrico de los diferentes trozos a través de un vector de nudos (*knots*). Dependiendo de la distribución del vector de *knots* tendremos distintas familias de B-Splines. Finalmente, cuando podemos tener cualquier distribución del vector de *knots* se generan los llamados NUBS (*Non Uniform Rational BSplines*). En este caso se incorpora un conjunto de pesos que controlan cómo la curva se aproxima a los puntos de control.

Curvas Splines

$$C_j(u) = \sum_{i=0}^{3} a_{ij} u^i$$

C_j: curva j-ésima de grado 3



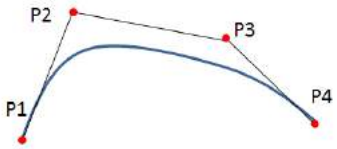
Curvas Bezier

$$C(u) = \sum_{i=0}^{n} B_{i,n}(u) P_i \qquad 0 \le u \le 1$$

$$B_{i,n}(u) = \frac{n!}{i!(n-1)!} u^i (1-u)^{n-i}$$

$$\sum_{i=0}^{n} B_{i,n}(u) = 1 \forall u$$

$B_{i,n}$: polinomios de Berstein de grado n
P_i: puntos de control

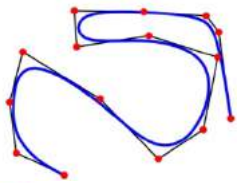


Curvas B-splines

$$C(u) = \sum_{i=0}^{n} N_{i,d}(u) P_i$$

$$u_{min} < u < u_{max} \qquad 2 \le d \le n+1$$

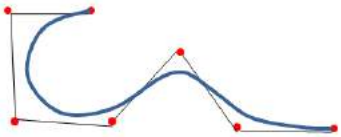
P_i: puntos de control
$N_{i,d}$: funciones base



Curvas NURBS

$$C(u) = \frac{\sum_{i=0}^{n} (w_i P_i N_{i,k}(u))}{\sum_{i=0}^{n} (w_i N_{i,k}(u))}$$

w_i: pesos
P_i: puntos de control
$N_{i,k}$: funciones B-spline de grado k

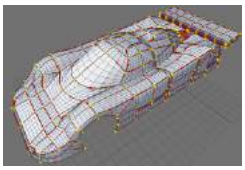


Si extendemos lo dicho en una dimensión a dos dimensiones, las curvas de aproximación pasan a ser respectivamente superficies de Bézier, superficies B-splines o superficies NUBS.

Superficies B-splines

$$S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,K}(u) N_{l,L}(v) P_{ij}$$

P_{ij}: puntos de control
$N_{i,K}$ $N_{j,L}$ Funciones base de grado K, L



Superficies NURBS

$$S(u,v) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} h_{ij} N_{i,K}(u) N_{l,L}(v) P_{ij}}{\sum_{i=0}^{m} \sum_{j=0}^{n} h_{ij} N_{i,K}(u) N_{l,L}(v)}$$

P_{ij}: puntos de control
$N_{i,K}$ $N_{j,L}$ Funciones base de grado K, L
h_{ij}: Peso asociado al punto P_{ij}

Los modelos NURBS se definen por su grado, un conjunto de puntos de control (P_{ij}) ponderados (h_{ij}), y un vector de nodos (N). Su principal ventaja, muy ligada al campo de la visión por computador, es su invarianza a transformaciones afines y de perspectiva. Sólo los puntos de control tienen que ser transformados para obtener la transformación apropiada de la forma NURBS.

En la práctica, cuando se tiene una nube de puntos de un objeto, la generación de superficies continuas que modelen el objeto se realiza habitualmente a través de NURBS. Por lo tanto es un modelo muy importante que forma parte de los paquetes estándar en la mayoría de los programas gráficos por computadora, software de escáneres y programas de ingeniería inversa, siendo las superficies más ampliamente usadas en los procesos de diseño industrial.

11.3.4. Modelos CSG

En los modelos CSG (Constructive Solid Geometry) un conjunto de primitivas de volumen se combinan a través de operadores booleanos de conjuntos. Están basados en la idea de que todo objeto puede ser descompuesto en un conjunto de primitivas combinadas.
Es un modelo popular, ya que simula el proceso natural de diseño añadiendo volúmenes elementales. Fácil de construir, visualizar, almacenar y validar, pero tiene ciertas limitaciones de representación.

En visión por computador existen algoritmos que pretenden identificar objetos a través de una segmentación previa de las partes constituyentes del objeto. Esta tarea se puede realizar a través de imágenes 2D o con imágenes de profundidad RGB-D. En estos casos, la representación con modelos CSG es la adecuada. Cada parte segmentada en la escena puede ser ajustada a una primitiva que conforme el objeto. Con todo, este tipo de modelos son mayoritariamente utilizados en gráficos por computador y para realizar ingeniería directa.

El modelo CSG contiene información topológica, a través del conjunto de operaciones sobre las primitivas, e información geométrica, con la posición y transformación de las primitivas. Las primitivas básicas de los modelos CSG son: prisma, cilindro, esfera, cono, cuña y toro. Como operaciones básicas podemos considerar la unión, intersección y diferencia
Los modelos B-rep son a menudo considerados como modelos superficiales de geometría constructiva. La diferencia principal entre el modelo CSG y los modelos B-rep radica en que el modelo CSG no almacena explícitamente caras, aristas y vértices, aunque pueden ser evaluadas cuando se desee. Además, el concepto de primitiva se extiende al proceso de diseño.

La estructura de datos utilizada en los modelos CSG puede estar basada en grafos o en árboles binarios. La representación mediante grafos es una representación sintetizada y eficiente para editar el modelo, pero es ineficiente para realizar cálculos geométricos. Por su parte, la representación en árbol contiene las primitivas en sus hojas y las operaciones de primitivas en sus nodos. Así, para *n* primitivas, se tienen *n-1* nodos interiores.

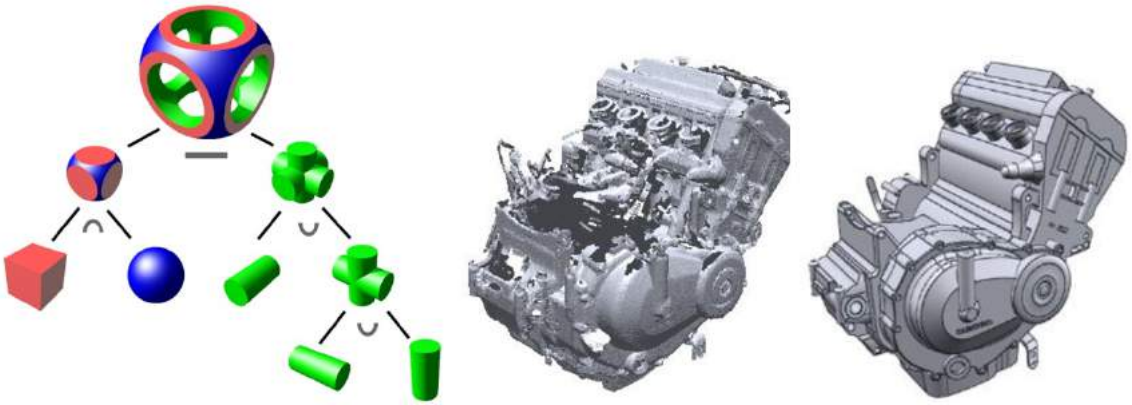


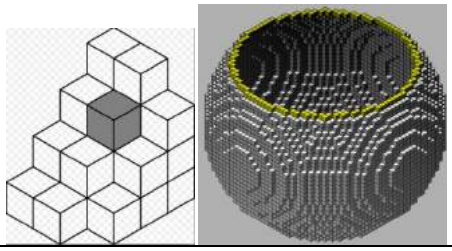
Figura 11.14. Modelos CSG. Ejemplos de representación en árbol binario y reconstrucción mediante modelos CSG.

11.3.5. Modelos de partición espacial

Los modelos de partición espacial dividen el espacio en un conjunto de celdas elementales llamadas *voxel* (contracción de las palabras inglesas "elemento de volumen") cuya yuxtaposición llena todo el espacio ocupado por el objeto. Para obtener la información de un objeto no hay más que estudiar si las celdas están ocupadas (total o parcialmente) o vacías. En función del grado de ocupación de las celdas, los métodos de ocupación espacial se diferencian en la forma de dividir el espacio y en el tratamiento de celdas parcialmente ocupadas. Principalmente tenemos dos tipos de modelos:

- *Modelos de partición regular*. En ellos se realiza una enumeración regular de ocupación espacial considerando celdas del mismo tamaño y orientación. El tamaño puede ser pequeño pero, si aumenta el número de celdas que componen la malla, también aumentará el espacio de almacenamiento y procesamiento.
- *Modelos de partición jerárquica*. Utilizan las estructuras quadtree/octree, que realizan una subdivisión recursiva del espacio en cuadrados/cubos de tamaño menor, dependiendo de la ocupación o no de ese elemento. Esta variante jerárquica de enumeración de ocupación espacial es diseñada para optimizar requisitos de almacenamiento y tiempo de cómputo.

Los modelos de partición espacial son especialmente útiles en aplicaciones de reparación de objetos (por ejemplo, rellenado de huecos) y generación de modelos 3D de escenas de gran dimensión. El procesamiento eficiente de nubes de puntos obtenidos a través de escáneres de larga distancia se realiza discretizando el espacio mediante modelos voxelizados regulares o con octrees. Estos modelos permiten realizar tareas de segmentación, reconocimiento y posicionamiento de componentes estructurales de la escena, como paredes, suelo o techo.

Voxel	Octree
	

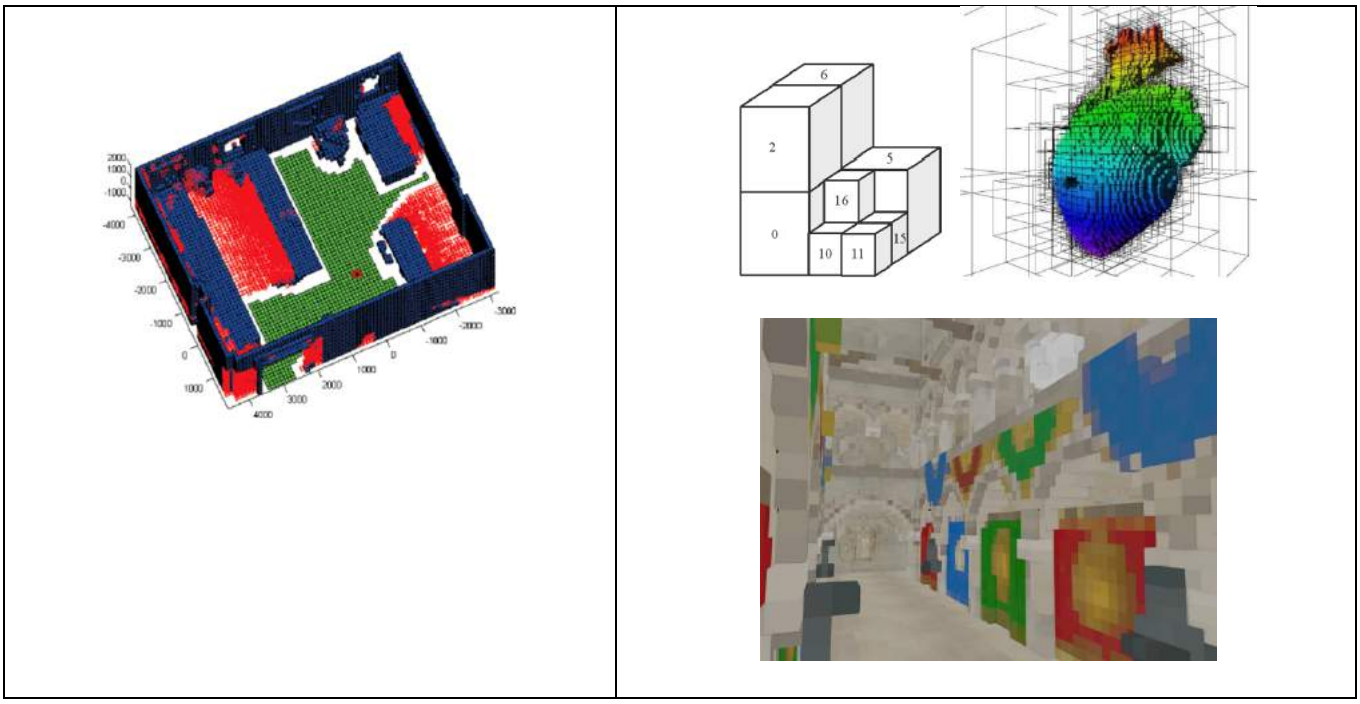


Figura 11.15. Modelos de partición espacial

Existen otros modelos de representación de objetos más propios de graficos y escasamente utilizados en visión por computador. Algunos de estos son: modelos de barrido, cilindros generalizados, modelos de partículas y modelos fractales.

11.4. Aplicaciones

Es difícil encontrar en la actualidad un campo de conocimiento o una actividad cotidiana en la que no se empleen modelos digitales 3D. Su uso se ha generalizado y sus aplicaciones exceden con mucho aquellas de *visión por computador* y *gráficas por ordenador* (Leonardis y col., 2000). Se exponen a continuación algunas de las más relevantes:

Industria

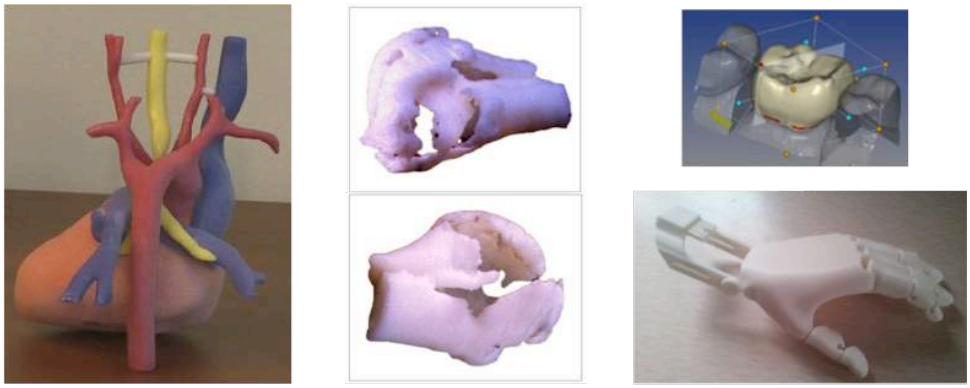
Las aplicaciones de los modelos 3D en la industria son numerosísimas, pero quizás haya que reseñar dos como las más importantes.

Las primeras son todas las relacionadas con las técnicas de *ingeniería inversa* para fabricación. En este caso se emplean escáneres 3D para la obtención de los datos y se siguen los pasos para la creación de las mallas triangulares que suelen ser almacenadas en ficheros *stl*. Dependiendo de la aplicación, también se pueden obtener modelos paramétricos, como superficies de Bézier. En cualquier caso, los modelos suelen ser usados para tener registros o para servir como ficheros de entrada a equipos de fabricación.

Otra de las aplicaciones típicas del ámbito de la industria es el *control de calidad* e *inspección*. En este caso, los modelos 3D obtenidos de forma similar a la anterior se emplean para ser comparados con modelos sintéticos 3D de la misma pieza, permitiendo detectar los posibles fallos de fabricación.

Medicina

La Medicina es uno de los campos al que más beneficios ha reportado el desarrollo y la generalización del uso de modelos 3D. Los modelos obtenidos, tan cercanos a la realidad, sirven como excelente herramienta de visualización y aprendizaje, favorecen la comunicación entre profesionales, permiten que los procedimientos quirúrgicos pueden ser planificados y simulados con antelación para minimizar el riesgo de complicaciones en casos de extrema dificultad y posibilitan el diseño de implantes y prótesis a medida.



(a) (b) (c)

Figura 11.16. *a)* Modelo del corazón de un bebé de 20 meses. *b)* parte superior del hueso del hombro fracturado en 4 piezas. L aproximación quirúrgica, los pasos para su reducción, la colocación de los implantes y la necesidad o no de injerto óseo pueden ser ahora planificados con antelación utilizando un modelo 3D. *c)* A 3D model used in digital dentistry

Construcción

El campo de la construcción tampoco es ajeno a la utilidad de estos modelos (Patraucean y col., 2015). Su uso, tanto en las etapas de diseño como en la construcción propiamente dicha, permite reducir considerablemente los tiempos de producción y, sobre todo, los costes, ya que permiten realizar pedidos de material más precisos a la vez que se reduce la cantidad de trabajo rechazado

Los modelos 3D también pueden ser utilizados como fuente de mejora de la sostenibilidad de las edificaciones, puesto que facilitan la obtención de información sobre la iluminación, los sistemas de calefacción y refrigeración, y permiten identificar problemas y posibles mejoras de la eficiencia energética. Asociados a cámaras térmicas, posibilitarían la localización de áreas de grandes pérdidas que precisarían de un mejor aislamiento.

Patrimonio

La aplicabilidad de los modelos digitales al patrimonio cultural es incuestionable y se ha visto reflejada en multitud de proyectos y colaboraciones multidisciplinares que han tenido lugar en los últimos años. Más allá de suponer una novedosa y muy eficiente herramienta de visualización, los modelos 3D han demostrado ser de gran utilidad en tareas que pueden agruparse en dos grupos diferenciados:

• *El estudio de obras de arte*. Las réplicas digitales permiten el diseño de nuevos procesos para la realización de investigaciones específicas sobre ellas. La disponibilidad de estas réplicas, junto con el desarrollo de metodologías innovadoras, basadas en análisis de forma, conducen a la generación de nuevos conocimientos y puntos de vista,figura 11.17.

• *La creación de archivos y repositorios digitales***.** Permiten preservar, indexar, recuperar, visualizar, compartir y difundir el patrimonio cultural, figura 11.18.

(a) (b) (c)

Figura 11.17 ***a)*** **Réplicas en cera obtenidas a partir de los modelos digitalizados.** ***b)*** **Propuesta de restitución del grupo de Eneas.** ***c)*****Réplica a escala de la restitución virtual**

(a) (b)

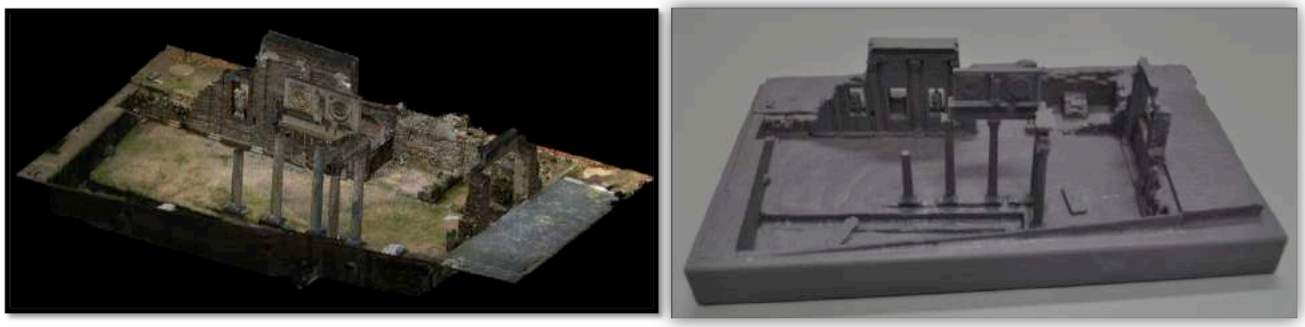


Figura 11.18 ***a)*** **Modelo del Pórtico del Foro (Mérida).** ***b)*** **Réplica en poliresina**

11.5. Bibliografía

Brutzman, D.; Daly, L. (2007) X3D: Extensible 3D Graphics for Web Authors, 1ª ed,; Morgan Kaufmann Publishers Inc.: San Francisco, USA.

Bourke, P. (2009) Ply - Polygon File Format. Disponible online: http://paulbourke.net/dataformats/ply/ (accedido 8 de octubre de 2015).

Campbell, R.J.; Flynn, P.J. (2001) A Survey of Free Free-Form Object Representation and Recognition Techniques Techniques. *Computer Vision and Image Undestanding*, *81*, 166–210.

Hon Wah, W. (1999) Introduction to STL Format. Disponible online: http://download.novedge.com/Brands/FPS/Documents/Introduction_To_STL_File_Format.pdf (accedido 8 de octubre de 2015).

Leonardis, A.; Solina, F.; Bajcsy, R. (2000) Confluence of computer vision and computer graphics; Kluwer.

Mortenson, M. (2006) Geometric Modeling, 3ª ed.; Industrial Press Inc., USA.

Murray, J.; Van Ryper, W. (2005). Wavefront OBJ File Format Summary. Disponible online: http://www.fileformat.info/format/wavefrontobj/egff.htm (accedido 8 de octubre de 2015).

Patraucean, V.; Armenia, I.; Nahangib, M.; Yeungb, J.; Brilakisa, I.; Haas, C. (2015) State of research in automatic as-built modeling. *Advanced Engineering Informatics, 29(2)*, 162–171.

Pérez, E. (2011) Tecnicas de Relleno de Superficie Adquiridas Mediante Escáneres 3D. , Universidad Nacional de Educación a Distancia, Madrid.

Rusinkiewicz, S.; Levoy, M. (2001) Efficient Variants of the ICP Algorithm. Proc. 3-D Digital Imaging and Modeling, Quebec City, Canada, pp. 145-152.

Salamanca, S. (2005) Modelado Tridimensional de Vistas Parciales para Objetos con Forma Libre Orientado a Tareas de Visión por Computador, Universidad Nacional de Educación a Distancia, Madrid.

Tanga, P.; Huberb, D.; Akincic, B.; Lipmand, R.;Lytlee, A. (2010) Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques. *Automation in Construction, 19(7),* 829 – 843.

CAPÍTULO 12

VISIÓN 3D: MODELO DE CÁMARA

Antonio J. SÁNCHEZ-SALMERÓN[1] , Eugenio IVORRA-MARTÍNEZ[1]

1 Instituto de Automática e Informática Industrial, Universitat Politècnica de València, Valencia, España

El propósito de este capítulo es introducir y proporcionar una sólida base sobre la temática de la visión tridimensional basada en técnicas geométricas proyectivas.

Una imagen es la proyección del mundo real (3D) sobre un subespacio bidimensional (2D). Para poder recuperar, a partir de una imagen, la dimensión perdida al proyectar hace falta conocer la relación entre las coordenadas de la imagen y la escena 3D del mundo definiendo un modelo de cámara, concretamente en este capítulo se ha empleado el modelo de cámara de orificio o *pinhole*. Los parámetros que definen este modelo de cámara se estiman realizando una calibración de la misma. Aunque existen muchas técnicas de calibración, en este capítulo se explica una de las más empleadas que consiste en relacionar puntos 3D de un patrón conocido con sus correspondientes coordenadas en imagen (2D) mediante un sistema de ecuaciones que se resuelven utilizando el algoritmo *Direct Linear Transformation*. Una vez conocido el modelo de cámara hace falta establecer una serie de restricciones adicionales que resuelvan de forma inequívoca el problema de reconstrucción 3D a partir del 2D. En este capítulo las restricciones que se utilizan están basadas en homografías que restringen la escena tridimensional a un plano.

12.1. Introducción

Los primeros fundamentos de la visión tridimensional datan entre el siglo IV y el siglo III a.C. cuando el geómetra alejandrino Euclides describió observaciones geométricas tan importantes como la propagación rectilínea de la luz en sus escritos titulados "Óptica" y "Catróptica". Entre los años 99 y 55 a. C. Lucrecio describe la ley de la reflexión de la luz en su libro "De la naturaleza de las cosas" donde se dice claramente que el ángulo de incidencia es igual al ángulo de reflexión. Fue durante los años 3 y 65 d.C. cuando Séneca mencionó la capacidad amplificadora de las lentes convergentes al describir cómo se veían las cosas a través de un globo de vidrio lleno de agua. Posteriormente, Alhazen, científico árabe (965-1040 d.C.) describió el comportamiento de la luz y la óptica mediante la cámara oscura (equivalente al modelo de cámara *pinhole*) en su obra "Kitabal Manadhi" (Perspectiva). En el siglo XIII, el científico inglés Roger Bacon describió claramente las propiedades de una lente y talló las primeras lentes en forma de lenteja que permitió el desarrollo posterior de los telescopios. En 1569, Daniel Barbaro (1528-1569) introdujo el uso del diafragma para controlar la apertura y la cantidad de luz que pasaba a través de la lente. Finalmente, fue en el siglo XIX cuando Niepce en 1826 y Daguerre en 1839 lograron fijar las primeras imágenes denominadas fotografías. Posteriormente, en el mismo siglo, se empezó a usar el término *fotogrametría* por manos de Meydenbauer (1867) y Laussedat (1898), término que aun actualmente se encuentra vigente. *Fotogrametría* significa literalmente 'medir de fotos' y se trata de una técnica para determinar las propiedades geométricas de objetos en escenas tridimensionales a partir de imágenes fotográficas.

12.1.1 ¿Cómo inferir información 3D a partir de una imagen 2D?

Una imagen se forma mediante la proyección del mundo tridimensional al subespacio bidimensional de la imagen, perdiendo en este proceso de captación de la imagen la dimensión de profundidad en la escena. Esta dimensión no se puede recuperar a partir de una sola imagen. Para poder recuperar dicha información, además de utilizar la relación de proyección existente entre las coordenadas de la imagen y las de la escena 3D del mundo, se requiere añadir restricciones adicionales (Faugeras, 1993). En este capítulo se ha optado por determinar cuáles son estas relaciones mediante técnicas geométricas proyectivas y además se restringe la escena tridimensional a un plano para poder recuperar la información 3D a partir de una sola imagen.

12.1.2 Sistemas comerciales y aplicaciones

Alrededor del 90% de los sistemas comerciales en uso en la industria son sensores basados en luz estructurada. Estos sensores constan fundamentalmente de tres elementos: un patrón de luz, un sistema de detección o cámara equipado con un sensor CCD y un sistema de procesado de datos. La fuente de luz es la encargada de emitir un patrón de luz sobre la escena que provocará una deformación en el patrón que será medido por una cámara con CCD. La información 3D se obtiene de la deformación de

este patrón basándose en trigonometría (Peiravi y Taabbodi, 2010). Existen muchos tipos de patrones desde el más sencillo que es un punto láser a otros más complejos como la emisión de múltiples patrones simultáneos de franjas de frecuencias sinusoidales (Wang y Zhang, 2010).

En la figura 12.1 se muestra un ejemplo de un patrón basado en la proyección de un plano láser que genera una reflexión de los puntos de la escena que pertenecen a este plano. La posición 3D de los puntos reflejados se puede obtener mediante la estimación de una homografía o también conociendo la distancia entre el láser y la cámara (llamada línea base) y el ángulo entre dicha línea y el rayo del láser. Las coordenadas 3D *(x,y,z)* de un punto del objeto sobre el que se proyecta el láser se pueden calcular conociendo el píxel *(x',y')* sobre el que proyecta usando la siguiente ecuación (12.1):

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{1}{f \cot \alpha - x'} \begin{bmatrix} bx' \\ by' \\ bf \end{bmatrix} \tag{12.1}$$

Donde *f* es la distancia focal, *b* es la línea base y α es el ángulo entre el plano del rayo láser y el eje x.

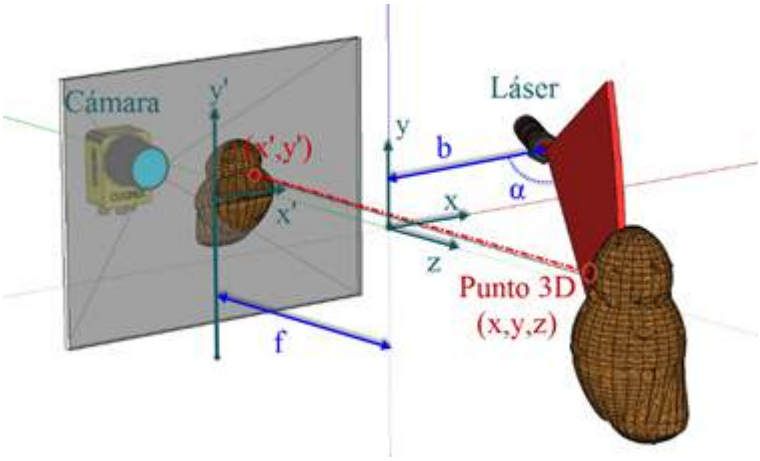


Figura 12.1. Esquema de un sistema de captura 3D basado en luz estructurada

En el libro de Wöhler (2009) "An introduction to 3D Computer Vision Techniques and Algorithms" se pueden encontrar muchos escenarios de aplicación diferentes. Algunas aplicaciones típicas son el guiado de robots, el control de procesos, la clasificación o el reconocimiento de objetos e incluso el reconocimiento de acciones. El guiado de robots consiste en controlar las trayectorias de un brazo robot para que éste pueda manipular los productos que se encuentran sobre el plano de trabajo (generalmente los productos se transportan sobre la superficie plana de una cinta transportadora) (Sánchez y Martínez, 2000). Por otra parte, un ejemplo de aplicación al reconocimiento de objetos se presenta en Sánchez y Marchant (2000) donde la información 3D generada por una cámara en movimiento permite clasificar las malezas en cultivos. Finalmente, un ejemplo de reconocimiento de acciones se describe en los trabajos de Bosch y col. (2012, 2014) donde se utilizan descriptores 3D

obtenidos a partir de una cámara de gran angular y la restricción planar del suelo de una habitación para detectar caídas de personas mayores.

12.2 Modelos de distorsión de la lente

Las lentes, generalmente de vidrio, son elementos ópticos montados dentro de la óptica de la cámara que se utilizan para enfocar la trayectoria de los rayos luminosos sobre el sensor y poder formar imágenes bien enfocadas con diámetros de apertura grandes para capturar suficiente cantidad de luz con menos tiempo de exposición. En teoría, si las lentes fueran perfectas no introducirían ningún tipo de deformación en la imagen resultante, sin embargo en la práctica se producen diferentes tipos de deformaciones. Estas distorsiones se producen principalmente por dos motivos: porque la lente no esté bien alineada con el sensor y porque la lente no tenga una forma parabólica perfecta. Las ópticas de distancia focal pequeña, de alrededor de los 3 milímetros, se denominan ópticas de gran angular. Estas ópticas presentan un amplio ángulo de visión pero también la denominada distorsión *radial* que se caracteriza porque se pierde nitidez en los bordes de la imagen y se produce el llamado efecto barril que consiste en que las líneas rectas se visualizan con mayor curvatura a medida que nos alejamos del punto principal de la imagen. La distorsión debida a los fallos mecánicos de alineación de los componentes de las cámaras genera la denominada distorsión *tangencial* y provoca que rectas que deberían proyectar como paralelas en la imagen no lo sean (Bradski y Kaehler, 2008).
En la figura 12.2 se puede observar varias imágenes tomadas desde un punto de vista cenital. En la imagen 12.2(a) no se aprecian distorsiones, sin embargo en la figura 12.2(b) se observa distorsión radial y la figura 12.2(c) presenta distorsión tangencial.

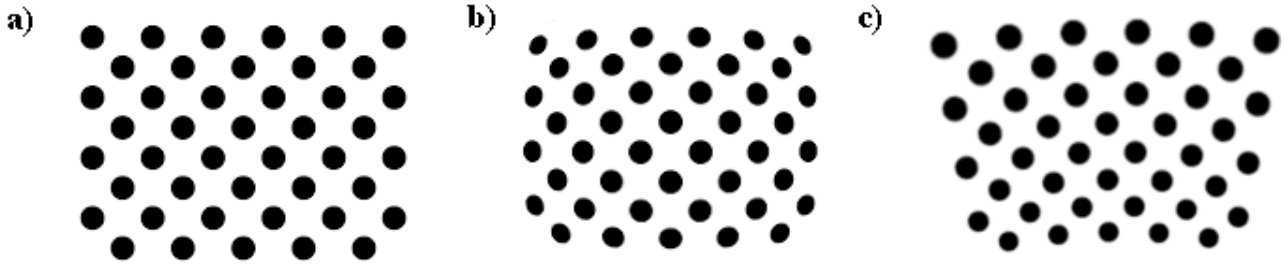


Figura 12.2. a) Imagen original, b) imagen a con distorsión radial, c) imagen a con distorsión tangencial.

El tipo de distorsión radial presentada en la figura 12.2.b se denomina de barril porque muestra una distorsión nula en el centro de la imagen (específicamente es de 0 en el punto principal) pero la distorsión se va incrementando conforme más cerca se está de los bordes de la imagen. Este fenómeno se produce en ópticas de gran angular y también en cámaras económicas debido a que es más barato producir lentes esféricas que parabólicas. También se puede dar el efecto contrario en la curvatura de las rectas periféricas provocando una distorsión radial denominada de cojín. Para modelar la distorsión radial, y por tanto poder corregir este tipo de problemas, existen muchos modelos de distorsión

(Ricolfe y Sánchez, 2010), aunque se suele emplear los tres primeros términos de la serie de Taylor calculada alrededor del punto principal **0**.
La serie matemática de Taylor se emplea para modelar de forma aproximada funciones complejas alrededor de un punto de forma que con cada nuevo término polinomial la aproximación es mejor. En este caso, se expande la función de Taylor polinomial de la siguiente forma: $f(r) = a_0 + a_1 r_1 + a_2 r_2^2 + ... + a_6 r_6^6$. Debido a que se calcula en el punto principal **0**, $f(r) = 0; r = 0 \rightarrow a_0 = 0$ y al ser una función simétrica, sólo los coeficientes pares 2, 4 y 6 son necesarios (a_2, a_4 y a_6) a los cuales llamaremos respectivamente k_0, k_1 y k_2. Una vez calculados estos coeficientes, es posible corregir la distorsión radial siguiendo la ecuación (12.2) donde $(u',v')^T$ son las coordenadas de los píxeles corregidos y $(u,v)^T$ las de los píxeles de la imagen distorsionada.

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \left(1 + k_0 r^2 + k_1 r^4 + k_2 r^6\right) \begin{bmatrix} u \\ v \end{bmatrix} \tag{12.2}$$

En la figura 12.2(c) se puede observar una distorsión tangencial importante donde los puntos originales que formaban columnas paralelas, figura 12.2(a) dejan de ser paralelas e incluso tienen un menor tamaño. Esta distorsión se puede corregir mediante la siguiente ecuación 12.3 conociendo los parámetros d_0 y d_1.

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} + d_0 \begin{bmatrix} 2v \\ r^2 + 2v^2 \end{bmatrix} + d_1 \begin{bmatrix} r^2 + 2u^2 \\ 2u \end{bmatrix} \tag{12.3}$$

Tanto en la ecuación 12.2 como en la 12.3 los parámetros desconocidos se pueden estimar mediante técnicas de calibración.

12.3 Modelo proyectivo de la cámara

Dado un sistema de referencia arbitrario del mundo 3D, W={$\mathbf{o}_w$, $\mathbf{x}_w$, $\mathbf{y}_w$, $\mathbf{z}_w$}, y un sistema de referencia 2D de la imagen D={$\mathbf{o}_d$, $\mathbf{u}_d$, $\mathbf{v}_d$}, donde el origen $\mathbf{o}_d$ está situado en la esquina superior izquierda de la imagen, el modelo de cámara se define como una matriz de proyección de perspectiva **M**, de dimensiones 3x4, que relaciona los puntos 3D del mundo ${}^w\mathbf{p}$ = $[X,Y,Z,1]^T$ definidos con respecto a W con sus respectivos píxeles en la imagen ${}^d\mathbf{q}$ = $[u,v,1]^T$ definidos respecto a D, de tal manera que ${}^d\mathbf{q} = {}^d\mathbf{M}_w \cdot {}^w\mathbf{p}$.
La matriz de proyección perspectiva **M** se puede descomponer en el producto de las siguientes tres matrices de transformación:

$${}^d\mathbf{M}_w = {}^d\mathbf{K}_p \cdot {}^p\mathbf{P}_c \cdot {}^c\mathbf{T}_w \tag{12.4}$$

En esta ecuación aparecen cuatro sistemas de referencia W, C, P y D que se localizan estratégicamente de la siguiente forma para simplificar al máximo la expresión matemática obtenida.

C={**o**, $\mathbf{x}_c$, $\mathbf{y}_c$, $\mathbf{z}_c$}, sistema de referencia de la cámara que tiene el origen en el centro óptico **o** y el eje $\mathbf{z}_c$ coincide con el eje óptico de la cámara.
P={**0**, $\mathbf{u}_p$, $\mathbf{v}_p$}, sistema de referencia asociado al plano de imagen, cuyo origen está situado en el punto principal **0** de la imagen y cuya orientación de sus ejes coinciden con la orientación de los ejes de C, es decir $\mathbf{u}_p = \mathbf{x}_c$; $\mathbf{v}_p = \mathbf{y}_c$.
D={$\mathbf{o}_d$, $\mathbf{u}_d$, $\mathbf{v}_d$}, sistema de referencia cuya orientación de sus ejes coinciden con la orientación de los ejes de P, es decir $\mathbf{u}_d = \mathbf{u}_p$; $\mathbf{v}_d = \mathbf{v}_p$.

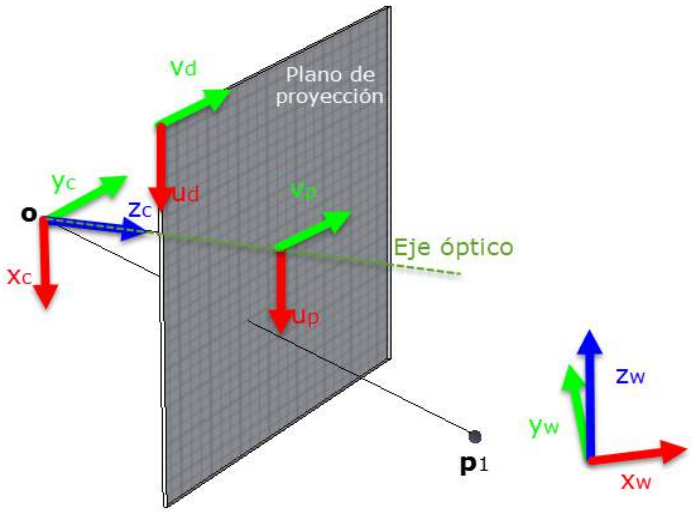


Figura 12.3. Localización de los sistemas de referencia del modelo de cámara.

12.3.1. Parámetros extrínsecos del modelo de cámara (matriz T)

Dado el sistema de referencia W de los puntos 3D del mundo ${}^w\mathbf{p}=[X,Y,Z,1]^T$ y el sistema de referencia C asociado a la cámara, cuyo origen es el centro óptico **o** y el eje $\mathbf{z}_c$ es el eje óptico de la cámara, la matriz de transformación homogénea **T** es una matriz 4x4 que define la localización del sistema de coordenadas del mundo W con respecto al sistema de coordenadas de la cámara C, de tal manera que ${}^c\mathbf{q} = {}^c\mathbf{T}_w \cdot {}^w\mathbf{p}$, es decir, obtenemos las coordenadas del punto ${}^c\mathbf{q}$ expresado con respecto a C a partir del mismo punto ${}^w\mathbf{p}$, expresado con respecto a W, aplicando la matriz de transformación **T**.

$$\mathbf{T} = {}^c\mathbf{T}_w = \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{t}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12.5}$$

donde **R**, es la matriz de rotación 3x3, que define la orientación de W con respecto a C; **t** es el vector columna de traslación 3x1 que define dónde está el origen de W con respecto a C y **0** es el vector fila 1x3 nulo (no confundir con el punto principal).
El centro óptico de la cámara ${}^w\mathbf{o}$ expresado respecto a W se puede calcular a partir de **T**, aplicando la siguiente ecuación $w_o = -\mathbf{R}^T \cdot \mathbf{t}$.

12.3.2. Proyección de perspectiva (matriz P)

El modelo proyectivo de perspectiva más básico empleado para calcular la proyección de la información 3D del mundo al plano de imagen es el modelo de cámara de orificio invertido también llamado modelo "*Pinhole invertido*" (Cyganek y Siebert, 2009). Este modelo simula cómo la luz entra en una caja oscura (el cuerpo de la cámara) a través de un agujero muy pequeño, que deja pasar únicamente un rayo de luz por punto del espacio 3D y que son proyectados sobre el sensor que forma la imagen. Tal y como se puede ver en la figura 12.4 la información 3D del mundo se proyecta en el plano de la imagen a través del centro óptico **o** situado a una distancia igual a la distancia focal *f* del plano de la imagen. El eje óptico es la línea perpendicular al plano de la imagen y que pasa por el centro óptico.

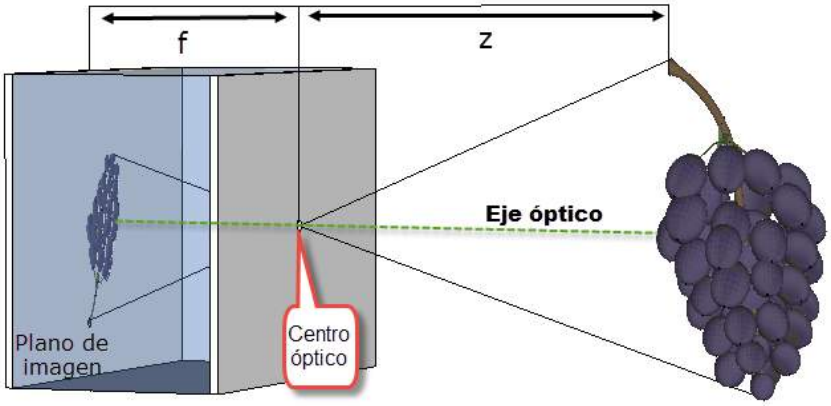


Figura 12.4. Esquema del modelo proyectivo de cámara Pinhole invertido.

Por trigonometría básica de los triángulos semejantes formados en la figura 12.4 se puede obtener la siguiente ecuación (12.6) que relaciona la coordenada u de un punto expresado con respecto a P con la posición de un punto en *X* y *Z* definido con respecto a C. Esta relación es la misma para la coordenada v pero sustituyendo *X* por *Y*.

$$-u = f\frac{X}{Z} \tag{12.6}$$

La figura 12.5 es equivalente matemáticamente a la figura 12.4, cambiando el signo y estando de esta forma la imagen en el mismo sentido que el objeto real. La ecuación general del modelo Pinhole que relaciona el punto ${}^{p}\mathbf{q} = [u,v,1]^T$ (en coordenadas homogéneas) con el punto ${}^{c}\mathbf{p} = [X,Y,Z,1]^T$ es la ecuación (12.7). Esta ecuación difiere de la ecuación (12.6) en que ya no aparece el signo negativo al estar la imagen en el mismo sentido que el objeto real. El punto $\mathbf{p}_2$, al encontrarse en la misma recta visual que $\mathbf{p}_1$, proyecta sobre el mismo punto **q**.

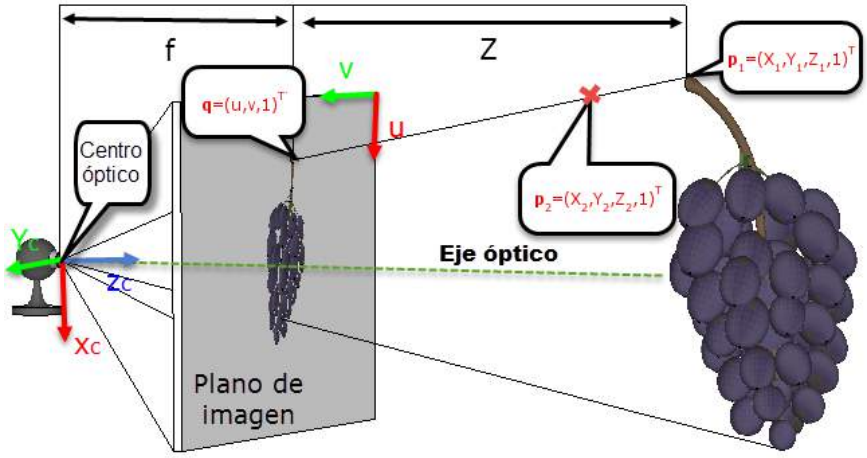


Figura 12.5. Esquema matemáticamente equivalente al modelo proyectivo de cámara Pinhole donde el objeto no se encuentra invertido y en el que el plano de imagen se encuentra entre el objeto real y el centro óptico.

Dado el sistema de coordenadas C asociado a la cámara y un sistema de coordenadas 2D de proyección P asociado al plano de imagen, cuyo origen está situado en el punto principal de la imagen y cuya orientación de sus ejes coinciden con la orientación de los ejes de C, la matriz de proyección **P** transforma las coordenadas del punto ${}^{c}\mathbf{p} = [X,Y,Z,w]^T$ en las coordenadas de este punto proyectado sobre el plano de imagen ${}^{p}\mathbf{q} = [u',v',s]^T$ o ${}^{p}\mathbf{q} = [u, v,1]^T$, de tal manera que ${}^{p}\mathbf{q} = {}^{p}\mathbf{P}_c \cdot {}^{c}\mathbf{p}$.

$$\begin{bmatrix} u' \\ v' \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}; \quad u = \frac{u'}{s}; v = \frac{v'}{s} \tag{12.7}$$

siendo *f* la distancia focal de la óptica.

12.3.3. Parámetros intrínsecos del modelo de cámara (matriz K)

Dado el sistema de coordenadas P asociado al plano de imagen y el sistema de coordenadas pixélicas D, cuyo origen de coordenadas está en la esquina superior izquierda y cuya orientación de los ejes coincide con la de los ejes del sistema P, la matriz de transformación **K** relaciona las coordenadas del punto ${}^{p}\mathbf{q} = [x,y,1]^T$ en las coordenadas de este punto respecto al sistema de coordenadas de imagen D ${}^{d}\mathbf{q} = [u,v,1]^T$, de tal manera que ${}^{d}\mathbf{q} = {}^{d}\mathbf{K}_p \cdot {}^{p}\mathbf{q}$.

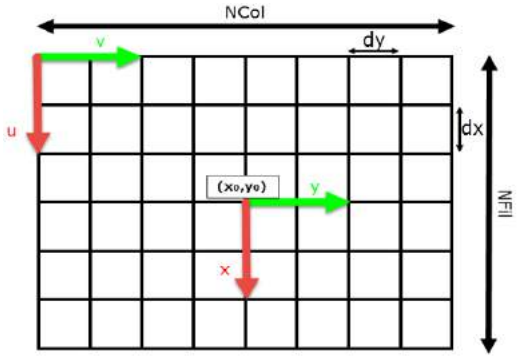


Figura 12.6.Esquema de la relación existente entre un sistema de coordenadas de un sensor con el plano de imagen.

Donde Nfil es el número de filas del sensor; Ncol es el número de columnas del sensor; dx y dy son las dimensiones de un píxel en milímetros y $[x_0,y_0]^T$ son las coordenadas del punto principal en píxeles.

$$\mathbf{K} = \begin{bmatrix} 1/d_x & 0 & x_0 \\ 0 & 1/d_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{12.8}$$

$$\begin{aligned} u &= \frac{x}{d_x} + x_0 \\ v &= \frac{y}{d_y} + y_0 \end{aligned} \tag{12.9}$$

Finalmente el modelo de cámara **M** queda definido por la siguiente composición de matrices:

Parámetros Intrínsecos (K) — Proyección Perspectiva (P) — Parámetros Extrínsecos (T)

$$\begin{bmatrix} u' \\ v' \\ s \end{bmatrix} = \lambda \begin{bmatrix} 1/d_x & 0 & x_0 \\ 0 & 1/d_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{12.10}$$

O lo que es equivalente:

K — P — T

$$\begin{bmatrix} u' \\ v' \\ s \end{bmatrix} = \lambda \begin{bmatrix} a_x & 0 & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{12.11}$$

siendo:

- $a_x = f/dx$, distancia focal horizontal en píxeles
- $a_y = f/dy$, distancia focal vertical en píxeles
- d_x, dy , tamaño del píxel en x e y
- x_0, y_0, punto principal expresado en píxeles

Quedando así concentrados todos los parámetros intrínsecos de la cámara en la nueva matriz **K** y los parámetros extrínsecos que definen la localización de la cámara en la matriz **T**.

12.3.4 Calibración de cámaras

El proceso de calibración de una cámara consiste en estimar los parámetros intrínsecos y extrínsecos de la misma. Este proceso se puede realizar en dos pasos. Primero se estima la matriz de proyección **M** y después se pueden estimar los parámetros intrínsecos y extrínsecos de la cámara a partir de **M**.
Las técnicas de calibración se pueden clasificar, según el tipo de patrón, en técnicas con patrón conocido 3D (Tsai, 1987), patrón conocido planar (Zhang, 2000) e incluso existen técnicas de autocalibración que utilizan puntos de la escena sin introducir ningún tipo de patrón conocido (Faugeras y col., 1992).

Figura 12.7. Ejemplos de patrones de calibración de cámaras. Izquierda: patrón de calibración 3D. Derecha: patrón de calibración planar.

La estimación de la matriz de proyección **M** se puede realizar partiendo de diferentes escenarios conocidos. En estos escenarios se sitúan una serie de puntos en el mundo 3D cuyas posiciones ${}^{w}\mathbf{p} = [X_w, Y_w, Z_w, 1]^T$ respecto al sistema de coordenadas del mundo 3D son conocidas. Se captura una imagen de la escena y se obtiene la posición en píxeles ${}^{d}\mathbf{q}=[u,v,1]^T$ de sus correspondientes proyecciones en la imagen.

$$\begin{bmatrix} w\cdot u \\ w\cdot v \\ w \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{12.12}$$

Esta ecuación está sujeta al factor de escala w que puede ser cualquier número real. Por lo tanto la matriz **M** tiene 11 grados de libertad y se podría forzar el valor de uno de sus elementos (por ejemplo $m_{34} = 1$). Un conjunto de puntos 3D ${}^{w}\mathbf{p}$ de cardinal N junto con sus correspondientes píxeles en la imagen ${}^{d}\mathbf{q}$ se utiliza para definir la siguiente ecuación lineal del sistema, que consta de 2N filas, la cual

se puede resolver para obtener los elementos de la matriz de proyección **M** a partir de al menos seis puntos ${}^{w}\mathbf{p}$ no coplanares, ya que cada par de puntos (${}^{w}\mathbf{p}$, ${}^{d}\mathbf{q}$) añade dos restricciones (o filas) a la ecuación.

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1 \cdot X_1 & -u_1 \cdot Y_1 & -u_1 \cdot Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1 \cdot X_1 & -v_1 \cdot Y_1 & -v_1 \cdot Z_1 & -v_1 \\ \text{M} & \text{M} & \text{M} & \text{M} & \text{M} & \text{M} & \text{M} & \text{M} & \text{M} & \text{M} & \text{M} & \text{M} \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_n \cdot X_n & -u_n \cdot Y_n & -u_n \cdot Z_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_n \cdot X_n & -v_n \cdot Y_n & -v_n \cdot Z_n & -v_n \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ \text{M} \\ m_{33} \\ m_{34} \end{bmatrix} = 0 \qquad (12.13)$$

Esta relación entre las coordenadas de los puntos utilizados para la calibración y los elementos de la matriz de proyección se conoce como *Direct Linear Transformation* (DLT).

$$\begin{gathered} A \cdot m = 0 \\ m_{34} = 1 \rightarrow A' \cdot m' = b \rightarrow m' = \left(A^T \cdot A' \right)^{-1} A^T \cdot b \end{gathered} \qquad (12.14)$$

Una vez obtenidos por mínimos cuadrados los once valores restantes de la matriz **M**, se estima el error geométrico E como indicador del error de calibración.

$$E = \sum_{i=1}^{N} \frac{|\mathbf{q} - \mathbf{q}'|}{N} \qquad (12.15)$$

donde **q**'= **M**·**p**.

Tras esta fase de estimación lineal de la matriz de proyección **M** se puede aplicar un proceso de optimización no lineal para minimizar E. A continuación se presenta un ejemplo numérico de calibración de una cámara.

Ejemplo 12.1 *En la figura 12.8 se muestra el escenario de calibración donde se dispone de un conjunto de seis puntos* ***X*** *en el mundo 3D, situados en este caso sobre dos planos perpendiculares.*

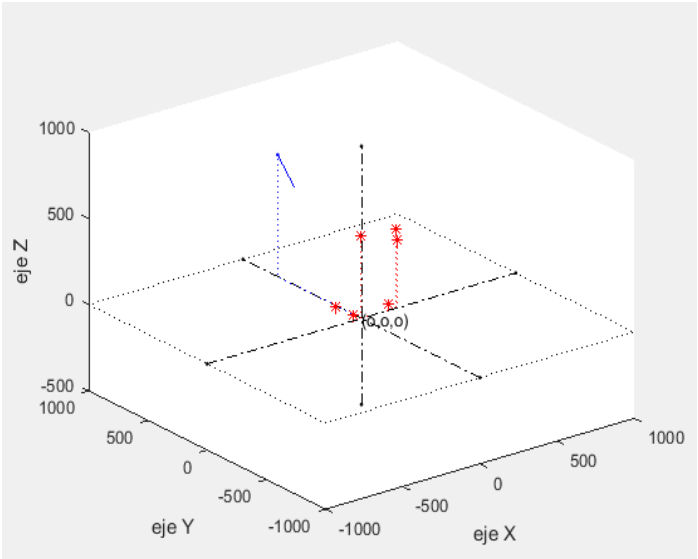


Figura 12.8. Escenario de calibración.

Se parte del modelo de cámara siguiente $\mathbf{M} = \mathbf{K}\cdot\mathbf{P}\cdot\mathbf{T}$ *donde:*

$$\mathbf{K} = \begin{bmatrix} 1580 & 0 & 350 \\ 0 & 1580 & 300 \\ 0 & 0 & 1 \end{bmatrix};$$

K *define los parámetros intrínsecos de la cámara.*

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}; \quad \mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -0.7 & 0.7 & 0 \\ 0 & -0.7 & -0.7 & 1000 \\ 0 & 0 & 0 & 1 \end{bmatrix};$$

T *define los parámetros extrínsecos de la cámara. La cámara está orientada hacia la recta intersección de los dos planos perpendicular en los que se encuentran los puntos y la distancia de la cámara a esa recta es de 1000 mm.*
Teniendo en cuenta que cuando se intenta calibrar una cámara tanto los puntos 3D del mundo ***X*** *como sus imágenes* ***U*** *tendrán un cierto error de medida se añade ruido a los puntos quedando* ***Ur****.*

$$\mathbf{X} = \begin{bmatrix} 228 & 232 & -1 & 235 & 228 & -7 \\ 0 & 0 & 0 & 60 & 70 & 210 \\ 457 & 396 & 479 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix};$$

Se calculan las proyecciones de ***X****, es decir* $\boldsymbol{U}=\boldsymbol{M}\cdot\boldsymbol{X}$ *y se redondean los valores al entero más próximo, quedando:*

$$\mathbf{Ur} = \begin{bmatrix} 882 & 859 & 34 & 340 & 729 & 337 \\ 1054 & 914 & 1109 & 230 & 218 & 24 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix};$$

1.- Calcule el error geométrico E de la calibración obtenida.

2.- ¿Qué sucede si todos los puntos del mundo están en un plano? ¿Por qué?

Solución: Para estimar la matriz de proyección **Mr** se fuerza $m_{34} = 1$ y partiendo de los conjuntos de puntos **X** y **Ur**, se plantea el sistema de ecuaciones $\mathbf{A}'\cdot\mathbf{m}=\mathbf{b}$, donde:

$$\mathbf{A'} = \begin{bmatrix} 228 & 0 & 457 & 1 & 0 & 0 & 0 & 0 & -201096 & 0 & -403074 \\ 0 & 0 & 0 & 0 & 228 & 0 & 457 & 1 & -240312 & 0 & -481678 \\ 232 & 0 & 396 & 1 & 0 & 0 & 0 & 0 & -199288 & 0 & -340164 \\ 0 & 0 & 0 & 0 & 232 & 0 & 396 & 1 & -212048 & 0 & -361944 \\ -1 & 0 & 479 & 1 & 0 & 0 & 0 & 0 & 348 & 0 & -166692 \\ 0 & 0 & 0 & 0 & -1 & 0 & 479 & 1 & 1109 & 0 & -531211 \\ -6 & 60 & 0 & 1 & 0 & 0 & 0 & 0 & 2040 & -20400 & 0 \\ 0 & 0 & 0 & 0 & -6 & 60 & 0 & 1 & 1380 & -13800 & 0 \\ 228 & 70 & 0 & 1 & 0 & 0 & 0 & 0 & -166212 & -51030 & 0 \\ 0 & 0 & 0 & 0 & 228 & 70 & 0 & 1 & -49704 & -15260 & 0 \\ -7 & 210 & 0 & 1 & 0 & 0 & 0 & 0 & 2359 & -70770 & 0 \\ 0 & 0 & 0 & 0 & -7 & 210 & 0 & 1 & 168 & -5040 & 0 \end{bmatrix}; \quad \mathbf{b} = \begin{bmatrix} 882 \\ 1054 \\ 859 \\ 914 \\ 348 \\ 1109 \\ 340 \\ 230 \\ 729 \\ 218 \\ 337 \\ 24 \end{bmatrix};$$

Aplicando mínimos cuadrados se obtiene:

$$\mathbf{Mr} = \begin{bmatrix} 1.5787 & -0.2551 & -0.2471 & 349.8712 \\ 0.0013 & -1.3306 & 0.9039 & 299.7494 \\ 0.0000 & -0.0007 & -0.0007 & 1.0000 \end{bmatrix};$$

1.- El resultado es E=0.0536 después de aplicar la ecuación (12.15).
2.- No se podría resolver el sistema de ecuaciones planteado en la ecuación (12.14) que estima la matriz de proyección **M**. Esto es debido a que los puntos coplanares no introducen nuevas restricciones que permitan resolver la ecuación para obtener las 11 incógnitas.

12.4 Homografía

12.4.1 Definición de homografías

Una homografía **H** es una transformación proyectiva en perspectiva que determina una correspondencia biyectiva entre los puntos de dos planos.

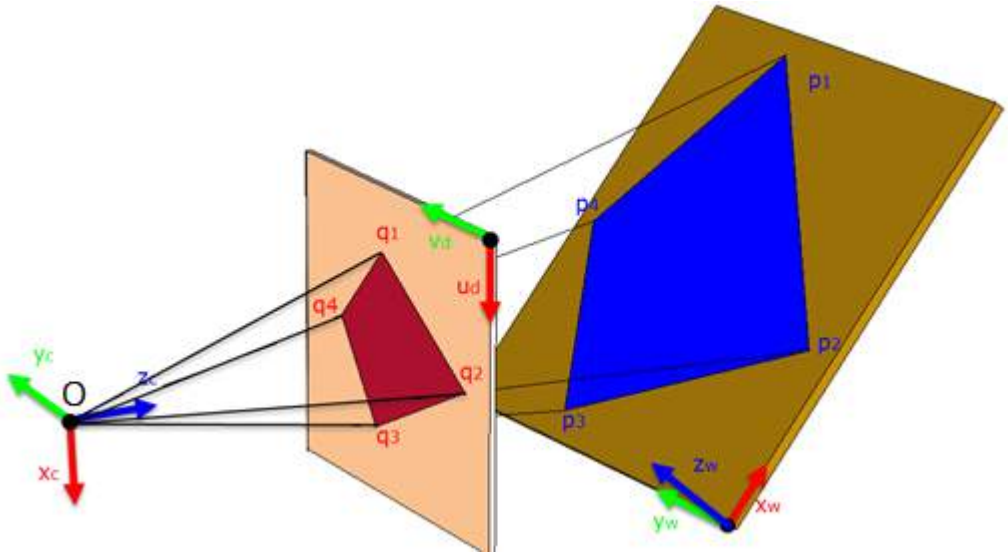


Figura 12.9.Homografía entre los planos P y P' desde el centro de radiación O donde los puntos q_1, q_2, q_3 y q_4 tienen sus puntos homólogos p_1, p_2, p_3 y p_4 respectivamente

Desde el punto de vista del modelo de cámaras una homografía es el modelo de proyección de perspectiva que se puede utilizar cuando se restringe el escenario del mundo 3D a un plano. En este caso se proyecta un plano del mundo 3D sobre el plano del sensor de la cámara.

En el caso de la matriz de proyección **M**, la proyección se produce desde un espacio de tres dimensiones, como es el mundo, a un espacio de dos dimensiones como es el plano del sensor de la cámara, por ello **M** es una matriz de 3x4. Al tratarse de un matriz 3x4 cuando se proyecta se pierde la información de profundidad. Sin embargo, la matriz **H** de la homografía es una matriz 3x3 y al proyectar con **H** no se pierde información, por lo tanto en este caso el problema de proyección inversa tiene solución utilizando únicamente las relaciones de proyección, sin necesidad de más restricciones adicionales.

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{12.16}$$

$$\begin{bmatrix} u \\ v \\ s \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \mathbf{H_{3x3}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{12.17}$$

$$\mathbf{H} = \lambda \begin{bmatrix} a_x & 0 & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \mathrm{g} \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix} \tag{12.18}$$

En la actualidad existen muchos escenarios de aplicaciones donde la información tridimensional se obtiene empleando técnicas de homografía como base. Algunos ejemplos recientes de estos escenarios de aplicaciones en el área de control de procesos son el control del proceso de fermentación de masas de harina (Ivorra y col., 2014) o la medición en línea del volumen de diferentes alimentos (Verdú y col., 2013).

12.4.2 Estimación de homografías

La matriz **H** es una matriz 3x3 bajo el efecto de un factor de escala por lo tanto tiene ocho grados de libertad. Para poder estimar esta matriz se requieren al menos cuatro puntos ${}^{w}\mathbf{p} = [X_w, Y_w, 1]^T$ sobre un plano del mundo sin que tres de ellos estén alineados y sus respectivas proyecciones ${}^{d}\mathbf{q}=[u,v,1]^T$.

$$s_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \mathbf{H_{3x3}} \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix} \tag{12.19}$$

Fijando $h_{33} = 1$:

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \tag{12.20}$$

Se plantea la siguiente ecuación:

$$\begin{bmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -y_1X_1 & -y_1Y_1 \\ X_2 & Y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 \\ 0 & 0 & 0 & X_2 & Y_2 & 1 & -y_2X_2 & -y_2Y_2 \\ X_3 & Y_3 & 1 & 0 & 0 & 0 & -x_3X_3 & -x_3Y_3 \\ 0 & 0 & 0 & X_3 & Y_3 & 1 & -y_3X_3 & -y_3Y_3 \\ X_4 & Y_4 & 1 & 0 & 0 & 0 & -x_4X_4 & -x_4Y_4 \\ 0 & 0 & 0 & X_4 & Y_4 & 1 & -y_4X_4 & -y_4Y_4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \\ x_4 \\ y_4 \end{bmatrix} \tag{12.21}$$

Donde se parte como datos de entrada de un conjunto de puntos en el mundo 3D situados en un plano y de las proyecciones de estos puntos sobre el plano de imagen de la cámara.

12.4. Bibliografía

Bosch-Jorge, M.; Sanchez-Salmeron, A. J.; Ricolfe-Viala, C. (2012). Visual-based human action recognition on smart phones based on 2D and 3D descriptors. International Journal of Pattern Recognition and Artificial Intelligence, 26(08).

Bosch-Jorge, M.; Sánchez-Salmerón, A. J.; Valera, Á.; Ricolfe-Viala, C. (2014). Fall detection based on the gravity vector using a wide-angle camera. Expert Systems with Applications, 41(17), 7980-7986.

Bradski,G; Kaehler,A. (2008) Learning OpenCV; 1ªed; O'Reilly Media Inc, Sebastopol, United States of America, pp. 370-396.

Cyganek,B; Siebert, J.Paul (2009) An introduction to 3D Computer Vision Techniques and Algorithms; 1ªed; John Wiley Sons, Ltd; Chichester, United Kingdom, pp. 3-31.

Faugeras, O. (1993). Three-dimensional computer vision: a geometric viewpoint. MIT press.

Faugeras, O. D.; Luong, Q. T.; Maybank, S. J. (1992, January). Camera self-calibration: Theory and experiments. In Computer Vision—ECCV'92 (pp. 321-334). Springer Berlin Heidelberg.

Ivorra, E.; Amat, S. V.; Sánchez, A. J.; Barat, J. M.; Grau, R. (2014). Continuous monitoring of bread dough fermentation using a 3D vision Structured Light technique. Journal of Food Engineering, 130(0), 8–13.

Laussedat, A. (1898). Recherches sur les instruments: Les méthodes et le dessin topographiques. Gauthier-Villars.

Meydenbauer, A. (1867). Ueber die Anwendung der Photophie zur Architektur-und Terrain-Aufnahme. Zeitschrift für Bauwesen, 17,61-70.

Peiravi, A.; Taabbodi, B. (2010) A reliable 3D laser triangulation-based scanner with a new simple but accurate procedure for finding scanner parameters. J. Am. Sci. 6, 80–85

Ricolfe-Viala, C.; Sánchez-Salmerón, A.J. (2010). Robust metric calibration of non-linear camera lens distortion. Pattern Recognition, 43(4), 1688-1699.

Ricolfe-Viala, C.; Sanchez-Salmeron, A.J. (2010). Lens distortion models evaluation. Applied Optics, 49(30), 5914-5928.

Sánchez, A. J.; Martínez, J. M. (2000). Robot-arm pick and place behavior programming system using visual perception. In Pattern Recognition, 2000. Proceedings. 15th International Conference on (Vol. 4, pp. 507-510). IEEE.

Sánchez, A. J.; Marchant, J. A. (2000). Fusing 3D information for crop/weeds classification. In Pattern Recognition, 2000. Proceedings. 15th International Conference on (Vol. 4, pp. 295-298). IEEE.

Tsai, R. Y. (1987). A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses.Robotics and Automation, IEEE Journal of, 3(4), 323-344.

Verdú, S.; Ivorra, E.; Sánchez, A. J.; Girón, J.; Barat, J. M.; Grau, R. (2013). Comparison of TOF and SL techniques for in-line measurement of food item volume using animal and vegetable tissues. Food Control, 33(1), 221–226.

Wang, Y. J.; Zhang, S. (2010). Optimal pulse width modulation for sinusoidal fringe generation with projector defocusing, Opt. Lett. 35, 4121–4123.

Wöhler,G. (2009) 3D Computer Vision; 1ªed; Springer-Velag,Berlin, Germany, pp. 243-343.
Zhang, Z. (2000). A flexible new technique for camera calibration. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 22(11), 1330-1334.

Apéndice: nomenclatura

- ${}^{j}\mathbf{p}= [X,Y,Z,1]^T$, vector columna que representa un punto 3D expresado en coordenadas homogéneas respecto al sistema de coordenadas J. Un punto en coordenadas homogéneas tiene infinitas representaciones ${}^{j}\mathbf{p}= [X\cdot t, Y\cdot t, Z\cdot t, t]^T$ siendo t cualquier número real.
- ${}^{i}\mathbf{q} = [u,v,1]^T$, vector columna que representa un punto 2D expresado en coordenadas homogéneas respecto al sistema de coordenadas I.
- ${}^{i}\mathbf{A}_j$, matriz de transformación homogénea que relaciona el sistema de coordenadas J con respecto al sistema de coordenadas I, de tal manera que ${}^{i}\mathbf{p} = {}^{i}\mathbf{A}_j \cdot {}^{j}\mathbf{p}$.

- Para simplificar se puede eliminar los subíndices y los superíndices de las anteriores vectores y matrices siempre que por el contexto se pueda interpretar claramente su significado.
- k_0, k_1 y k_2 son los coeficientes para corregir la distorsión radial
- d_0 y d_1 son los coeficientes para corregir la distorsión tangencial
- f, es un valor escalar que representa la distancia focal de la óptica.
- **o**, es un vector que representa el centro óptico de la óptica.
- F máximo, es un valor escalar que representa el valor máximo de la abertura relativa del diafragma.
- Charge-Coupled Device (CCD), sensor de la cámara con tecnología de dispositivo de carga acoplada.
- Nfil, es un valor escalar que representa el número de filas del sensor.
- Ncoln, es un valor escalar que representa el número de columnas del sensor.
- dx, dy, son valores escalares que representan el tamaño del pixel en milímetros.
- $\mathbf{0}=[x0,y0]^T$, es un vector que representa el punto principal o imagen del eje óptico en píxeles.

- $W=\{\mathbf{o}_w, \mathbf{x}_w, \mathbf{y}_w, \mathbf{z}_w\}$, sistema de referencia arbitrario del mundo.
- $C=\{\mathbf{o}, \mathbf{x}_c, \mathbf{y}_c, \mathbf{z}_c\}$, sistema de referencia de la cámara que tiene el origen en el centro óptico y el eje $\mathbf{z}_c$ coincide con el eje óptico.
- $P=\{\mathbf{0}, \mathbf{u}_p, \mathbf{v}_p\}$, sistema de referencia asociado al plano de imagen, cuyo origen está situado en el punto principal de la imagen y cuya orientación de sus ejes coinciden con la orientación de los ejes de C, es decir $\mathbf{u}_p= \mathbf{x}_c$; $\mathbf{v}_p=\mathbf{y}_c$.
- $D=\{\mathbf{o}_d, \mathbf{u}_d, \mathbf{v}_d\}$, sistema de referencia de la imagen donde el origen $\mathbf{o}_d$ está situado en la esquina superior izquierda y cuya orientación de sus ejes coinciden con la orientación de los ejes de P, es decir $\mathbf{u}_d= \mathbf{u}_p$; $\mathbf{v}_d= \mathbf{v}_p$.

- Todos los sistemas de referencia que se utilizan son dextrógiros. Esto quiere decir que $\mathbf{x}_i \times \mathbf{y}_i = \mathbf{z}_i$.

- **T**, matriz de transformación homogénea de parámetros extrínsecos 4x4 (indica la transformación del sistema de coordenadas del mundo W al sistema de coordenadas de la cámara C, por ello también se expresa como ${}^{c}\mathbf{T}_{w}$).
- **R**, matriz de rotación 3x3 de la matriz **T**.
- **t**, vector de traslación 3x1 de la matriz **T**.
- **P**, matriz de proyección 3x4 en función de la distancia focal *f*.

CAPÍTULO 13

VISIÓN 3D: ESTEREOSCOPÍA

Rafael C. GONZÁLEZ[1], José A. CANCELAS[1], Ignacio ÁLVAREZ[1], José M. ENGUITA[1]

[1] Profesores Titulares de Universidad: Departamento de Ingeniería Eléctrica, Electrónica, de Computadores y Sistemas, Universidad de Oviedo, Gijón, España

Las técnicas de visión estereoscópica permiten inferir información geométrica (3D) sobre la escena. Se basan en aprovechar la información procedente de dos imágenes distintas para suplir la pérdida de información que supone el proceso de proyección de la escena en la imagen 2D. Para poder realizar la reconstrucción es necesario identificar las proyecciones de un mismo punto de la escena en cada una de las imágenes, para después, utilizar la información geométrica sobre la disposición relativa de las imágenes y así poder obtener la posición de dichos puntos en el espacio. La restricción epipolar liga la geometría de ambas imágenes y resulta clave para facilitar la búsqueda de correspondencias, así como para inferir la posición y orientación de una cámara respecto a la otra cuando el par estéreo no ha podido ser calibrado.

13.1. Introducción

En el presente capítulo se analiza la relación existente entre dos imágenes de una misma escena. Este es el caso más simple de análisis multi-imagen y su comprensión es fundamental para poder profundizar en el tema. El análisis de múltiples imágenes permite realizar reconstrucciones de la escena para aplicaciones de guiado de robots, ingeniería inversa, reconstrucción de panoramas, etc.

Para resolver el problema de la reconstrucción de la escena es necesario resolver dos problemas:

- La determinación de la geometría relativa de las cámaras.
- La identificación de las proyecciones de un mismo punto de la escena sobre cada una de las imágenes.

El capítulo está estructurado para atender estos dos grandes problemas. En la sección 2 se estudian las relaciones existentes entre dos imágenes desde el punto de vista del proceso geométrico de

formación de la imagen. Para ello, se comienza con el estudio de la matriz fundamental que fija una restricción que deben cumplir las proyecciones de un mismo punto de la escena expresadas en coordenadas de cada imagen. A continuación se analiza la matriz esencial, equivalente a la matriz fundamental pero que relaciona coordenadas normalizadas de la imagen. Además, la matriz esencial permite obtener la posición y la orientación de una cámara respecto de la otra salvo por un factor de escala. La sección continúa con el estudio de métodos para la triangulación de la posición 3D de los puntos que se identifican en las imágenes. Un caso de particular interés es aquel en el que las cámaras tienen sus planos imagen coincidentes y los centros ópticos separados una cierta distancia en la dirección de las filas de la imagen. Esta configuración, conocida como configuración canónica, facilita enormemente el desarrollo de algoritmos de emparejamiento y la posterior reconstrucción de la escena. La sección dedicada al análisis geométrico de un sistema estéreo finaliza con la introducción del concepto de rectificación de imágenes, un mecanismo que nos permite llevar las imágenes de un par a una configuración equivalente a la configuración canónica.

El resto del capítulo se dedica al establecimiento de correspondencia, abordando primero aspectos generales para entrar después en la descripción de los algoritmos básicos que permiten obtener los emparejamientos de todos los puntos de una imagen: algoritmos basados en correlación y algoritmos basados en programación dinámica.

13.2. Geometría de los sistemas estéreo

El mecanismo de formación de la imagen implica pérdida de información como consecuencia de la proyección de un espacio tri-dimensional (3D), la escena, en uno bi-dimensional (2D), la imagen. Para poder recuperar la información original es necesario recurrir al uso de información adicional. La mayoría de los seres vivos recurre al uso de varias imágenes para poder recuperar la información 3D por triangulación. Los sistemas de visión estéreo buscan imitar esta capacidad mediante la captura simultánea de varias vistas de una escena o, en el caso de una escena estática, el uso de una secuencia de imágenes al desplazar la cámara por la escena. En este capítulo se estudia el problema de cómo recobrar la información tridimensional a partir de dos imágenes de la escena tomadas desde distintas posiciones.

Dos imágenes de una misma escena están relacionadas entre sí ya que son proyecciones de una misma información 3D. La geometría que relaciona ambas vistas se conoce como *geometría epipolar* y las relaciones de dependencia que se establecen entre las proyecciones de un punto de la escena en ambas imágenes se denominan *restricciones epipolares*.

Las entidades geométricas que intervienen están representadas en la figura 13.1 y son:

- *Línea base*: es el segmento 3D que une los centros de proyección de ambas cámaras.
- *Plano epipolar*: Es el plano que queda definido por un punto de la escena y los centros de proyección de ambas cámaras. Como consecuencia, los planos epipolares forman un haz de planos cuyo eje es precisamente la línea base.
- *Epipolo*: para cada vista del sistema estéreo, el epipolo es el punto en el que la línea base corta al plano imagen asociado a dicha vista. El epipolo es pues la proyección en una cámara, del centro de proyección de la otra cámara que forma el par estéreo. El epipolo puede caer fuera del trozo del plano imagen que devuelve la cámara. En particular, cuando

la línea base es paralela al plano imagen, el epipolo es un punto en el infinito que puede ser representado por un punto del plano proyectivo de la forma $[a,b,0]^T$.

- *Recta epipolar*: es la recta intersección del plano epipolar con el plano imagen. Para entender su interpretación geométrica, considérese un punto $\mathbf{x}_1$ de la imagen 1 (imagen izquierda en la figura 13.1). La recta definida por $\overline{\mathbf{x}_1\mathbf{O}_1}$ es el lugar geométrico de todos los puntos de la escena cuya proyección en la imagen 1 es $\mathbf{x}_1$. Esta recta define, junto con la línea base, el plano epipolar que puede ser interpretado como el lugar geométrico de todos los rayos que proyectan los puntos de la recta $\overline{\mathbf{x}_1\mathbf{O}_1}$ sobre la imagen 2. Las intersecciones de estos rayos con el plano imagen 2 forman una recta, la recta epipolar, en el plano imagen de la cámara 2. En resumen, la recta epipolar en la imagen 2 es la proyección sobre esta imagen de la recta definida por el punto $\mathbf{x}_1$ y el centro de proyección de la cámara 1.

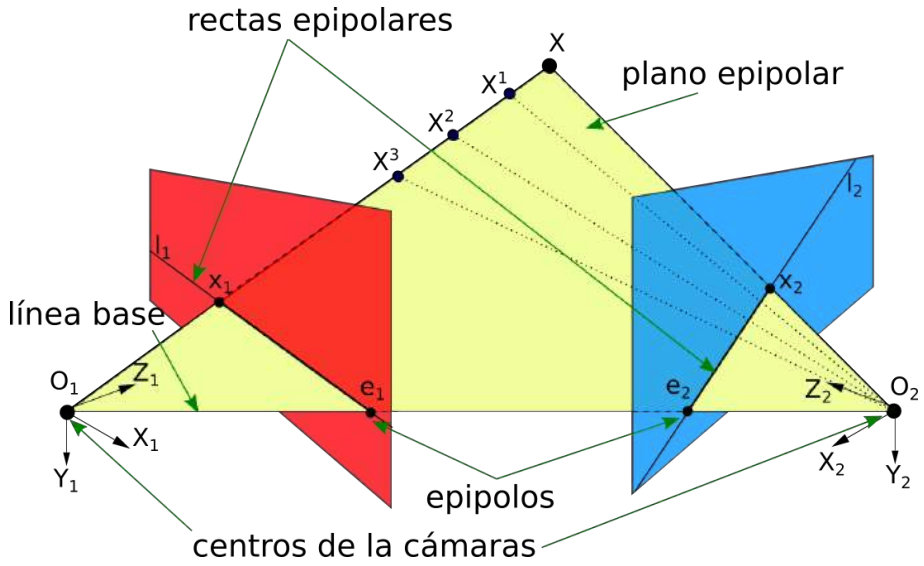


Figura 13.1. Representación de la geometría epipolar. El plano formado por el punto X y los centros de proyección de ambas cámaras es el plano epipolar. Su intersección con los planos imagen determina las rectas epipolares $(\mathbf{l}_1, \mathbf{l}_2)$. Cualquier punto situado en la línea $\overline{\mathbf{x}_1\mathbf{O}_1}$ tiene la misma proyección en la imagen 1 $(\mathbf{x}_1)$. Su proyección en la imagen 2 debe pertenecer a la correspondiente recta epipolar.

Ejemplo 0.1. *Demostrar que el epipolo es el punto de corte de todas las rectas epipolares de la imagen.*

Solución: La línea base es el eje común del haz de planos epipolares, por tanto, pertenece a todos ellos. Dado un plano epipolar, el epipolo es la intersección de la línea base con el plano imagen de la cámara, por ello, formará parte de la correspondiente recta epipolar. Esta relación se cumple para cualquier plano epipolar que se seleccione. Por tanto, las rectas epipolares forman un haz cuyo vértice es el epipolo. En la figura 13.2 se esquematiza esta explicación.

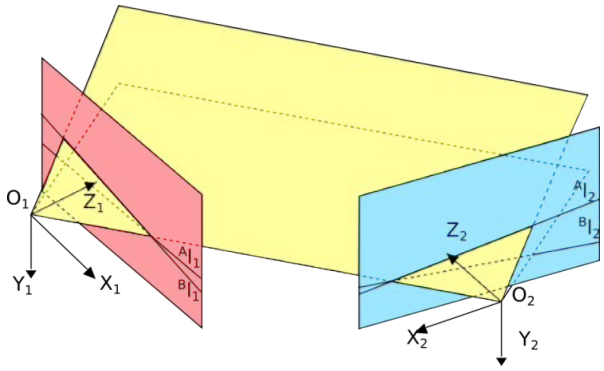


Figura 13.2. Representación del haz de planos epipolares y de las líneas epipolares asociadas a cada uno de ellos.

13.2.1.Matriz Fundamental

Supongamos un sistema estéreo del que conocemos las matrices de proyección asociadas a cada una de las cámaras $\mathbf{P}_1$ y $\mathbf{P}_2$. Sin pérdida de generalidad, podemos suponer que ambas matrices están referidas al sistema de coordenadas asociado a la primera cámara, con lo que dichas matrices tendrán la forma:

$$\mathbf{P}_1 = \mathbf{K}_1\left[\mathbf{I}_{3\times 3}|\mathbf{0}_{3\times 1}\right] \quad , \quad \mathbf{P}_2 = \mathbf{K}_2\left[\mathbf{R}|\mathbf{t}\right] \tag{13.1}$$

Sea X un punto 3D cuyas proyecciones son $\mathbf{x_1}$ y $\mathbf{x_2}$ respectivamente. La restricción epipolar determina que el punto $\mathbf{x}_1$ define una recta $\mathbf{l}_2$, en la segunda imagen, que contiene al punto $\mathbf{x}_2$. En el plano proyectivo, la pertenencia de un punto a una recta se expresa como:

$$\mathbf{x}_2^{\mathrm{T}}\mathbf{l}_2 = 0 \tag{13.2}$$

La recta l_2 es la proyección de la recta $\overline{\mathbf{x}_1\mathbf{O}_1}$ sobre la imagen 2. La recta $\overline{\mathbf{x}_1\mathbf{O}_1}$ es la proyección inversa del punto $\mathbf{x}_1$, que resulta:

$$\mathbf{X}(\lambda) = \begin{bmatrix}\mathbf{K}_1^{-1}\mathbf{x}_1 \\ 0\end{bmatrix} + \lambda\begin{bmatrix}0_{3\times 1} \\ 1\end{bmatrix} = \mathbf{D} + \lambda\mathbf{O}_1 \tag{13.3}$$

Donde $\mathbf{D}$ es el punto en el infinito de la recta y $\mathbf{O_1}$ es el centro de proyección de la cámara 1. La proyección de dicha recta en la imagen 2 es la recta definida por los puntos resultantes de proyectar $\mathbf{D}$ y $\mathbf{O}_1$ sobre la imagen 2, con lo que resulta:

$$\mathbf{e}_2 = \mathbf{P}_2\mathbf{O}_1 = \mathbf{K}_2\left[\mathbf{R}|\mathbf{t}\right]\begin{bmatrix}0_{3\times 1} \\ 1\end{bmatrix} = \mathbf{K}_2\mathbf{t} \tag{13.4}$$

$$\mathbf{d}_2 = \mathbf{P}_2\mathbf{D} = \mathbf{K}_2\left[\mathbf{R}|\mathbf{t}\right]\begin{bmatrix}\mathbf{K}_1^{-1}\ \mathbf{x}_1 \\ 0\end{bmatrix} = \mathbf{K}_2\mathbf{R}\ \mathbf{K}_1^{-1}\mathbf{x}_1 \tag{13.5}$$

$$\mathbf{l}_2 = \mathbf{e}_2 \times \mathbf{d}_2 = [\mathbf{e}_2]_x \mathbf{d}_2 = \left(\mathbf{K}_2^{-T}[\mathbf{t}]_\times \mathbf{R}\mathbf{K}_1^{-1}\right)\mathbf{x}_1 = \mathbf{F}\mathbf{x}_1 \tag{13.6}$$

Sustituyendo en la ecuación (13.2), resulta que la restricción epipolar puede expresarse algebraicamente como:

$$\mathbf{x}_2^T \mathbf{F} \mathbf{x}_1 = 0 \tag{13.7}$$

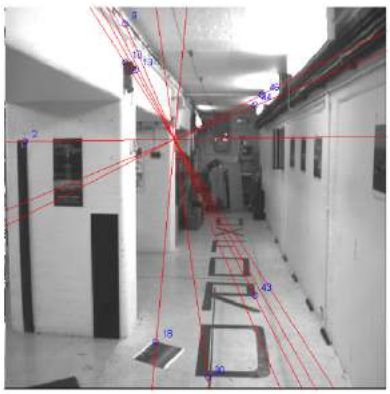
La matriz **F** se conoce como matriz fundamental y recoge todas las restricciones asociadas a la geometría epipolar.

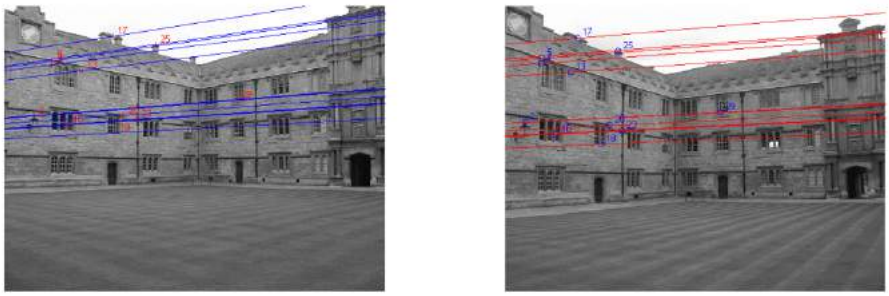
13.2.1.1. Propiedades de la matriz fundamental

Sea **F** la matriz fundamental asociada a un par de cámaras $\mathbf{P}_1$ y $\mathbf{P}_2$, entonces, para cada pareja de puntos correspondientes ($\mathbf{x}_1 \leftrightarrow \mathbf{x}_2$) se cumplirá la ecuación (13.7). Si invertimos el orden de las imágenes, la matriz fundamental asociada al nuevo par de cámaras $\mathbf{P}_2$ y $\mathbf{P}_1$ es $\mathbf{F}^T$.

La matriz **F** es una matriz de rango 2, ya que $[\mathbf{t}]_\times$ es una matriz antisimétrica y su rango ha de ser par. Esto concuerda con el hecho de que la matriz fundamental representa un mapeo entre el plano proyectivo de la imagen 1 (de dimensión 2) y la correspondiente recta epipolar (de dimensión 1). Aunque **F** es una matriz con nueve elementos, sólo dispone de siete grados de libertad debido a que debe cumplir dos restricciones. Por un lado, al ser una matriz de rango 2, su determinante debe ser cero. Por otro, **F** representa una relación en el plano proyectivo y por tanto está determinada salvo por un factor de escala, es decir, la ecuación (13.7) se verifica para cualquier matriz de la forma $\lambda\mathbf{F}$, con λ un escalar distinto de cero

(a)





(b)

Figura 13.3. Ejemplo del cálculo de las líneas epipolares.

Las líneas epipolares se obtienen directamente a partir de la matriz **F**, tal y como se deduce de la ecuación (13.6). Para obtener la línea epipolar correspondiente a un punto $\mathbf{x}_2$ de la imagen 2, solo es necesario utilizar $\mathbf{F}^T$, por lo que $\mathbf{l}_1 = \mathbf{F}^T\mathbf{x}_2$. La figura 13.3 muestra dos ejemplos de cálculo de rectas epipolares. En (a) el desplazamiento ha sido fundamentalmente una traslación paralela al eje óptico de la cámara. En este caso los epipolos aparecen dentro de la imagen. En (b) las cámaras se han desplazado aproximadamente de forma perpendicular a la dirección del eje óptico y además ha habido una pequeña rotación. Los epipolos están fuera de la imagen.

Para el cálculo de los epipolos, es necesario tener en cuenta que éstos pertenecen a todas las rectas epipolares de la imagen por lo que:

$$\mathbf{x}_2^T\mathbf{F}\mathbf{e}_1 = 0 \quad \forall \mathbf{x}_2 \quad \text{y} \quad \mathbf{e}_2^T\mathbf{F}\mathbf{x}_1 = 0 \quad \forall \mathbf{x}_1 \tag{13.8}$$

La única forma de que se cumplan estas condiciones para cualquier punto de la imagen es que al transformar el epipolo mediante el uso de la matriz fundamental, obtengamos el vector nulo:

$$\mathbf{F}\mathbf{e}_1 = \mathbf{0} \qquad \text{y} \quad \mathbf{F}^T\mathbf{e}_2 = \mathbf{0} \tag{13.9}$$

Las ecuaciones (13.9) pueden ser resueltas mediante descomposición SVD de la matriz $\mathbf{F} = \mathbf{U}\mathbf{D}\mathbf{V}^T$, siendo $\mathbf{e}_1$ la tercera columna de **V** y $\mathbf{e}_2$ la tercera columna de **U**.

En el caso de que las matrices de proyección de las cámaras estén referidas a un sistema de coordenadas genérico, la matriz fundamental puede obtenerse mediante la expresión:

$$\mathbf{F} = [\mathbf{e}_2]_x \mathbf{P}_2\mathbf{P}_1^+ \quad \text{con } [\mathbf{e}_2]_x = \mathbf{P}_2\mathbf{C} \text{ y } \mathbf{P}_1\mathbf{C} = \mathbf{0} \tag{13.10}$$

$\mathbf{P}_1^+$ es la pseudo inversa de $\mathbf{P}_1$ y **C** es el centro de la cámara 1, cuya posición se obtiene al resolver el sistema de ecuaciones indicado en la ecuación (13.10).

Las matrices de proyección asociadas a un par de cámaras cuya matriz fundamental es **F**, pueden ser elegidas como:

$$\mathbf{P}_1 = [\mathbf{I}_{3\times3} \,|\, \mathbf{0}_{3\times1}] \quad \mathbf{P}_2 = [[\mathbf{e}_2]_x\mathbf{F} \,|\, \mathbf{e}_2] \tag{13.11}$$

La matriz fundamental es invariante a transformaciones proyectivas, por lo que si **H** de dimensión 4x4 es una transformación proyectiva en el espacio3D, la matriz fundamental asociada a $\mathbf{P}_1\mathbf{H}$ y $\mathbf{P}_2\mathbf{H}$ es la misma que la asociada a $\mathbf{P}_1$ y $\mathbf{P}_2$.

13.2.1.2. Obtención de la matriz fundamental: Algoritmo de 8-puntos

Este algoritmo fue propuesto originalmente por Lounguet-Higgins (1981) y nos permite estimar, de forma lineal, el valor de **F** a partir del conocimiento de un número suficiente de parejas de puntos ($\mathbf{x}_1^i \leftrightarrow \mathbf{x}_2^i$). Si se aplica la restricción epipolar a cada pareja de puntos, se obtiene una ecuación de la forma:

$$\begin{bmatrix} & & & & & & & & & \\ & & & & & & & & & \\ & & & & & \mathrm{M} & & & & \\ x_2^i x_1^i & x_2^i y_1^i & x_2^i & y_2^i x_1^i & y_2^i y_1^i & y_2^i & x_1^i & y_1^i & 1 \\ & & & & & \mathrm{M} & & & & \end{bmatrix} \mathbf{f} = \mathbf{A}\mathbf{f} = \mathbf{0} \tag{13.12}$$

De esta forma se obtiene un sistema de ecuaciones lineal. Si se aplica la restricción de que **F** está definida salvo por un factor de escala, quedan 8 grados de libertad y la ecuación (13.12) puede resolverse mediante la descomposición en valores singulares de la matriz de coeficientes $\mathbf{A}=\mathbf{U}\mathbf{D}\mathbf{V}^{\mathbf{T}}$. La solución es la última columna de la matriz **V**.

Un problema asociado a este método es que la matriz **F** que se obtiene será en general rango 3, ya que en ningún momento se impone la condición de que su determinante sea 0. Imponer dicha condición requeriría la utilización de métodos no lineales. La solución consiste en sustituir **F** por la matriz de rango 2 más próxima. Si la descomposición SVD de **F** es $\mathbf{F} = \mathbf{U}diag(\sigma_1, \sigma_2, \sigma_3)\mathbf{V}^{\mathrm{T}}$, con $\sigma_1 \geq \sigma_2 \geq \sigma_3$, la matriz buscada es:

$$\mathbf{F}' = \mathbf{U}diag(\sigma_1, \sigma_2, 0)\mathbf{V}^T \tag{13.13}$$

Hartley (1997) propuso una mejora, consistente en la normalización previa de los valores de las proyecciones de los puntos con el fin de mejorar la estabilidad numérica del algoritmo. Para los puntos de cada imagen se calcula una transformación que traslada y centra los puntos de modo que:

1. El centroide de los puntos trasladados se sitúe en el origen de coordenadas.
2. Se reescalan de modo que la distancia media al origen sea $\sqrt{2}$.

El algoritmo completo queda como sigue:

Algoritmo 0.1. *Algoritmo normalizado de 8-puntos*

Objetivo: Obtener la matriz fundamental, **F**, a partir de n ≥ 8 correspondencias entre puntos de ambas imágenes $\{\mathbf{x}_1^i \leftrightarrow \mathbf{x}_2^i\}$.

Solución:

1. Normalización de los datos: transformar las coordenadas de los puntos de imagen según $\hat{\mathbf{x}}_1^i = \mathbf{T}_1\mathbf{x}_1$ y $\hat{\mathbf{x}}_2^i = \mathbf{T}_2\mathbf{x}_2$, cumplen las condiciones de normalización de Hartley.
2. Calcular la matriz fundamental correspondiente a las correspondencias en coordenadas normalizadas:
   a. Aplicar la ecuación (13.12) a las parejas $\{\hat{\mathbf{x}}_1^i \leftrightarrow \hat{\mathbf{x}}_2^i\}$, para obtener la matriz de coeficientes $\widehat{\mathbf{A}}$.

b. Obtener la descomposición en valores singulares $\widehat{\mathbf{A}} = \mathbf{UDV}^{\mathrm{T}}$
c. Construir $\hat{\mathbf{F}}$ colocando por filas los elementos de la última columna de $\mathbf{V}$.
d. Si $\hat{\mathbf{F}} = \hat{\mathbf{U}} diag(\sigma_1, \sigma_2, \sigma_3)\hat{\mathbf{V}}^{\mathrm{T}}$ con $\sigma_1 \geq \sigma_2 \geq \sigma_3$, obtener $\widehat{\mathbf{F}'} = \hat{\mathbf{U}} diag(\sigma_1, \sigma_2, 0)\hat{\mathbf{V}}^{\mathrm{T}}$, para asegurar que la solución es de rango 2.

3. Deshacer la normalización: La matriz buscada es $\mathbf{F} = \mathbf{T}_2^{\mathrm{T}} \widehat{\mathbf{F}'} \mathbf{T}_1$

El cálculo de **F** es muy sensible a la presencia de emparejamientos erróneos. Una forma de solucionar este problema consiste en el uso de RANSAC (Random Sample Consensus), algoritmo propuesto por Fischler & Bolles (1981) para mejorar la robustez en la estimación de modelos. La idea básica de RANSAC consiste en extraer, de forma aleatoria, una muestra mínima de los datos que permita la estimación del modelo. El resto de los puntos se clasifican en *inliers* o *outliers* en función del error de ajuste al modelo obtenido. El proceso se repite el suficiente número de veces como para asegurar que hay un 95% de probabilidad de que una de las muestras sólo contenga datos correctos. La solución es aquella que produce el mayor número de *inliers*. La aplicación de este método al cálculo de la matriz fundamental se describe en el siguiente algoritmo.

Algoritmo 0.2. *Cálculo robusto de la matriz fundamental mediante RANSAC*

Objetivo: Obtener la matriz fundamental **F** entre dos imágenes a partir de un conjunto de correspondencias $\{\mathbf{x}_1^{\mathrm{i}} \leftrightarrow \mathbf{x}_2^{\mathrm{i}}\}$ que puede contener emparejamientos erróneos.

Solución:

1. N ← ∞, repeticiones ← 0
2. Mientras N > repeticiones
   a. Seleccionar aleatoriamente 8 parejas de puntos y estimar **F** usando el algoritmo X.1.
   b. Calcular el error cometido con cada emparejamiento utilizando la distancia de Sampson: $d^2 = \frac{\mathbf{x}_2^{\mathrm{T}}\mathbf{F}\mathbf{x}_1}{(\mathbf{F}\mathbf{x}_1)_u^2 + (\mathbf{F}\mathbf{x}_1)_v^2 + (\mathbf{F}_2^{\mathrm{T}}\mathbf{x}_2)_u^2 + (\mathbf{F}_2^{\mathrm{T}}\mathbf{x}_2)_v^2}$ donde los subíndices u y v representan las los dos primeras coordenadas de las correspondientes rectas epipolares en coordenadas homogéneas.
   c. Calcular el número de *inliers*, es decir, aquellas correspondencias para las que el error calculado en el paso anterior está por debajo de un umbral.
   d. La probabilidad de escoger un *outlier* es p ← 1-(número de *inliers* / número de correspondencias).
   e. Actualizar el número de repeticiones: N ← $\frac{\log(1-0.95)}{\log(1-(1-p)^8)}$
   f. Incrementar repeticiones
3. Estimar la matriz fundamental utilizando todos los inliers identificados para la mejor solución encontrada en el paso 2.

13.2.2. Matriz Esencial

La matriz fundamental establece una restricción entre dos imágenes sin calibrar ya que relaciona puntos expresados en píxeles. Por tanto, la matriz fundamental codifica información relacionada tanto con la posición relativa de cada cámara como con los parámetros intrínsecos de cada una de ellas. Si las cámaras están calibradas se puede trabajar con coordenadas normalizadas:

$$\hat{\mathbf{x}} = \mathbf{K}^{-1}\mathbf{x} \tag{13.14}$$

con lo que la ecuación (13.7) queda expresada como:

$$\hat{\mathbf{x}}_2^{\mathrm{T}}\left(\left[\mathbf{t}\right]_{\times}\mathbf{R}\right)\hat{\mathbf{x}}_1 = \hat{\mathbf{x}}_2^{\mathrm{T}}\mathbf{E}\hat{\mathbf{x}}_1 = 0 \quad , \quad \mathbf{E} = \mathbf{K}_2^{\mathrm{T}}\mathbf{F}\mathbf{K}_1 = \left[\mathbf{t}\right]_{\times}\mathbf{R} \tag{13.15}$$

La matriz E se denomina matriz esencial y también define la restricción epipolar, pero en unas coordenadas de imagen diferentes. Al haber eliminado el efecto de los parámetros intrínsecos de las cámaras, la matriz esencial sólo depende de la posición relativa de una cámara respecto a la otra.

La matriz esencial sólo tiene 5 grados de libertad. De la ecuación (13.15) se deduce que tanto la traslación como la matriz de rotación aportan 3 grados de libertad cada una, pero como se trata de una relación entre coordenadas homogéneas, la matriz está definida salvo por un factor de escala. Una propiedad interesante de la matriz esencial es que además de ser de rango 2, sus valores singulares no nulos son iguales.

Es posible obtener la posición relativa de una cámara respecto de la otra a partir de la factorización de la matriz esencial.

Algoritmo 0.3. *Obtención de la posición relativa de las cámaras a partir de la matriz esencial.*

Objetivo: Conocida **E**, obtener la rotación **R** y la traslación **t** de una cámara respecto de la otra mediante la factorización de la matriz esencial.

Solución:

1. Calcular la descomposición en valores singulares de **E**: $\mathbf{E} = \mathbf{U}diag(\sigma, \sigma, 0)\mathbf{V}^{\mathrm{T}}$
2. La matriz de proyección de la primera cámara es: $\mathbf{P}_1 = [\mathbf{I}_{3\times3}|\mathbf{0}_{3\times1}]$.
3. Entonces las cuatro posible s descomposiciones en **R** y **t** son:
   a. $\mathbf{P}_2 = [\mathbf{U}\mathbf{W}\mathbf{V}^{\mathrm{T}}| + \mathbf{u}\_3]$
   b. $\mathbf{P}_2 = [\mathbf{U}\mathbf{W}\mathbf{V}^{\mathrm{T}}| - \mathbf{u}\_3]$
   c. $\mathbf{P}_2 = [\mathbf{U}\mathbf{W}^{\mathrm{T}}\mathbf{V}^{\mathrm{T}}| + \mathbf{u}\_3]$
   d. $\mathbf{P}_2 = [\mathbf{U}\mathbf{W}^{\mathrm{T}}\mathbf{V}^{\mathrm{T}}| - \mathbf{u}\_3]$

   con $\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
4. De las cuatro posibles soluciones se debe seleccionar aquella que sitúe los puntos reconstruidos delante de ambas cámaras. En ausencia de información adicional, la escala de

la traslación no puede ser determinada. Con la solución propuesta, la traslación queda normalizada a 1, por lo que todas las medidas posteriores quedarían escaladas por la distancia base real.

13.2.3. Reconstrucción de la escena

El problema de la reconstrucción de la escena consiste en determinar la posición de un punto a partir de sus proyecciones en dos o más imágenes y de los modelos de proyección. La reconstrucción se puede hacer a tres niveles, dependiendo del tipo de información de que dispongamos sobre las cámaras:

1. Proyectiva: se caracteriza porque la escena real y la reconstrucción obtenida están relacionadas por una transformación proyectiva genérica. Existe una matriz invertible de dimensión 4x4 tal que $\widehat{\mathbf{X}} = \mathbf{H}\mathbf{X}$, donde $\widehat{\mathbf{X}}$ y $\mathbf{X}$ son las coordenadas homogéneas del punto reconstruido y del punto real respectivamente. Esta situación aparece cuando los parámetros internos de la cámara no son conocidos y las matrices asociadas a cada cámara han sido obtenidas a partir de la matriz fundamental.
2. Métrica: se caracteriza porque la relación entre la escena reconstruida se diferencia de la real por un factor de escala que no puede ser determinado. En general, tendremos que $\widehat{\mathbf{X}} = \lambda\mathbf{X}$, donde λ es un escalar no nulo e igual para todos los puntos de la imagen. Esta situación aparece cuando conocemos los parámetros internos de las cámaras y las matrices asociadas a cada una han sido obtenidas a partir de la matriz esencial.
3. Euclídea: la escena ha podido ser reconstruida correctamente incluyendo la escala. Este caso se presenta cuando todas las cámaras han sido calibradas respecto a un patrón común a todas ellas, por lo que las posiciones relativas son completamente conocidas.

Sea cual sea el caso, el procedimiento a seguir es el mismo en todos los casos. La idea más simple consiste en buscar el punto de intersección de los rayos que reproyectan el punto desde cada imagen. Sin embargo, debido al ruido y a los errores acumulados en la localización de los puntos o el modelo de las cámaras, los rayos no se cortarán en un punto sino que se cruzarán.

Se puede plantear una solución lineal que permite resolver el problema de la reconstrucción como la solución de un sistema de ecuaciones homogéneo y que tiene la característica de que se generaliza fácilmente al caso de disponer de más de dos cámaras. Se parte de la ecuación que representa el modelo proyectivo de una cámara:

$$\mathbf{x} = \mathbf{P}\mathbf{X} \tag{13.16}$$

donde el subíndice *i* representa la cámara y $\mathbf{X}$ es el punto cuya posición en el espacio se desea conocer. Los valores de $\mathbf{x}$ y $\mathbf{P}$ son conocidos. La igualdad anterior indica que los vectores a ambos lados de la igualdad son paralelos por lo que su producto vectorial será nulo:

$$\mathbf{x} \times \mathbf{P}\mathbf{X} = [\mathbf{x}]_{\times} \mathbf{P}\mathbf{X} = \mathbf{0} \tag{13.17}$$

como $[\mathbf{x}]_x$ es una matriz de rango 2, por lo que la expresión anterior sólo aporta dos ecuaciones independientes. Acumulando las ecuaciones obtenidas para cada imagen resulta el sistema:

$$\begin{bmatrix} \text{M} \\ x\mathbf{p}^{3T} - \mathbf{p}^{1T} \\ y\mathbf{p}^{3T} - \mathbf{p}^{2T} \\ \text{M} \end{bmatrix} \mathbf{X} = \mathbf{0} \tag{13.18}$$

En la ecuación anterior (x, y) son las coordenadas de la proyección del punto en la imagen y $\mathbf{p}^{iT}$ es la fila *i* de la matriz de calibración de la cámara. **X** es el vector que expande el subespacio nulo por la izquierda de la matriz de coeficientes. Conviene recordar que **X** es un punto expresado en un espacio proyectivo de dimensión 3 y que, como se indicó al principio de la sección, el nivel de la reconstrucción obtenida dependerá del método seguido para obtener las matrices **P** de cada cámara.

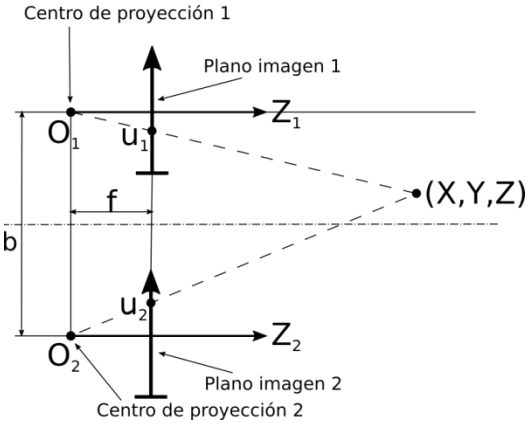


Figura 13.4. Esquema de un sistema estéreo en configuración canónica.

13.2.4. Configuración de cámaras en paralelo

Una configuración de especial interés para los sistemas estéreo es aquella formada por dos cámaras iguales, cuyos ejes ópticos son paralelos y la traslación entre sus centros de proyección es paralela a la línea de exploración de la cámara, es decir, a las filas de la imagen. Esta configuración se conoce como configuración estéreo canónica y se muestra esquemáticamente en la figura 13.4. Con esta configuración, los planos imagen son coincidentes y las líneas epipolares coinciden con las filas de las imágenes. Los epipolos están el punto del infinito correspondiente al eje **u**, y por tanto son de la forma $[1,0,0]^T$.

Dos puntos correspondientes sólo diferirán en el valor de la coordenada horizontal del punto. Dicha diferencia se conoce como disparidad y permite determinar totalmente la posición 3D del punto de la escena si se conoce la separación entre las cámaras (b) y la distancia focal (*f*). Si las coordenadas del punto 3D referidas al sistema de la cámara 1 son $[X,Y,Z]^T$, entonces, sus coordenadas en el sistema de la cámara 2 son $[X+b,Y,Z]^T$. Teniendo esto en cuenta, resulta que:

$$x_1 = \frac{fX}{Z} \quad , \quad x_2 = \frac{f(X+b)}{Z} \tag{13.19}$$

Eliminando X de las ecuaciones anteriores:

$$Z = \frac{fb}{(x_1 - x_2)} = \frac{fb}{d} \tag{13.20}$$

Una vez calculada la profundidad, las coordenadas X e Y pueden obtenerse sin más que aplicar las ecuaciones del modelo de proyección. De la ecuación (13.20) se deduce que la profundidad del punto y la disparidad son inversamente proporcionales, es decir, cuanto más lejos esté el punto menor será la discrepancia en su posición en la imagen. Siguiendo el mismo razonamiento, la disparidad debe ser siempre una cantidad positiva y su valor máximo está determinado por la distancia de la cámara al objeto más cercano.

13.2.5. Rectificación de imágenes

El objetivo de los algoritmos de rectificación de imágenes es obtener una transformación proyectiva para cada imagen de modo que los pares de líneas epipolares correspondientes sean colineales y paralelos a uno de los ejes de la imagen, típicamente el horizontal. Con esta transformación se simplifican enormemente las búsquedas a lo largo de las líneas epipolares, lo que lleva a la obtención de implementaciones más eficientes.

Fusiello y col. (2000) han propuesto un algoritmo sencillo para obtener las matrices que permiten rectificar un par estéreo cuando se conoce la calibración del par estéreo:

Algoritmo 0.4. *Rectificación de imágenes.*

Objetivo: Obtener dos transformaciones $\mathbf{T_1}$ y $\mathbf{T_2}$ que rectifican las imágenes de un par estéreo calibrado.

Solución:

1. Factorizar las matrices del par estéreo calibrado: $\mathbf{P_i} = [\mathbf{Q_i}|\mathbf{q_i}] = \mathbf{K_i}[\mathbf{R_i}|\mathbf{t_i}]$
2. Obtener la posición de los centros de proyección: $\mathbf{c_i} = -\mathbf{Q_i^{-1}}\,\mathbf{q_i}$
3. Los nuevos ejes son:
   a. Eje x paralelo a la traslación de las cámaras: $\mathbf{v_1} = (\mathbf{c_1} - \mathbf{c_2})/\|\mathbf{c_1} - \mathbf{c_2}\|$
   b. Eje y perpendicular al eje óptico de la cámara 1 y a $\mathbf{v}_1$: $\mathbf{v_2} = \mathbf{R_1^{3T}} \times \mathbf{v_1}$
   c. Eje z perpendicular a $\mathbf{v}_1$ y $\mathbf{v}_2$: $\mathbf{v\_3} = \mathbf{v_1} \times \mathbf{v_2}$
   d. La matriz de rotación para el par rectificado es: $\mathbf{R} = \begin{bmatrix} \mathbf{v_1^T} \\ \mathbf{v_2^T} \\ \mathbf{v_3^T} \end{bmatrix}$
4. La nueva matriz de parámetros internos se puede elegir arbitrariamente, en este caso se toma: $\mathbf{K} = \frac{\mathbf{K_1} + \mathbf{K_2}}{2}$
5. Las matrices de proyección asociadas a las imágenes rectificadas son: $\mathbf{P_{ni}} = \mathbf{K}[\mathbf{R}|-\mathbf{R}\mathbf{c_i}]$
6. Las matrices de transformación son de la forma: $\mathbf{T}_i = (\mathbf{AR}) * (\mathbf{K}_i\mathbf{R}_i)^{-1}$

7. Aplicar la transformación $\mathbf{T}_i$ a la imagen $\mathbf{I}_i$.

Figura 13.5. Resultado de aplicar el algoritmo 13.4 para la rectificación de un par estéreo. (a) Par estéreo original y (b) para estéreo rectificado.

En la figura 13.5 puede observarse el resultado de aplicar el algoritmo 13.4 a un par estéreo. En la parte superior se observan las imágenes originales y la fila inferior muestra ambas imágenes una vez rectificadas. Este algoritmo falla cuando el eje óptico es paralelo a la línea base.

13.3. Establecimiento de correspondencias

Para poder reconstruir la escena es necesario conocer qué puntos de cada imagen son proyecciones de un mismo punto de la escena. Hasta ahora se ha supuesto que dicha información es conocida y se ha usado como información de partida para poder establecer las relaciones geométricas a partir de las que llevar a cabo la reconstrucción. El establecimiento de correspondencias se basa en dos suposiciones principales: la mayor parte de los puntos de la escena son visibles en ambas imágenes y las distintas vistas de un mismo elemento de la escena son similares. El resto de este capítulo se dedica al problema de cómo identificar correspondencias entre imágenes de forma automática.

Lo primero que se debe decidir es qué elementos se van a emparejar: píxeles individuales, regiones, puntos característicos o contornos. A priori, los puntos característicos son una buena opción puesto que por definición, son puntos que se diferencian bastante de sus vecinos. El uso de descriptores robustos ante transformaciones afines de la imagen, como SIFT (Scale-Invariant Feature Transform (Lowe, 2004)) o SURF (Speeded-Up Robust Features (Bay y col., 2008)), facilita el emparejamiento. Por ello, esta opción es la que se emplea inicialmente para poder estimar la geometría epipolar de una escena. Los algoritmos tipo RANSAC permiten además eliminar correspondencias erróneas por lo que un algoritmo de reconstrucción estéreo seguiría el siguiente esquema:

Algoritmo 0.5. *Reconstrucción 3D dispersa.*

Objetivo: Recuperar la posición 3D de puntos de presentes en un par imágenes de una misma escena.

Solución:

1. Identificar puntos de interés en cada imagen. Se pueden emplear algoritmos como el detector de Harris, el detector de SIFT o el de SURF entre otros.
2. Determinar emparejamientos entre los dos conjuntos de puntos. Un algoritmo de emparejamiento simple consiste en emparejar cada punto de la primera imagen con el punto de la segunda que tenga el descriptor más parecido.
3. Determinar la geometría epipolar a partir de las correspondencias identificadas en el paso 2, usando un algoritmo tipo RANSAC para eliminar las posibles correspondencias erróneas.
4. Con los *inliers* detectados, reconstruir la posición 3D de los puntos mediante triangulación.
5. Se puede refinar la solución buscando nuevos emparejamientos. Para ello, buscaremos correspondencias para los puntos característicos descartados en el paso 3, restringiendo la búsqueda a la línea epipolar.

El principal inconveniente de esta solución es que lleva a una reconstrucción dispersa, ya que son pocos los puntos de la imagen que se emparejan y se reconstruyen. Por otro lado, presenta la ventaja de que no es necesario tener ningún tipo de información adicional sobre el par estéreo y es bastante robusta frente a la posición relativa de las cámaras.

13.4. Mapas densos de disparidad

Para poder reconstruir la escena en su conjunto es necesario establecer correspondencias entre características que proporcionen un recubrimiento más denso de la escena. Típicamente se recurre al emparejamiento directo entre los píxeles de ambas imágenes. Sin información adicional, el algoritmo básico consistirá en tomar cada punto la imagen 1 y buscar a lo largo de su correspondiente línea epipolar en la imagen 2 con el mejor candidato. Para facilitar la discusión, en lo que resta del capítulo se considerará que se tiene un par estéreo en configuración canónica o que las imágenes han sido rectificadas.

13.4.1. Métodos basados en correlación

Resulta evidente que la decisión para seleccionar el mejor candidato de entre todos los píxeles de la línea epipolar no puede basarse en la simple comparación del color. Una solución tan simple se enfrenta con dos problemas:

1. El ruido presente en la imagen y las variaciones en el color o el nivel de gris como consecuencia del cambio de perspectiva.
2. Existirán muchos candidatos con las mismas características por lo que tendríamos un problema de ambigüedad al no disponer de un criterio para elegir entre los empates.

Como en muchos otros casos, la solución pasa por comparar las vecindades (W) de los puntos candidatos a formar una pareja. Para determinar la fuerza del emparejamiento se pueden emplear varios tipos de métricas. Las más habituales utilizan las imágenes normalizadas:

$$\begin{aligned}
&\bar{\mathbf{I}}_W(x,y) = \tfrac{1}{|W|}\sum_{(i,j)\in W}\mathbf{I}(i,j) \quad \text{Valor medio de gris en la vecindad del píxel}\\
&\|\mathbf{I}\|_W = \sqrt{\sum_{(i,j)\in W}\mathbf{I}^2(i,j)} \quad \text{Magnitud del nivel de gris en la ventana}\\
&\hat{\mathbf{I}}(x,y,W) = \frac{\mathbf{I}(x,y)-\bar{\mathbf{I}}_W(x,y)}{\|\mathbf{I}-\bar{\mathbf{I}}_W\|_W} \quad \text{Imagen normalizada}
\end{aligned} \tag{13.21}$$

y se basan en minimizar la suma del error cuadrático normalizado de media cero (ZNSSD – Zero-mean Normalised Sum of Squared Differences) entre las vecindades de los puntos que se desean emparejar:

$$ZNSSD(x,y,d,W) = \sum_{(i,j)\in W}\left(\hat{\mathbf{I}}_1(i,j,W)-\hat{\mathbf{I}}_2(i-d,j,W)\right)^2 \tag{13.22}$$

donde x,y son las coordenadas del pixel en la imagen de referencia, d es la disparidad y W es la vecindad elegida para hacer la comparación. Otra alternativa es maximizar la correlación normalizada de media cero (ZNCC – Zero-mean Normalised Cross Correlation) entre dichas ventanas:

$$ZNCC(x,y,d,W) = \sum_{(i,j)\in W}\left(\hat{\mathbf{I}}_1(i,j,W)\hat{\mathbf{I}}_2(i-d,j,W)\right) \tag{13.23}$$

El resultado depende del tamaño de la vecindad elegida para hacer la correlación. Si ésta es demasiado pequeña y hay poca variación de intensidad en la misma, la estimación de la disparidad será pobre ya que la relación señal/ruido será baja. Sin embargo, al aumentar el tamaño de la ventana, aumenta la probabilidad de incluir puntos con diferentes planos y que por tanto violan la condición de que la disparidad es constante en toda la ventana, lo que produciría de nuevo una mala estimación de la disparidad. Kanade y Okutomi (1990) han propuesto un algoritmo que estima el tamaño de la ventana de forma adaptativa en función de la variación local del nivel de gris y de la disparidad.

13.4.2. Restricciones en los emparejamientos

Hasta ahora se han considerado dos restricciones fundamentales que se deben cumplir para que dos puntos puedan ser considerados una correspondencia correcta: la restricción epipolar y la restricción de similitud entre los elementos emparejados. Sin embargo existen varias restricciones adicionales que pueden ser usadas para mejorar los resultados de los emparejamientos:

Restricción de unicidad: cada elemento de una imagen sólo puede ser emparejado con un único elemento de la otra. Esta restricción es muy razonable cuando el elemento emparejado es un píxel de la imagen, pero es más discutible cuando los elementos que se quieren emparejar son segmentos o contornos, ya que éstos pueden aparecer cortados en alguna de las imágenes.

Restricción de continuidad: representa el hecho de que la materia es continua y, por tanto, la disparidad debe variar suavemente, excepto cuando cambie el objeto visualizado. Asociadas a estas zonas aparecerán puntos de oclusión, es decir, puntos que sólo son visibles en una de las imágenes.

Restricción de orden: esta restricción implica que dado un conjunto de emparejamientos a lo largo de una línea epipolar, el orden relativo de los elementos de la imagen 2 debe ser el mismo que el de los elementos de la imagen 1 y viceversa. Esta restricción no se cumple cuando existen objetos estrechos o muy próximos al sistema estereoscópico en primer plano. Afortunadamente este no es un caso habitual.

13.4.3. Métodos basados en programación dinámica

El método presentado en la sección 13.4.1 se basa en una búsqueda voraz de emparejamientos. Es un método de búsqueda sencillo, pero que no garantiza un óptimo global para toda la línea epipolar, ni el cumplimiento de condiciones adicionales como la restricción de unicidad, la de continuidad o la de orden. Para poder hacer uso de este tipo de restricciones, es necesario optimizar una función que tenga en cuenta simultáneamente todos los emparejamientos realizados.

Una posibilidad, si se cumple la restricción de orden, consiste en el uso de algoritmos basados en programación dinámica. La programación dinámica es un método de optimización introducido por Bellman (1954) que se caracteriza por:

1. La solución óptima del problema puede obtenerse mediante el encadenamiento de un conjunto de subproblemas más sencillos.
2. La solución al problema de optimización consiste en una secuencia de decisiones de modo que dado un punto de la solución, la secuencia de decisiones que resuelve el resto del problema no depende de las decisiones tomadas previamente. Este principio se conoce como el principio de optimalidad de Bellman.

Bajo la restricción de orden, si el emparejamiento $\mathbf{x}_1 \leftrightarrow \mathbf{x}_2$ es correcto, los puntos a la izquierda de $\mathbf{x}_1$ deben estar emparejados con los puntos a la izquierda de $\mathbf{x}_2$ y lo mismo sucederá con los puntos situados a su derecha. Esta característica es la que hace que se cumpla el principio de optimalidad de Bellman.

El caso general consiste en calcular la solución óptima al emparejamiento de los *m* primeros puntos de la imagen 1 con los *n* primeros puntos de la imagen 2. Para ello es necesario considerar tres posibles casos:

1. Los puntos *m* y *n* forman un emparejamiento correcto: el coste de emparejar ambas líneas epipolares será el coste asociado a la nueva correspondencia más el coste de emparejar los *m*-1 puntos anteriores de la imagen 1 con los *n*-1 puntos de la imagen 2.
2. El punto *m* no está entre los *n* primeros puntos de la imagen 2: el coste total será el coste de emparejar los *m*-1 primeros puntos de la imagen 1 con los *n* primeros puntos de la imagen 2 más el coste de considerar la oclusión.
3. El punto *n* no está entre los *m* primeros puntos de la imagen 1: el coste total será el coste de emparejar los *m* primeros puntos de la imagen 1 con los *n*-1 primeros puntos de la imagen 2.

$$C_T(m,n) = \min\left(C_T\left(m-1,n-1\right)+c(m,n), C_T\left(m-1,n\right)+c(m,\infty), C_T\left(m,n-1\right)+c(\infty,n)\right) \quad (13.24)$$

El caso trivial consiste en emparejar los puntos de una de las imágenes con una línea vacía. En este caso todos los puntos están ocultos. Para obtener la solución es necesario construir la matriz de costes, para después volver hacia atrás recuperando las decisiones que terminaron conduciendo a la solución óptima. La figura 13.6 muestra el resultado de aplicar el algoritmo 13.6 a un par de imágenes estéreo de una pieza de automóvil. Puede observarse cómo las zonas con menos textura producen resultados menos fiables, mientras que las zonas con más textura el resultado es claramente mejor. Las zonas ocultas se han marcado con disparidad cero.

Algoritmo 0.6. *Obtención de emparejamientos mediante programación dinámica.*

Objetivo: Obtener el conjunto óptimo de emparejamientos entre dos líneas epipolares de un par estéreo. Aunque el procedimiento es recursivo, en este caso la solución puede ser planteada fácilmente como una solución iterativa. Como datos se parte de los valores de las líneas epipolares, el número de puntos en cada una (l_i) y el coste asociado a la oclusión.

Solución:

1. Inicializar la matriz de costes para los casos triviales:

$$C_T(0,0) = oclusión;$$
$$C_T(m,0) = C_T(m-1,0) + oclusión;\ m = 1 \dots l_1$$
$$C_T(0,n) = C_T(0,n-1) + oclusión;\ n = 1 \dots l_2$$

2. Calcular la matriz de costes, recorriendo ambas líneas epipolares en sentido ascendente:

   Para m=1 hasta l_1

   Para n=1 hasta l_2

   calcular $C_T(m,n)$ aplicando la ecuación (13.24)

   guardar en $M(m,n)$ la decisión que produjo el óptimo.

3. Recuperar los emparejamientos que producen la solución óptima

   $d_1(m) = -1; m = 1 \dots l_1$

   m=l_1, n=l_2

   Mientras (m!=0 y n!=0)

   Según $M(m,n)$

   sea 1: $d_1(m) = m - n$; m--; n--; % emparejamiento correcto

   sea 2: $d_1(m) = -1$; m--; % el punto sólo es visible en la imagen 1

   sea 3: n--; % el punto sólo es visible en la imagen 2

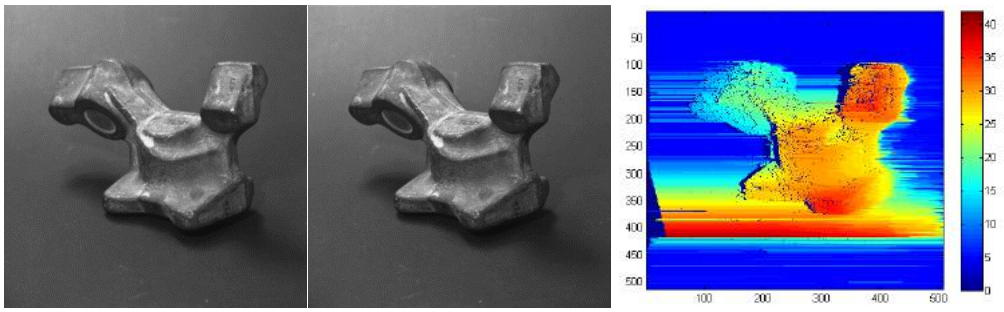


Figura 13.6. Par estéreo de una pieza de automóvil y resultado del cálculo de disparidades mediante programación dinámica (Fuente: "USC Institute for Robotics and Intelligent Systems, Gerard Medioni" http://vasc.ri.cmu.edu//idb/html/stereo/parts/index.html).

13.5. Bibliografía

Bay, H.; Ess, A.; Tuytelaars, T.; Van Gool, L. (2008). Speeded-up robust features (SURF). *Computer vision and image understanding*, *110*(3), 346–359.

Bellman, R. (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society*, *60*(6), 503-515.

Fischler, M. A.; Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395.

Fusiello, A.; Trucco, E.; Verri, A. (2000). A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, *12*(1), 16–22.

Hartley, R. I. (1997). In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(6), 580-593.

Kanade, T.; Okutomi, M. (1994). A stereo matching algorithm with an adaptive window: Theory and experiment. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *16*(9), 920-932.

Lounguet-Higgins, H. C. (1981). A computer algorithm for reconstructing a scene from two projections. *Nature*, *293*(5828), 133-135.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.

CAPÍTULO 14

RECONOCIMIENTO DE OBJETOS 3D CON DESCRIPTORES DE SUPERFICIE

Pablo GIL[1], Carlos M. MATEO[2], Jorge POMARES[1], Gabriel J. GARCIA[1], Fernando TORRES[1]

[1] Depto. Física, Ingeniería de Sistemas y Teoría de la Señal, Universidad de Alicante, Alicante, España

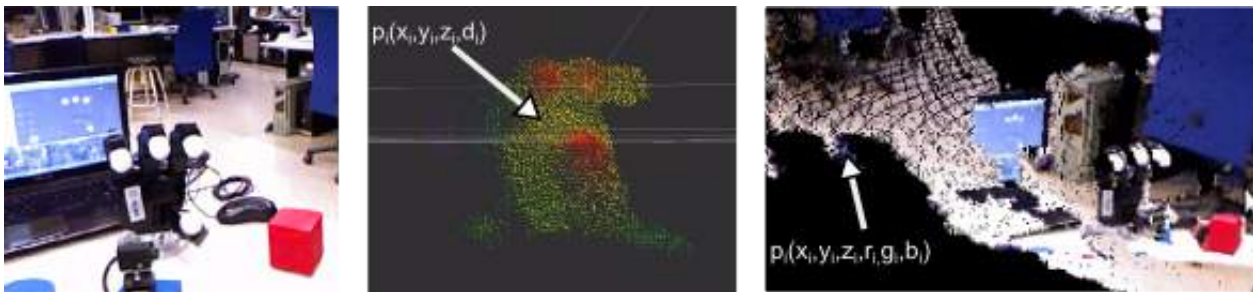
[2] Instituto Universitario de Investigación Informática, Universidad de Alicante, Alicante, España

Este capítulo presenta un conjunto de descriptores basados en información de superficie para reconocer objetos asumiendo que se dispone de un modelo previo de éstos. Las superficies son representadas como nubes de puntos tridimensionales. De éstas se estudian características geométricas como la variación en dirección de los vectores normales y las curvaturas a lo largo de la nube de puntos que las definen, para así construir un descriptor matemático que identifique el tipo de objeto según su superficie. En este capítulo se proporcionan fundamentos geométricos de superficie, metodologías de filtrado para eliminación de ruido y puntos atípicos, características para describir una superficie, métricas de clasificación por distancia. Además de un conjunto de técnicas y métodos de procesamiento básicos que constituirán las etapas necesarias que se precisan implementar para identificar objetos 3D, sin la necesidad de considerar parámetros como la textura o el color.

14.1. Representación de una escena. La nube de puntos

La estructura básica para representar una superficie en el Espacio Euclídeo se conoce como nube de puntos 3D. En la actualidad, existen una gran cantidad de sensores visuales que son capaces de adquirir información de profundidad de una escena real. Entre estos sensores han adquirido gran popularidad, las cámaras RGB-D y las cámaras de tiempo de vuelo ('ToF: *Time of Flight*' en el ámbito anglosajón) (Gil y col. 2014). Estos sensores se basan en principios físicos y tecnologías distintas, pero en ambos tipos es posible almacenar la información adquirida en forma de nubes de puntos, figura 14.1.

En su definición más sencilla, una nube de puntos **P** es un conjunto de puntos 3D $p_i(x_i, y_i, z_i)$ (Klette y col. 1998). El número de puntos que forman **P** depende de la resolución y la precisión con la que la cámara es capaz de adquirir información del entorno. Generalmente, cada uno de los puntos $p_i \in \mathbf{P}$ tiene un orden, ya que está referenciado respecto a un sistema de coordenadas fijo situado en el mismo sensor con el que se ha adquirido la información. Así, las coordenadas (x_i, y_i, z_i) de un punto p_i representan la posición que ocupa éste respecto al sensor con el que fue adquirido. Para que la cámara obtenga de una manera fiable las coordenadas de cada punto p_i requiere haber sido calibrada previamente. El proceso de calibración (Cyganek y Siebert 2009) permite obtener los parámetros intrínsecos y extrínsecos de un sensor.



(*a*) (*b*)

Figura 14.1. Ejemplo de nube de puntos adquirida desde una cámara: (*a*) ToF (*b*) RGBD

Es habitual que las estructuras como nubes de puntos permitan almacenar información adicional además de la posición tridimensional de los puntos. Esta información adicional, frecuentemente, corresponde a las propiedades que los puntos tienen en la escena real. En estos casos, cada p_i tiene más de 3 dimensiones. Así, por ejemplo, si la nube de puntos está generada a partir de un sensor RGBD cada punto consta de 6 dimensiones $(x_i, y_i, z_i, r_i, g_i, b_i)$ si se opta por almacenar la información de color. Esto se puede generalizar, ya que en una nube de puntos podrían almacenarse otras muchas propiedades de un punto de la escena como por ejemplo posición, color, geometría, etc. De modo que se podría considerar que cada punto $p_i \in \mathbf{P}$ se puede representar como $p_i(f_1, \ldots, f_n)$ donde f_i es cada una de las características del punto y n el número de estas.

Las nubes de puntos son estructuras que por su tamaño y por las dimensiones de cada uno de sus elementos requieren, en muchos casos, de una gran cantidad de memoria. Además, su volumen de datos y su manera de organización incrementa los tiempos y costes computacionales, sobre todo cuando se aplican etapas de procesamiento para tratar y analizar los datos que contiene. Un ejemplo para ilustrar este problema sería la búsqueda del punto p_j en la escena más próximo a otro punto p_i , representados ambos en la nube de puntos **P** como p_j y p_i, respectivamente. En la escena, la proximidad o lejanía entre dos puntos 3D cualesquiera se puede calcular con la distancia Euclídea, y además la geometría de la escena nos garantiza que, en el mundo, el punto más próximo P_j a un punto P_i escogido arbitrariamente siempre se encontrará en el entorno de vecindad de P_i. Este hecho no es trasladable a la nube de puntos, es decir, en **P** estos dos elementos p_i y p_j pueden estar separados. De modo, que los puntos que están situados como vecinos p_i no tienen por qué ser los más próximos en distancia Euclídea en la escena. Esto hace que, por lo general, no se trabaje directamente sobre la nube de puntos y se opte por almacenar ésta como una estructura de datos más eficiente para hacer operaciones sobre ella. Las estructuras de datos más empleadas para trabajar con nubes son los árboles.

Es frecuente emplear estructuras de datos como '*kd-tree*', '*octree*' (Eberhard y col. 2010) y otras variantes para organizar y descomponer espacialmente **P**.

Por otro lado, en ocasiones la nube de puntos **P** determina la superficie de un objeto en vez de una escena más compleja, en estos casos es habitual que se empleen técnicas de reconstrucción de superficies. Por ejemplo se pueden emplear métodos de triangulación para construir una malla de triángulos que constituyan una aproximación poligonal de toda la superficie del objeto, u otros polígonos planos o formas geométricas.

14.1.1. El concepto de entorno de vecindad y similitud entre puntos de un entorno

En una nube de puntos **P** se puede definir un entorno de vecindad de un punto $p_q \in \mathbf{P}$ con radio r como el conjunto de todos los puntos $p_i \in \mathbf{P}$ tal que $i \neq q$:

$$\|p_q - p_i\| \leq d_r \tag{14.1}$$

donde d_r es la distancia máxima que limita el radio especificado, figura 14.2. Para medir esa distancia, puede utilizarse distintas métricas, la más popular es la métrica L_2 también conocida como distancia Euclídea.

Cuando se emplea una nube de puntos **P** para representar un objeto o escena, que se quiere reconocer no basta con considerar **P** como un conjunto de puntos con coordenadas Cartesianas (x, y, z). En estos casos, es necesario tener en cuenta la relación existente entre los puntos que componen la nube. Así, si se toma un punto $p_q \in \mathbf{P}$ y se fija un radio r que delimita su entorno de vecindad P^r, la relación entre él y sus vecinos p_i constituye un descriptor local en P^rque describe cómo varía la geometría y la forma de la superficie en p_q respecto a los puntos de su entorno P^r. La representación de esta información geométrica local se puede definir como un vector:

$$F_q\left(p_q, \mathrm{P}^r\right) = \{f_1, f_2, \ldots, f_n\} \tag{14.2}$$

donde f_i es la característica geométrica que relaciona p_q con cada p_i, siendo n el número de puntos en P^r.

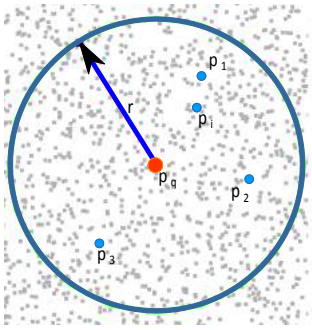


Figura 14.2. Ejemplo de entorno de vecindad de radio r en una nube de puntos

Si se desea comparar dos puntos cualesquiera p_i y p_j de una nube de puntos **P**, se procede tomando la información geométrica local de cada uno de ellos F_i y $F_j$en un entorno de vecindad P^r y comparando ésta. Generalmente, para llevar a cabo esta comparación se emplean métricas de distancia. Por consiguiente, es imprescindible escoger un entorno de vecindad adecuado para obtener una

representación del entorno del punto lo más significativa posible. La idea principal subyace en escoger el radio de modo que a partir de P^r sea posible obtener un vector de información geométrica local que no cambie (sea invariante) en presencia de transformaciones de cuerpos rígidos sólidos (rotaciones y traslaciones en el espacio Euclídeo, es decir en 6 grados de libertad), variación en la densidad de P^r (número de puntos que contiene) y ruido en los datos debido al proceso de adquisición (cierta falta de precisión a la hora de determinar las coordenadas Cartesianas de cada punto desde el sensor).

Por un lado, como ya se ha comentado con anterioridad, no existe una relación directa entre el concepto de proximidad entre dos puntos en la escena y entre esos dos puntos tal y como han sido registrados en **P**. En otras palabras, puntos cercanos en la escena no se garantiza que se hayan registrado cercanos en la nube de puntos. Esto implica que, si se quieren obtener los puntos cercanos a un punto p_q en **P**, se requiere calcular una métrica de distancia entre él y todos los puntos de la nube, además este proceso es costoso computacionalmente. Por otro lado, el proceso de registro de **P** puede introducir ruido, y hacer que el vecino más cercano a p_q no sea aquel que minimice la distancia. Esto a veces es más frecuente cuanto más densa es **P**. En estos casos, se aplican mecanismos de estimación para la aproximación en la búsqueda de los puntos p_i que pertenezcan al conjunto de vecinos más cercanos a p_q. Este mecanismo trabaja en función de un parámetro de error permitido ε que determina la calidad de la solución proporcionada, de la siguiente manera:

$$\|p_q - p_i\| \leq (1 + \varepsilon)\|p_q - p^*\| \tag{14.3}$$

donde p^* sería el punto real más cercano. Así, la ecuación 14.3 determina para cualquier punto p_i en un entorno de vecindad P^r, que la relación entre su distancia a p_q y la distancia de p_q al vecino real más cercano denotado por p^* debe ser de al menos ε+1.

Como consecuencia de todo esto, queda patente que hay dos parámetros que el usuario requiere definir cuando se trata de trabajar y procesar nubes de puntos: el radio r (define el tamaño a considerar como entorno de vecindad P^r) y el número n de puntos en ese radio r. Así, es posible fijar el radio en función del número mínimo de puntos que se desea que contenga P^r. Más adelante se mostrará cómo influyen valores grandes o pequeños de r o n a la hora de calcular el vector normal en un punto en función del tamaño y densidad de su entorno de vecindad.

14.1.2. Métricas de distancia entre puntos de una superficie

En particular, en un entorno de vecindad P^r, y en general, en una nube de puntos **P**, se hace imprescindible definir una métrica para medir la cercanía o lejanía de un punto de la superficie a otro cualquiera. Entre las métricas de distancia más comunes se encuentran: *Manhattan*, Euclídea, χ-Cuadrado y Núcleo de Intersección de Histograma (‘*HIK: Histogram Intersection Kernel’* en el ámbito anglosajón).

Sean $\bar{v}$ y $\bar{u}$ dos vectores de n dimensiones en un espacio vectorial, donde ambos están referenciados respecto al mismo sistema de coordenadas fijo.

Se define la distancia *Manhattan* o distancia L1, como:

$$d_{L1} = \|\bar{v} - \bar{u}\|_{L1} = \sum_{i=1}^{n} |v_i - u_i| \tag{14.4}$$

La distancia Euclídea o distancia L2, se calcula como:

$$d_{L2} = \|\bar{v} - \bar{u}\|_{L2} = \sqrt{\sum_{i=1}^{n} (v_i - u_i)^2} \tag{14.5}$$

Se define la distancia χ-Cuadrado, como:

$$d_{\chi^2} = \|\bar{v} - \bar{u}\|_{\chi^2} = \sum_{i=1}^{n} \frac{(v_i - u_i)^2}{v_i + u_i} \tag{14.6}$$

Y finalmente, la distancia *HIK*, como:

$$d_{HIK} = \|\bar{v} - \bar{u}\|_{HIK} = \sum_{i=1}^{n} \min\{v_i, u_i\} \tag{14.7}$$

Los puntos en una nube **P** se pueden considerar vectores de 3 o más dimensiones, tal y como se mostró previamente, dependiendo de si tienen únicamente información de posición espacial o incorporan información adicional como color, geometría, etc.

14.2. Procesamiento y extracción de características geométricas de superficie

14.2.1. Filtrado del ruido y puntos atípicos

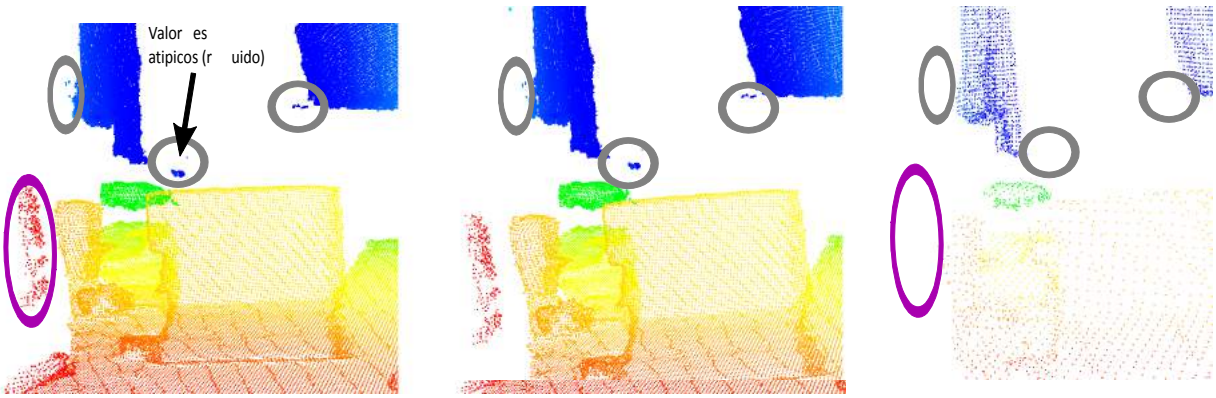
Cuando las nubes de puntos **P** han sido registradas a partir de un proceso de adquisición de una cámara RGBD o ToF, frecuentemente, incorporan puntos con ruido. El ruido de un punto dado p_q es la variación aleatoria de su posición (x_q, y_q, z_q). Este concepto es extensible a cada una de las n dimensiones del punto $p_q(f_1, \dots, f_n)$ de **P**. Así, si el punto es representado por un vector de 6 dimensiones $(x_q, y_q, z_q, r_q, g_q, b_q)$, el ruido sería la variación aleatoria en posición y color. Generalmente, se asume que si la nube de puntos se genera de manera sintética a partir de programas de modelado CAD, el problema del ruido desaparece. Principalmente, el ruido en cámaras RGBD o ToF se genera cuando la escena tiene alta reflectancia (superficies metálicas pulidas) o cuando la escena es compleja y hay varios objetos, en cuyo caso se producen transiciones entre superficies de objetos distintos que están situadas a distintas profundidades. Las transiciones se dan en los bordes donde hay salto de superficie o zona de oclusión por solapamiento y se caracterizan por estar representadas en **P** con bajo nivel de densidad de puntos. Una primera etapa de procesamiento puede consistir en eliminar de **P** todos aquellos puntos atípicos que están en zonas de transición o salto de superficie. De este modo, sólo se considerarían los puntos de la nube que están influenciados en menor

medida por el ruido del proceso de adquisición. Otra aproximación para eliminar puntos atípicos de la nube, consiste en llevar a cabo un análisis estadístico de puntos en sus entornos de vecindad. Primero, para cada punto p_q de **P**, considerar un entorno de vecindad P^r y calcular la distancia media μ_d y la varianza en distancia σ_d a sus n vecinos dentro del entorno de vecindad de radio r. Segundo, eliminar todos aquellos puntos cuya distancia está alejada de la distancia media del conjunto de puntos de P^r. Al igual que en la búsqueda del vecino más cercano, ecuación (14.3), es necesario definir un parámetro de tolerancia, similar a ε. Este parámetro delimitará el nivel de densidad en P^r, eliminando todos aquellos puntos que no cumplan la restricción de distancia media. Por lo tanto, el ruido siempre influirá negativamente si dado un punto p_q se desea calcular la relación existente con sus vecinos en un entorno P^r. Consecuentemente, el ruido afectará a la representación geométrica de la superficie local $F_q(p_q, P^r)$.

14.2.2. Remuestreo de una nube de puntos

El proceso de remuestreo de una nube de puntos se suele emplear cuando después de filtrar una nube de puntos **P**, el resultado es otra nube de puntos **P'** cuya densidad global de puntos es prácticamente la misma a la que tenía la nube sin filtrar. El proceso de remuestreo tiene como objetivo eliminar las imperfecciones, suavizando la superficie y descartando todos aquellos puntos que generan anomalías como picos o similares (concavidades abruptas), o rellenando pequeños agujeros (generados por falta de puntos debido a convexidades abruptas) en la superficie, figura 14.3.

El remuestreo se basa en técnicas de interpolación mediante optimización matemática y se emplea comúnmente para eliminar todos aquellos puntos atípicos de la nube. La técnica de optimización más sencilla es la conocida como Mínimos Cuadrados Promedio (*'LMS: Least Mean Square'*) (Trucco, 1998). Esta técnica busca una función continua que aproxime un conjunto de datos ordenados y minimice el error cuadrático medio. El método *LMS* requiere de un gran número de iteraciones para converger y obtener la función que aproxima y minimiza el error. Además, en este método se asume que el ruido de cada punto debe estar distribuido de forma aleatoria para que la aproximación sea correcta.



(a) *(b)* *(c)*

Figura 14.3. Ejemplo de filtrado y remuestreo. (*a*) Original (*b*) Nube de puntos eliminando ruido (*c*) Remuestreada eliminando puntos por suavizado de la superficie

Una variación de este tipo de ajuste es el método de Mínimos Cuadrados Ponderados o Mínimos Cuadrados Móviles (*'MLS: Moving Least Square'*) (Lancaster y Salkanskas 1981), que consiste en dar un peso a cada punto de la nube en función de que se considere que éste tenga más o menos error de medida. El método de MLS permite ajustar una superficie desde un conjunto de puntos de muestra no organizados o estructurados, asegurando que la superficie pasa por los puntos de la muestra. Para ello, mediante la asignación de pesos se da mayor o menor importancia a cada uno de los puntos de la muestra. Así, se pondera con mayor peso aquellos puntos de la muestra que son más representativos de la superficie.

En ocasiones, la nube de puntos **P** contiene superficies que no son completamente suaves (es decir con cambios no bruscos de dirección entre sus puntos), y que no son debidos a errores o anomalías en la etapa de registro, sino que son puntos que caracterizan el tipo de superficie, por ejemplo objetos que tienen regiones angulares e interesa preservar estas zonas angulosas sin que se consideren valores atípicos. Un objeto con forma de esfera se representa con una nube de puntos de superficie suave, mientras que un cubo presenta en la superficie cambios bruscos en la zona de aristas. En estos casos, suele emplearse para ajustar superficies planas RANSAC ('*RANSAC: RANdom SAmple Consensus*') (Trucco y Verri, 1998) o alguna de sus variantes como RMSAC ('*Randomized M-Estimator SAmple Consensus*'). Los métodos RANSAC y sus variantes proporcionan métodos de ajuste más robustos que mínimos cuadrados cuando hay muchos puntos atípicos, es decir datos con ruido, que pueden interferir en el ajuste. Además, puede interesar rellenar pequeños huecos, añadiendo puntos en aquellas áreas de la superficie donde hay ausencia de estos. La manera de hacerlo escapa al objetivo de este capítulo.

14.2.3. Vectores normales y curvatura de una superficie

Los conceptos de vector normal $\bar{n}$ y de curvatura c de una superficie han sido ampliamente usados, en aplicaciones de visión por computador, para proporcionar datos de medida de la superficie y definir propiedades geométricas de ésta (Klette y col. 1998). El término de curvatura, menos conocido que el de vector normal, tiene su origen en la geometría diferencial de curvas. En geometría diferencial, una curva es una región de puntos donde existe una misma variación de superficie. Y la manera más sencilla de medir esta variación es calculando los cambios de orientación de los vectores normales en dicha superficie. La medida de esa variación se conoce con el término de parámetro de curvatura en visión por computador, figura 14.4.

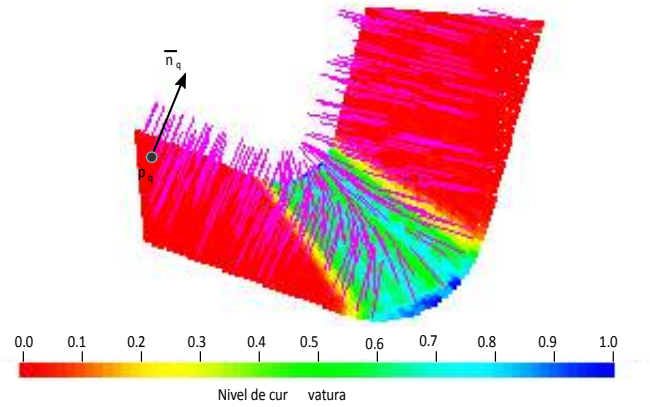


Figura 14.4. Ejemplo de cálculo de vectores normales y curvaturas. El color indica el valor de curvatura y las flechas los vectores normales

El cálculo de estas propiedades geométricas, $\bar{n}$ y c en un punto p_q de **P** es muy dependiente del entorno de vecindad P^r que se considera para dicho punto. Es decir, en el cálculo del vector normal y de la curvatura en p_q influyen tanto el radio como el número de puntos que contiene P^r. Así, si hay pocos puntos dentro del entorno de vecindad o si el radio es muy pequeño, el cálculo de estas propiedades geométricas se llevará a cabo con inexactitud. Por el contrario, muchos puntos o un radio muy amplio distorsionan los valores obtenidos dificultando la detección de cambios de orientación de los vectores normales y, como consecuencia, se genera falta de precisión en el cálculo de la curvatura. En este último caso, el valor de curvatura tiende a igualarse como si se realizará un suavizado.

Tal y como se ha comentado en la sección 14.1, si dada una nube de puntos **P** se desea almacenar información geométrica además de la posición cartesiana, el punto se denotaría por $p_q(f_1, f_2, f_3)$ y se requeriría trabajar en un entorno de vecindad P^r definido por el usuario. En el caso que nos ocupa, f_1 es una tupla que representa la posición cartesiana de cada punto (x_i, y_i, z_i), f_2 representa el vector normal a la superficie en dicho punto, y f_3 el parámetro de variación en la superficie o curvatura en P^r.

Una manera geométrica sencilla de calcular el vector normal a una superficie P^r en un punto $p_q(x_q, y_q, z_q)$ consiste en estimar el vector normal $\bar{n}$ a un plano tangente a la superficie en dicho punto. El plano tangente a P^r en p_q se define matemáticamente como $\Pi_q(x, y, z) = a(x - x_q) + b(y - y_q) + c(z - z_q) = 0$ y su vector normal es el vector gradiente al plano denotado por $\bar{n} = \nabla\Pi(x, y, z) = \{a, b, c\}$. El problema reside en que hay infinitos planos que pasan por p_q y se desea estimar el plano tangente a p_q que mejor aproxima todos los puntos de P^r, es decir, que se ajusta minimizando la distancia entre él y cada uno de los puntos de P^r. Por lo tanto, el problema se reduce a un problema de ajuste por mínimos cuadrados del sistema $d_i = 0$ donde d_i representa la distancia entre $p_i \in P^r$ y Π_q, y ésta se define como $d_i = (p_i - p_q) \cdot \bar{n}$.

Para obtener el vector normal $\bar{n}$ en p_q se procede realizando un análisis de valores y vectores propios de la matriz de covarianza C_q de todos los puntos $p_i \in P^r$ (Mateo y col 2015). La matriz de covarianza C_q se define aplicando Análisis de Componentes Principales (*'PCA: Principle Component Analysis'*) (Cyganek y Siebert 2009) del siguiente modo:

$$C_q = PP^T = \begin{bmatrix} p_1 - \bar{p} \\ \cdots \\ p_k - \bar{p} \end{bmatrix} \begin{bmatrix} p_1 - \bar{p} \\ \cdots \\ p_k - \bar{p} \end{bmatrix}^T \tag{14.8}$$

donde $\bar{p}$ es el centroide de P^r y k su número de puntos. Y resolviendo la ecuación (14.8) por descomposición en valores singulares ('*SVD: Singular Value Descomposition*') (Trucco y Verri 1998) se obtienen los valores propios λ_j y vectores propios $\bar{v}_j$ de C_q:

$$C \cdot \bar{v}_j = \lambda_j \cdot \bar{v}_j \tag{14.9}$$

La suma $\lambda_0 + \lambda_1 + \lambda_2$ de los valores propios proporciona un parámetro que describe la variación local de superficie que se detecta entre los puntos $p_i \in \mathrm{P}^r$ y su centroide $\bar{p}$. El valor propio más pequeño λ_0 proporciona información sobre la variación a lo largo del vector normal $\bar{n}$, y su vector propio $\bar{v}_0$ corresponde a una estimación de $\bar{n} = \{a, b, c\} = \{n_x, n_y, n_z\}$, siendo desconocido si el sentido del vector normal es positivo o negativo. Para resolver esta ambigüedad se requiere conocer el punto de vista con el que fue adquirida la nube de puntos P. El vector normal se puede representar en coordenadas esféricas como:

$$\emptyset = \operatorname{atan}\left(\frac{n_z}{n_y}\right), \theta = \operatorname{atan}\left(\frac{\sqrt{\left({n_z}^2 + {n_y}^2\right)}}{n_x}\right) \tag{14.10}$$

Hay varios estudios que permiten modelar el parámetro de curvatura cuando la superficie se modela con estructuras poligonales triangulares y no como nubes de puntos. No obstante, la gran mayoría de ellos calculan la curvatura a partir de puntos de vértice. Este es el caso de la curvatura Gaussiana. Para el caso que nos ocupa, una manera de estimar la curvatura máxima local a P^r en un punto p_q es el método de Pauly (Pauly y col. 2002) que tiene la ventaja de que es invariante a escala y donde la curvatura puede ser calculada a partir de los valores propios de la matriz de covarianza C_q del siguiente modo:

$$c_p = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{14.11}$$

Un valor pequeño de c_p indica que no hay variación en la superficie y por lo tanto, el punto yace sobre el plano tangente a la superficie.

14.2.4. Marco de Darboux

Cuando se trabaja con nubes de puntos, conviene calcular la relación geométrica que existe entre cada uno de los puntos de la nube y sus vecinos. En estos casos, el marco de Darboux constituye una herramienta matemática que permite calcular esta relación. Así, el marco de Darboux es una herramienta matemática de la geometría diferencial de superficies orientadas, que define tres vectores para representar matemáticamente una curva de la superficie. Se entiende como curva al lugar geométrico en la superficie de las distintas posiciones que ocupan los puntos en el espacio de acuerdo a

una trayectoria dada. Los tres vectores que definen la curva según Darboux son el vector normal, el vector tangente y el producto vectorial de éstos.

Si se traslada este concepto a un punto p_q de un entorno de vecindad de la superficie representada por P^r y se considera una curva sobre ella, el marco de Darboux se define como $\{\bar{u}, \bar{v}, \bar{w}\} = \{\bar{n}_q, \bar{t}_q, \bar{n}_q \wedge \bar{t}_q\}$ donde $\bar{n}_q$ es el vector normal unitario a la superficie en p_q y $\bar{t}_q$ es el vector tangente a la curva en p_q, figura 14.5. Así, el marco de Darboux define un sistema de referencia, como una base ortonormal en cada punto de la curva que representa un marco móvil natural para medir variaciones a lo largo de una curva. Este sistema de referencia está relacionado con la regla de la mano derecha, donde $\bar{u}$ representa el vector que apunta hacia nosotros, $\bar{w}$ está girado 90º en sentido horario desde el vector $\bar{v}$.

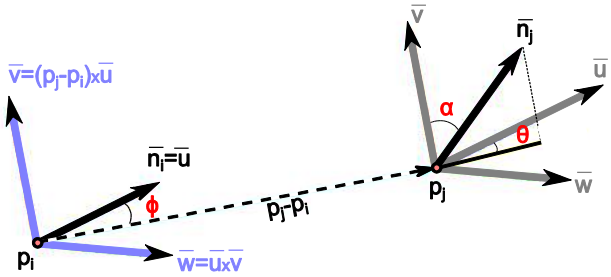


Figura 14.5. Variación geométrica entre dos puntos en función del marco de Darboux

14.3. Descriptores geométricos de superficie

En ocasiones, una nube de puntos puede representar un objeto 3D aislado. En procesos de reconocimiento de objetos 3D, es común crear descriptores como conjuntos de propiedades que los identifiquen, sección 14.4. Estos descriptores permiten realizar procesos de comparación de dos o más objetos mediante un análisis de semejanza de las propiedades de los objetos. En este apartado se pretende ilustrar algunos descriptores que ayudan a representar propiedades geométricas de la superficie de un objeto.

Los valores de los descriptores geométricos de un objeto 3D dependen de cómo se encuentran representadas la superficie o volumen de éste después de ser registrado por el sistema sensorial de adquisición. Las representaciones más comunes para representar un objeto 3D son: una nube de puntos de su superficie; una malla '*mesh*' de su superficie calculada a partir de la aproximación poligonal de la nube de puntos; y un volumen compuesto por '*voxels*' (cada voxel representa un pequeño volumen cúbico que envuelve un punto de la nube) (Cyganek y Siebert 2009). En este capítulo, los descriptores que se presentan se obtendrán siempre a partir de la representación de una nube de puntos, y éstos se pueden agrupar en tres grandes tipos:

a) Basados en características de puntos de la nube y su entorno de vecindad.
b) Basados en características de punto de vista.
c) Basados en orientación de superficie.

Estos descriptores hacen uso de conceptos básicos anteriormente comentados, tales como entorno de vecindad, vector normal, curvatura o marco de Darboux, para describir la geometría de la superficie de un objeto representada como una nube de puntos.

14.3.1. Histograma de características de tipo punto: PFH y FPFH

El histograma de características de tipo punto ('*PFH: Point Feature Histogram*') es un descriptor que hace uso de las características de puntos escogidos p_q de la superficie **P** a describir (Rusu, 2013). Los puntos escogidos representan su geometría en función de la relación a sus puntos vecinos p_i en cierto entorno de vecindad P^r. La información geométrica de p_q se obtiene midiendo la variación entre las características de p_q y los p_i en ese entorno de vecindad.

En primer lugar, para poder calcular PFH conviene definir el concepto de entorno de vecindad de doble radio. Así, para un punto cualquiera $p_q \in \mathbf{P}$ se pueden definir dos radios r_1 y r_2 con $r_1 < r_2$ que determinan dos entornos de vecindad concéntricos P^{r1} y P^{r2}. Cada uno de los dos entornos de vecindad proporcionan una representación distinta de p_q. De este modo, PFH permite codificar dos tipos de información por cada punto candidato. Por un lado, se calcula el vector normal a la superficie en p_q haciendo uso de PCA (ver sección 14.2.3) en el entorno P^{r1} ; para ello se emplea la ecuación (14.10). Por otro lado, PFH calcula el valor de curvatura media alrededor de p_q como una codificación de las propiedades geométricas en el entorno P^{r2} según la ecuación (14.11). Después, PFH codifica las posibles variaciones geométricas de superficie considerando los cambios de dirección de los vectores normales estimados en P^{r2}.

A continuación, se ilustra cómo se lleva a cabo el cálculo de la variación geométrica entre dos puntos cualesquiera p_i y p_j de P^{r2} que se encuentran separados una distancia Euclídea d_{L2}. Para ello, se parte de que previamente ya se han calculado todos los vectores normales para todos los puntos de P^{r2} y que se ha definido un sistema de referencia fijo (Marco de Darboux) en uno de los puntos, por ejemplo p_i. Este marco de referencia fijado en p_i queda definido por $\{\bar{u}, \bar{v}, \bar{w}\} = \{\bar{n}_i, \bar{t}_i, \bar{n}_i \wedge \bar{t}_i\}$ tal como se presentó en la Sección 14.2.4, de la siguiente manera:

$$\bar{u} = \bar{n}_i, \qquad \bar{v}=\bar{t}_i=\bar{u}\wedge\frac{(p_j - p_i)}{d_{L2}}, \qquad \bar{w} = \bar{u} \wedge \bar{v} = \bar{n}_i \wedge \bar{t}_i \tag{14.12}$$

De modo que la variación geométrica entre ambos puntos se puede expresar como la diferencia relativa entre la dirección de los vectores normales $\bar{n}_i$ y $\bar{n}_j$ de dichos puntos (ver Figura 14.5), y se calcula como:

$$\alpha = \text{acos}\,(\bar{v} \cdot n_j), \qquad \emptyset = \text{acos}\left(\bar{u}\cdot\frac{(p_j - p_i)}{d_{L2}}\right), \qquad \theta = \text{atan}\left(\bar{w} \cdot \bar{n}_j, \bar{u} \cdot \bar{n}_j\right) \tag{14.13}$$

En el algoritmo de cálculo del descriptor PFH se calcula la tupla $(\alpha, \phi, \theta, d_{L2})$ para cualquier par de puntos p_i y p_j de P^{r2}. Como consecuencia, si el entorno de vecindad P^{r2} contiene k número de puntos, entonces la representación PFH de ese entorno estará formada por k^2 tuplas $(\alpha, \phi, \theta, d_{L2})$, figura 14.6a. Por simplicidad, de ahora en adelante d_{L2} será llamado d.

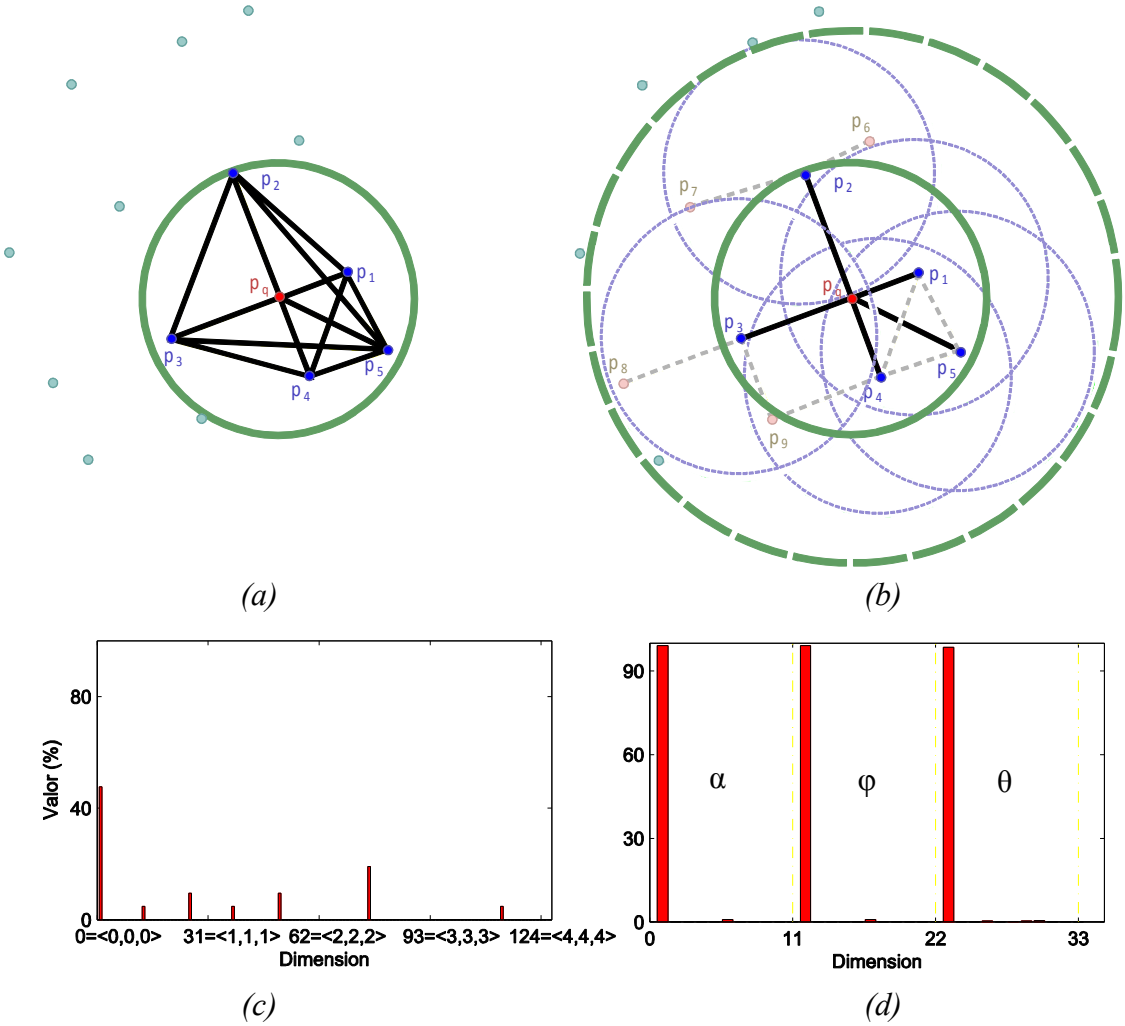


Figura 14.6. (*a*) Entorno de vecindad P^{r2} para un punto candidato p_q y relación con sus vecinos, (*b*) Influencia de vecinos en FPFH para p_q, (*c*) Firma del descriptor PFH de p_q como una conjunto de tuplas (α, ϕ, θ) con 125 codificaciones, (*d*) Firma del descriptor FPFH de p_q con 33 codificaciones.

El descriptor PFH de un punto candidato p_q se puede representar como un histograma multidimensional invariante a transformaciones espaciales (6 grados de libertad: traslación y rotación) de la superficie que describe. Por lo general, un histograma unidimensional representa el número de veces que se repite un valor. Por lo tanto, aplicando el mismo concepto, un histograma como PFH representa el número de veces que se repiten los valores de una tupla (α, ϕ, θ), figura 14.6c. Además, como cualquier histograma unidimensional, es necesario dividirlo en un número determinado de subdivisiones. Cada una de esas subdivisiones representa un rango de valores de cada elemento de la tupla y gráficamente, indican el número de ocurrencias en cada intervalo de valores, figura 14.6. Es conveniente que haya correlación entre valores de la tupla y, para conseguirlo, conviene crear el mismo número de subdivisiones para cada elemento de la tupla. Puesto que tres de los valores son angulares, el criterio más usado es emplear la máxima diferencia angular para determinar el máximo número de subdivisiones a crear. El descriptor PFH se comporta mejor que el descriptor FPFH frente a la presencia de ruido y con nubes de puntos cuyos entornos de vecindad varían sus densidades. Además, es posible reducir la complejidad de PFH despreciando la componente d, quedando la tupla tal que (α, ϕ, θ). Esto es frecuente cuando se trabaja con nubes de puntos que no representan distancia

en la escena real, si no que la información almacenada en distancia es una distancia relativa entre objetos en la nube y no en la escena. En este caso se dice que la nube de puntos contiene información $2^{1/2}$D y no 3D.

El principal problema de PFH es el coste computacional que supone trabajar con todos los puntos pertenecientes al entorno de vecindad. Una solución para reducir la complejidad de PFH manteniendo su poder de descripción, que nos va a permitir discriminar superficies distintas, es la variante FPFH ('*Fast PFH*'). El cálculo del descriptor FPFH (Rusu, 2013) se hace en dos pasos.

Inicialmente, se procede de modo similar a como se ha hecho en FPH. Así, para cada punto candidato p_q con P^{r2} que contiene k puntos p_i, se calcula el conjunto de todas las tuplas (α, ϕ, θ) de p_q a p_i según la ecuación (14.13). El histograma multidimensional resultante se llamará SPFH ('*Simply PFH*'). Posteriormente, para cada punto p_i vecino de p_q también se considera su propio entorno de vecindad P^{r2}, figura 14.6b, y se vuelve a calcular SPFH, pero esta vez, los valores se ponderan por un peso en función de la distancia de cada p_q a p_i, de acuerdo a:

$$FPFH(p_q) = SPFH(p_q) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{\omega_i}SPFH(p_i) \tag{14.14}$$

De este modo, habrá puntos vecinos p_i que contribuyen más a la geometría de p_q que otros de sus vecinos, como se observa en la figura 14.6b. A diferencia de PFH, FPFH no considera todas las relaciones entre puntos dentro de P^{r2}para determinar la geometría, sólo las relaciones entre p_q y p_i. Además, sí que considera puntos vecinos fuera de P^{r2}, aunque la influencia de estos será menor debido a la ponderación por proximidad a p_q.

En general, PFH trabaja con un histograma codificado de los valores angulares de la tupla (α, ϕ, θ), el número de posibles combinaciones es 125. Esto es debido a que cada uno de los valores angulares de la tupla se encuentra clasificado en 5 subdivisiones (una cada 72º). En la figura 14.6c, la 46ª subdivisión indica que $(\alpha, \phi, \theta) = (1,4,1) = 1\times1 + 4\times5 + 1\times25 = 46$, o lo que es lo mismo que los valores angulares son $\alpha = 72º, \phi = 288º, \theta = 72º$. FPFH trabaja con 11 subdivisiones (cada 32º) por elemento de la tupla. No obstante, para eliminar subdivisiones donde no hay ocurrencias (número de veces que se repite el valor de la tupla es 0), se emplean 3 histogramas independientes concatenados, uno para cada elemento de la tupla, de modo que se dan 33 posibles combinaciones en vez de 1331, figura 14.6d.

14.3.2. Histogramas de características de punto de vista: VFH y CVFH

El histograma VFH ('*VFH: View Feature Histogram*') (Rusu y col. 2010; Rusu, 2013) se crea como una extensión de FPFH, ya que éste es robusto al ruido y a la escala pero no es invariante al punto de vista de la escena. Si se recuerda, tanto PFH como FPFH codifican los ángulos de variación (cabeceo, alabeo y guiño) entre los vectores normales de dos puntos cualesquiera en una región de la superficie mediante un marco de Darboux (Figura 14.5), y éste se emplea para determinar la forma geométrica de la superficie. Esta misma idea se vuelve a emplear en VFH que, además, incorpora información para codificar la dirección del punto de vista. De este modo es posible emplear VFH tanto para procesos de

reconocimiento de objetos, como para procesos en los que es necesario calcular su posición *('pose'* en el ámbito anglosajón). Así, VFH consta de un descriptor de punto de vista y un descriptor de forma. El descriptor de forma se calcula como en FPFH, añadiendo la componente d, de modo que en esta sección sólo se comenta cómo se calcula la componente de punto de vista de VFH.

Para definir el punto de vista, se calcula el centroide de la superficie p_c para fijar un sistema de referencia en ese punto. Así, la dirección de referencia se escoge como un vector normal a la superficie $\bar{n}_c$ en el centroide p_c. De modo que el punto de vista se codifica como un histograma que representa la desviación angular entre $\bar{n}_c$ y el vector normal $\bar{n}_j$ en cada punto p_j, de modo similar a como se hizo en FPH y FFPH. Es decir, se asume que $p_i = p_c$ en la ecuación (14.12) para definir el marco de Darboux. Y posteriormente, se mide la variación angular de cabeceo, alabeo y guiño entre el punto de referencia del objeto p_c y cada uno de los puntos p_j de la superficie, como la desviación angular entre los vectores normales $\bar{n}_j$ a la superficie y la dirección de referencia de punto de vista definida por $\bar{n}_c$, de acuerdo a la ecuación (14.13).

El descriptor CVFH ('*CVFH: Clustered Viewpoint Feature Histogram*') (Aldoma y col. 2012) se crea como una extensión de VFH para solventar algunos de los problemas de éste último. Entre los inconvenientes que presenta VFH cabe mencionar que no permite determinar la localización de un objeto cuando hay giros de la escena sobre el eje de punto de vista de la cámara (eje z). Este hecho genera ambigüedad si se desea estimar la 'pose' del objeto. Además, VFH presenta deficiencias en el reconocimiento de objetos reales porque es sensible a la pérdida de información. Es decir, la estimación de centroides y normales se ve afectada por tres factores: la presencia de oclusiones en la superficie del objeto (el objeto se observa parcialmente y no se registran todos los puntos de la superficie); los procesos de segmentación para identificar sólo los puntos de objetos (eliminar otros puntos de la escena); y los puntos que yacen en aristas del objeto que suelen ser registrados con mucho ruido por el sensor RGBD o ToF. CVFH tiene una gran ventaja frente a VFH, y es que no considera aquellas partes de la nube de puntos que se sabe que no se han registrado de manera robusta por el sensor, como son las aristas del objeto.

CVFH divide la nube de puntos **P** que representa el objeto en regiones estables '*clusters*' $\mathbf{P}=\{\mathrm{P}_1, \mathrm{P}_2, \ldots, \mathrm{P}_n\}$. Cada subconjunto ('*cluster*') P_k representa una porción del objeto cuyos puntos han sido registrados con poco ruido. Para ello, se eliminan todos aquellos puntos que tienen un valor de curvatura elevado. Los puntos con curvatura grande suelen corresponder con puntos atípicos producidos por ruido o con puntos en las proximidades de aristas del objeto. Para dividir **P** en regiones P_k estables, se busca agrupar puntos p_i en función de la dirección de sus vectores normales. Es decir, se busca construir agrupaciones de puntos p_{ki} en regiones P_k que haga máxima la variación entre los valores medios de la dirección del vector normal de todos los *clusters* $\{\mathrm{P}_1, \mathrm{P}_2, \ldots, \mathrm{P}_n\}$ y a su vez hagan mínima la variación en dirección de todos los puntos p_{ki} de un mismo conjunto P_k. Así, a modo de ejemplo, un p_i se incluirá como un punto de P_k y pasaría a etiquetarse como p_{ki}, si dado un punto $p_{kj} \in \mathrm{P}_k$se cumpliera que ambos son próximos, es decir por debajo de un umbral $\|p_i - p_{kj}\|<d$, además de que las orientaciones de sus vectores normales son parecidas, es decir $\bar{n}_i \cdot \bar{n}_{kj}>n$. Una vez se tienen las regiones estables P_k se procede a calcular los descriptores VFH para cada una de ellas. Primero, se estima el centroide y la normal de éste para definir el punto de vista de cada región. Y

después, se calculan las desviaciones angulares para el conjunto de cada punto p_{ki} con respecto al centroide.

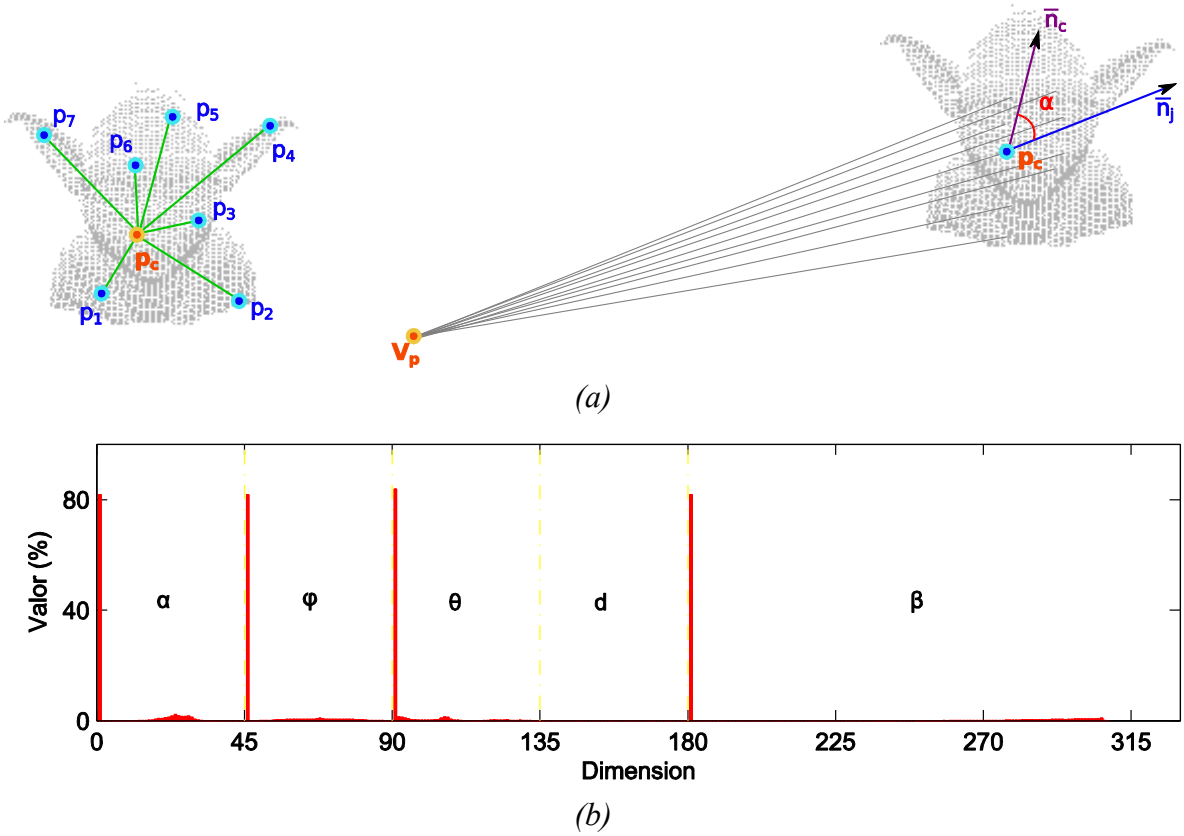


Figura 14.7. *a*) Descriptor de punto de vista VFH, (*b*) Firma del descriptor VFH/CVFH con las dos componentes encadenadas: forma y punto de vista β.

La representación del descriptor VFH trabaja con 45 subdivisiones para cada uno de los elementos de la tupla $(\alpha, \phi, \theta, d)$, en vez de 11 como FPFH y 128 codificaciones para el punto de vista β. En total 308 codificaciones para 5 variables distintas $(\alpha, \phi, \theta, d, \beta)$. La distribución de forma $(\alpha, \phi, \theta, d)$ permite diferenciar superficies similares en tamaño y densidad pero cuyos puntos guardan otra distribución.

14.3.3. Histogramas de orientación de superficies: SHOT

El descriptor 3D, SHOT ('*Signature of Histograms of OrienTation*') (Salti y col. 2014) codifica dos tipos de información. Por un lado, codifica en forma de histograma la información geométrica de los puntos de la superficie. Estos histogramas almacenan el número de veces que se repiten cada uno de los valores geométricos de la superficie, tal y como se ha hecho en los descriptores anteriores, aunque en este caso la tupla de información geométrica es distinta y no es calculada usando el marco de referencia de Darboux. Por otro lado, y de manera conjunta, se codifica una firma que registra para cada valor geométrico, cuáles son las coordenadas 3D de los puntos de la nube que contribuyen con la misma variación geométrica. De este modo, y a diferencia del resto de descriptores, SHOT registra no sólo información de variación geométrica sino que también aporta información espacial de donde se presenta ésta en la nube de puntos.

El nuevo marco de referencia se define a partir de una rejilla esférica isotrópica que se divide en sectores mediante paralelos (divisiones en ángulo de azimut), meridianos (divisiones en ángulo de elevación) y radios. Lo más habitual, es usar una rejilla esférica que consta de 32 sectores formados a partir de 8 divisiones en acimut, 2 divisiones en elevación y 2 a lo largo del radio, figura 14.8a.

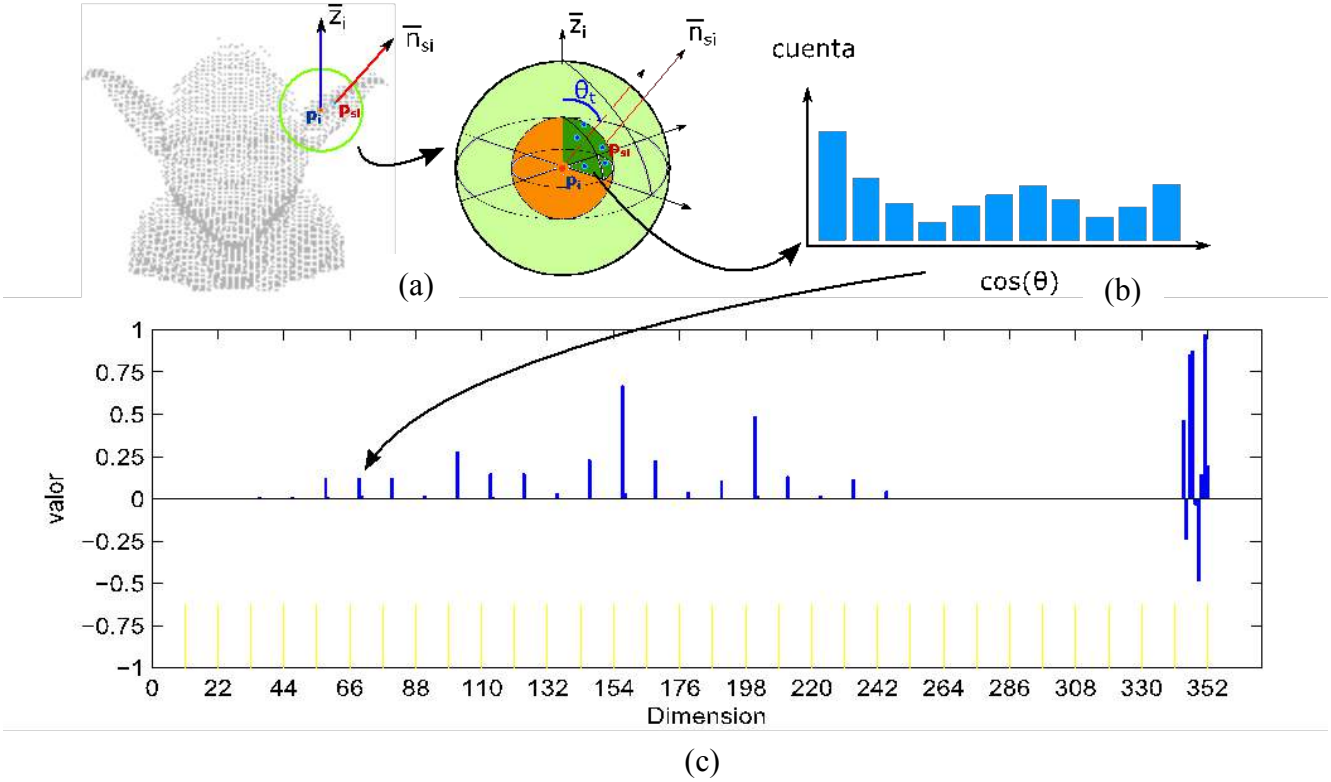


Figura 14.8. (*a*) Marco de referencia de SHOT en p_i (por simplicidad solo se han representado 4 secciones sobre acimut) (*b*) Histograma local de SHOT para una sección (*c*) Descriptor de SHOT en p_i para todas las regiones

Para un punto p_i cualquiera de **P** sobre el que se desea calcular la geometría, se construye una esfera cuyo centroide se hace coincidir con p_i. Después, la esfera se orienta de modo que su 'polo norte' coincida con la dirección del vector normal a $\bar{n}_i$ en p_i. De esta manera se ha fijado un nuevo marco de referencia donde el eje z de la esfera coincide con $\bar{n}_i$ denotándose ahora como $\bar{z}_i$. Posteriormente, y haciendo uso de ese marco de referencia, se construye un histograma local, figura 14.8b, que define la relación de variación geométrica entre p_i y cada uno de los sectores en los que ha quedado dividida la rejilla esférica isotrópica, figura 14.8a. En total 32 histogramas locales muestran la variación geométrica entre el punto p_i y la rejilla esférica. Cada uno de los 32 histogramas locales determina el número de veces que se repite un cierto valor angular θ_i. El ángulo θ_i se calcula para cada punto p_{si} dentro de la sección de la esfera y representa la variación angular entre el vector normal al sector $\bar{n}_{si}$ en ese punto, dentro del sector del marco de referencia y el eje z en el punto p_i de **P**. Así, el ángulo θ_i que permite construir el histograma local para cada sector de la rejilla se obtiene como:

$$\theta_i = \text{acos}(z_i, \bar{n}_{si}) \tag{14.15}$$

Y el histograma local representa el número de veces que se repite cada posible valor de θ_i. La concatenación de los 32 histogramas locales de p_i, representa su descriptor SHOT, figura 14.8c.

14.4. Aplicación al reconocimiento y clasificación de objetos 3D

En esta sección se presenta un esquema general para un método básico de reconocimiento y clasificación de objetos rígidos tridimensionales, como en Aldoma y col. (2014). El método propuesto asume que la escena ha sido capturada como una nube de puntos (ver sección 14.1). La escena además puede ser compleja o simple. Si es compleja por lo general representa varios objetos adquiridos desde un sensor RGBD o ToF, y si es simple, la escena representa un único objeto que se apoya sobre una mesa de trabajo.

El método de reconocimiento propuesto consta de cinco etapas de procesamiento básicas, figura 14.9. En la primera etapa '*Adquisición Datos*' el sensor adquiere información de una escena que es almacenada como una nube de puntos. Tanto si la escena es simple como si es compleja, la nube de puntos que la representa debe ser filtrada. Por ejemplo, en una escena simple, se requiere aislar los puntos que representan el objeto del resto de puntos (mesa de trabajo, fondo, etc.). Si la escena es compleja, además, hay que dividir la nube de puntos en regiones, una para cada objeto. En este ejemplo, se asume que la escena es simple. Para detectar el objeto presente en la escena, primero se realiza una etapa de '*Pre-procesado*', esta etapa consiste en aplicar filtros como los vistos en las secciones 14.2.1 y 14.2.2, de este modo se aíslan los puntos que determinan el objeto del resto de puntos de la escena. Así, se elimina el ruido y todos aquellos puntos demasiado cercanos o alejados del sensor. En un tercer paso, '*Extracción de objetos*' se emplea el algoritmo de Trevor (Trevor y col. 2013) para encontrar el plano que ajusta todos aquellos puntos de la nube que forman parte de la mesa de trabajo (plano dominante). En el caso de que la nube represente objetos aislados sin fondo (por ejemplo, una representación CAD), algunas de estas etapas pueden no ser necesarias.

Figura 14.9. Etapas generales del método de reconocimiento.

Una vez que se dispone de una nube de puntos que representan un único objeto rígido aislado $\mathbf{O_i}$, se calculan los vectores normales, como se ha comentado en la sección 14.2.3., y partir de éstos y empleando el marco de Darboux mostrado en la sección 14.2.4, se puede '*Calcular un descriptor*' escogiendo cualquiera de los comentados en la sección 14.3. Además, si el objeto $\mathbf{O_i}$ quiere ser identificado y clasificado se requiere disponer de un modelo $\mathbf{M_j}$ con el que se desea comparar. Para que objeto y modelo puedan ser comparados, se requiere que la representación de ambos sea similar, de ahí que también $\mathbf{M_j}$ estará representado por una nube de puntos. Más aún, si la clasificación se quiere realizar correctamente, se necesita disponer de un conjunto amplio de representaciones de $\mathbf{M_j}$, esto es, obtener diversas nubes de puntos que representen vistas de dicho modelo. Por ejemplo, si $\mathbf{M_j}$

es un modelo CAD de una forma geométrica se pueden aplicar distintas rotaciones, traslaciones y escalados para construir un conjunto de nubes de puntos que simulen los *n* distintos puntos de vista $\mathbf{M_j} = \{m_{j1}, \ldots, m_{jn}\}$. Y para cada una de estas vistas se calcula un descriptor, de entre los vistos.

Generalmente, en cualquier proceso de reconocimiento se almacenan en una base de datos un conjunto de descriptores para cada uno de las distintas clases de objetos que se desea reconocer. Cada clase de objeto viene representada por un modelo $\mathbf{M_j}$. La figura 14.10 representa los objetos empleados en este ejemplo de aplicación.

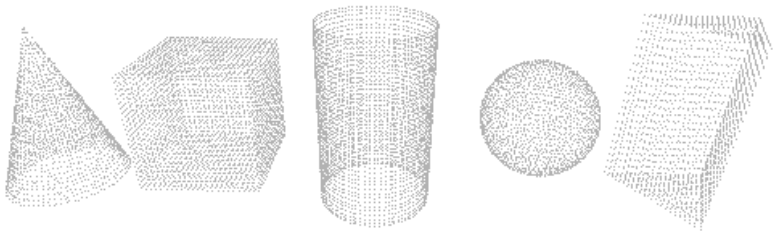


Figura 14.10. Modelos almacenados en la base de datos.

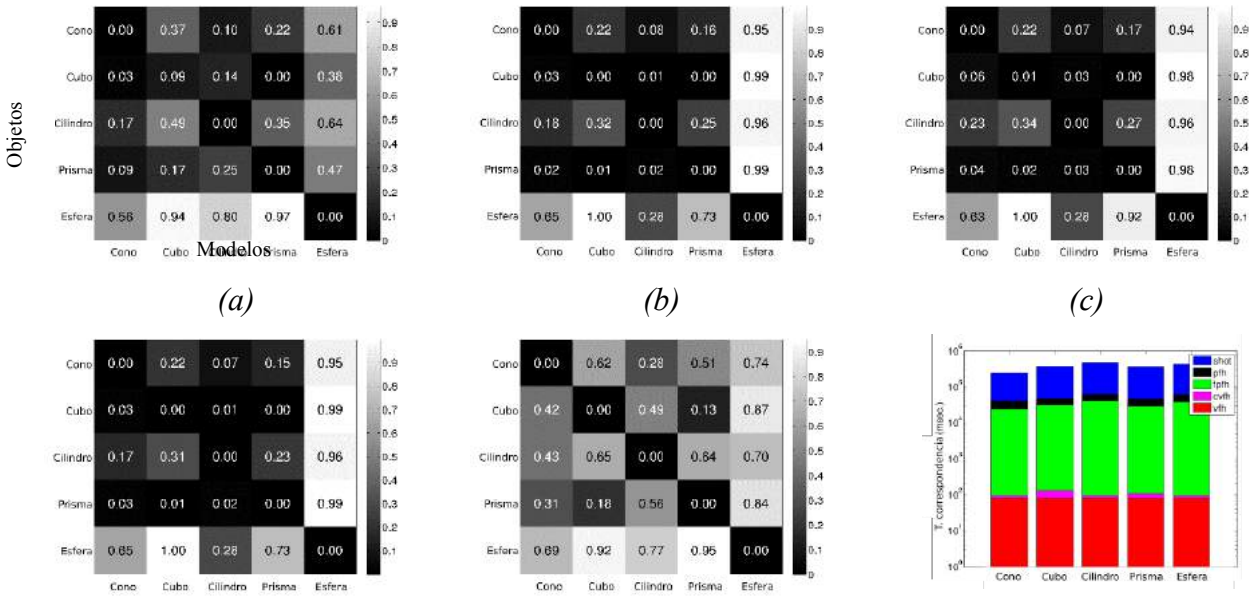


Figura 14.11. Distancias métricas calculadas en el proceso de correspondencia entre objeto y modelo para cada uno de los descriptores (*a*) PFH (*b*) FPFH (*c*) VFH (*d*) CVFH (*e*) SHOT (*f*) Comparativa de tiempos.

Finalmente, la clasificación se lleva a cabo mediante una etapa de '*Correspondencia*' donde se compara el descriptor de cada una de las vistas $\{m_{j1}, \ldots, m_{jn}\}$ del modelo $\mathbf{M_j}$ con la nube de puntos del objeto. Si desde el sensor se ha adquirido más de una vista del objeto, por ejemplo *m* vistas, se tendrá que comparar cada $\mathbf{O_i} = \{o_{i1}, \ldots, o_{im}\}$ con las vistas de los modelos $\mathbf{M_j}$., similar a lo que sucede en Mateo y col. (2015). Para compararlos, se emplean las distancias métricas vistas en 14.1.2 de acuerdo a la ecuación (14.15). Así, aplicando la ecuación (14.15) para cada uno de los descriptores, se obtienen las matrices de distancia, figura 14.11.

$$d(\mathbf{O_i}, \mathbf{M_j}) = \min\{d(o_k, m_r)\} \text{ donde } 1 < r < n \text{ y } 1 < k < m \quad (14.16)$$

14.5. Bibliografía

Aldoma, A.; Marton, Z.C.; Tombari, F.; Wohlkinger, W.; Potthast, C.; Zeisl, B.; Rusu, R.B.; Gedikli, S.;Vincze, M. (2014) Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation. *IEEE Robotics & Automation Magazine*, vol. 1070(9932/12), pp. 80-91.

Aldoma, A.; Tombari, F.; Rusu, R.; Vincze, M. (2012) OUR-CVFH – Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram for Object Recognition and 6DOF Pose Estimation. En *Pattern Recognition*, *Lecture Notes in Computer Science*; Springer-Verlag: Germany; Vol. 7476, pp.113-122.

Cyganek, B.; Siebert, J.P.; (2009) An introduction to 3D Computer vision techniques and algorithms. *Editorial John Wiley & Sons.* ISBN: 978-0-470-01704-3.

Eberhard, H.; Klumpp, V.; Hanebeck, U.B.; (2010) Density trees for efficient nonlinear state estimation. *13th Conference on Information Fusion (FUSION),* pp. 1-8, ISBN: 978-0-9824438-1-1

Gil, P.; Mateo, C.M.; Torres, F. (2014). 3D Visual Sensing of Human Hand for Remote Operation of a Robotic Hand. *International Journal of Advanced Robotics System*, vol. 11(26), pp. 1-13.

Klette, R.; Schlüns, K.; Koschan, A.; (1998) Computer Vision: Three-dimensional data from images. *Springer*. ISBN: 981-3083-71-9.

Lancaster, P.; Salkanskas K.; (1981) Surface Generated by Moving Least Squares Methods. *Mathematics of computation*, Vol. 37, No. 155, pp. 141-158.

Mateo, C.M.; Gil, P., Torres, F. (2015) Analysis of Shapes to Measure Surface: an approach for detection of deformations. En *Actas de 12th Conference on Informatics in Control Automation and Robotics (ICINCO)*, Colmar, Francia, 21-23 Julio 2015, vol. 2, pp. 60-65.

Mateo, C.M.; Gil, P.; Torres, F. (2015) Visual perception for the 3D recognition of geometric pieces in robotic manipulation. *The International Journal of Advanced Manufacturing*, doi: 10.1007/s00170-015-7708-8, pp. 1-15.

Pauly, M.; Gross, M., Kobbelt, L.P.; (2002) Efficient Simplification of Point-Sampled Surfaces. En *Actas de IEEE Conference on Visualization* pp. 163-170

Rusu, R. (2013) *Semantic 3D object maps for everyday robot manipulation*; STAR 85, Springer Series-Verlag, Germany, pp. 154–196.

Rusu, R.; Bradski, G., Thibaux, R.; Hsu, J. (2010) Fast 3D recognition and pose using the Viewpoint Feature Histogram. En *Actas de IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* Taipei, Taiwan, 18-22 October 2010; pp. 2155-2161.

Salti, S.; Tombari, F.; Di-Stefano, L. (2014) SHOT: Unique signatures of Surface and texture description. *Computer Vision and Image Understanding*, vol. 125, pp. 251-264.

Trevor A, Rusu RB, Christensen HI (2013) Efficient organized point cloud segmentation with connected components. En *Actas de* 3rd Workshop on Semantic Perception

Trucco, E.; Verri, A.; (1998) Introductory techniques for 3D computer vision. *Prentice Hall*. ISBN: 0-13-261108-2.

CAPÍTULO 15

AJUSTE Y DETECCIÓN DE MODELOS GEOMÉTRICOS EN IMÁGENES. ALGORITMO RANSAC

Félix MIGUEL-TRESPADERNE[1], Eusebio de la FUENTE[2], J. GÓMEZ-GARCÍA-BERMEJO[2]

[1] Dpto. Ingeniería Sistemas y Automática, Escuela Ingenierías Industriales, Univ.Valladolid, España
[2] Instituto Tecnologías Avanzadas de la Producción (ITAP), Universidad de Valladolid, España

Las aplicaciones industriales de visión artificial requieren de la detección de determinados patrones geométricos en la imagen. En estas aplicaciones es muy común la presencia de primitivas geométricas tales como rectas, circunferencias o elipses. Sin embargo, la detección de estos patrones en la imagen no es una tarea trivial.

En este capítulo, en primer lugar, recordaremos las técnicas tradicionales de mínimos cuadrados que nos permitirán ajustar un modelo geométrico a un determinado conjunto de puntos para pasar luego a analizar la técnica RANSAC, que permite la detección robusta de rectas, circunferencias y elipses en imágenes con mucho ruido o con patrones incompletos.

15.1. Ajuste por mínimos cuadrados

El ajuste por mínimos cuadrados es una técnica de optimización muy conocida en análisis numérico que permite obtener la solución de un sistema de ecuaciones sobredeterminado, es decir, que presenta un número de ecuaciones superior al de incógnitas.

Sea $\mathbf{p}$ el vector de m parámetros a estimar y $\mathbf{z}$ un vector formado a partir de los datos medidos. Sea $\varepsilon_i(\mathbf{p},\mathbf{z}),\ i = 1,\dots n$ un conjunto de n ecuaciones que modelan los errores cometidos al utilizar el vector de parámetros $\mathbf{p}$ como solución.

El método de mínimos cuadrados encuentra aquella solución $\mathbf{p}$ tal que al sustituir las incógnitas halladas en cada una de las ecuaciones $\varepsilon_i(\mathbf{p},\mathbf{z})$ la suma de todos los errores al cuadrado es minimizada:

$$\min_{\mathbf{p}} \sum_{i=1}^{n} \varepsilon_i^2(\mathbf{p}, \mathbf{z}) \tag{15.1}$$

El modelo del error $\varepsilon(\mathbf{p}, \mathbf{z})$ caracteriza el tipo de solución obtenida, pudiendo dividirse los algoritmos entre métodos geométricos y algebraicos:

- **Métodos geométricos**: Típicamente toman como modelo de error la distancia geométrica entre cada punto y la curva cuyos parámetros queremos estimar. En el caso de la recta, existe una elegante solución cerrada al problema. Lamentablemente, para circunferencias y elipses, el modelo de error basado en la distancia geométrica es una ecuación no lineal y no es posible obtener soluciones cerradas, debiéndose recurrir a métodos iterativos.
- **Métodos algebraicos**: Parten de escribir la ecuación de la curva o alguna otra propiedad relevante en forma implícita. En presencia de errores, la ecuación implícita deja de ser 0, y ese es precisamente el error algebraico. En general, existe una solución cerrada al problema. Sin embargo, muchos de estos métodos tienen inconvenientes, tales como ausencia de significado geométrico del error, no son invariantes por ejemplo a traslaciones o rotaciones de los datos, sus estimaciones son sesgadas, etc.

La elección del método está fuertemente condicionada a la aplicación. Si tenemos un tiempo de ciclo muy exigente, seguramente no pueda recurrirse a un método geométrico y haya que considerar un método algebraico. En contraposición, si los puntos de contorno con los que se estimarán los parámetros representan solo una pequeña porción del modelo, una mala selección del método algebraico puede proporcionar resultados catastróficos.

Afortunadamente, hoy en día casi todos los paquetes de programación disponen de excelentes rutinas que permiten despreocuparse de problemas de inestabilidades numéricas escogiendo adecuadamente los puntos de partida.

Presentaremos algunos procedimientos de mínimos cuadrados para ajustar rectas y circunferencias a un conjunto de puntos de contorno de la imagen. No lo haremos para elipses por razones de espacio. El lector interesado puede encontrar en las referencias Taubin (1991), Gander (1994), Fitzgibbon y col. (1995), Halif y Flusser (1998) y Kanatani y Rangarajan (2010) abundante información.

15.1.1. Solución algebraica general

La ecuación implícita de rectas y cónicas puede escribirse de forma general como:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0 \tag{15.2}$$

Por ejemplo, si exigimos $A = B = C = 0$ tenemos la ecuación de una recta o la de una elipse si se cumple $B^2 - 4AC < 0$.

Los errores ε_i, que podemos denominar también distancias algebraicas d_{A_i}, en cada uno de los n puntos disponibles permiten establecer el siguiente sistema de n ecuaciones lineales:

$$\begin{aligned}\varepsilon_1 = d_{A_1} &= Ax_1^2 + Bx_1y_1 + Cy_1^2 + Dx_1 + Ey_1 + F \\ \varepsilon_2 = d_{A_2} &= Ax_2^2 + Bx_2y_2 + Cy_2^2 + Dx_2 + Ey_2 + F \\ &\vdots \\ \varepsilon_n = d_{A_n} &= Ax_n^2 + Bx_ny_n + Cy_n^2 + Dx_n + Ey_n + F\end{aligned} \tag{15.3}$$

La función S a minimizar es la suma de los cuadrados de los errores:

$$S(A,B,C,D,E,F) = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} {d_{A_i}}^2 \tag{15.4}$$

En cualquier caso, la propiedad relevante es que la ecuación (15.2) es lineal en el vector de parámetros y podemos reescribirla en forma compacta como:

$$\mathbf{z^t p} = 0, \text{con} \quad \begin{cases} \mathbf{z} = (x^2, xy, y^2, x, y, 1)^t \\ \mathbf{p} = (A,B,C,D,E,F)^t \end{cases} \tag{15.5}$$

y, por tanto, el sistema de ecuaciones (15.3) queda de la siguiente forma:

$$\mathbf{Zp} = 0, \text{con} \quad \{ \quad \mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n)^t \tag{15.6}$$

La función $S(A,B,C,D,E,F)$ a minimizar descrita en (15.4) queda matricialmente como:

$$S(\mathbf{p}) = \sum_{i=1}^{n} \varepsilon_i^2 = (\mathbf{Zp})^t(\mathbf{Zp}) = \mathbf{p^t Z^t Zp} = \mathbf{p^t Mp} \tag{15.7}$$

Para evitar la solución trivial $\mathbf{p} = \mathbf{0}$, puede imponerse una restricción cuadrática sobre el vector de parámetros de tal forma que $\mathbf{p^t Np} = 1$, donde $\mathbf{N}$ es una matriz simétrica (Bookstein, 1979).

El problema consiste ahora en minimizar un problema con multiplicadores de Lagrange $L(\mathbf{p}, \lambda)$:

$$L(\mathbf{p}, \lambda) = S(\mathbf{p}) - \lambda \cdot (\mathbf{p^t Np} - 1) = \mathbf{p^t Mp} - \lambda \cdot (\mathbf{p^t Np} - 1) \tag{15.8}$$

Derivando respecto a $\mathbf{p}$ y λ obtenemos la pareja de ecuaciones matriciales:

$$\mathbf{Mp} = \lambda \mathbf{Np} \quad \text{y} \quad \mathbf{p^t Np} = 1 \tag{15.9}$$

por lo que la solución $\mathbf{p}$ al problema es uno de los autovectores generalizados de la pareja de matrices $(\mathbf{M}, \mathbf{N})$. Para ver qué solución escoger, basta observar que:

$$S(\mathbf{p}) = \mathbf{p^t Mp} = \mathbf{p^t} \lambda \mathbf{Np} = \lambda \mathbf{p^t Np} = \lambda \tag{15.10}$$

y como buscamos minimizar $S(\mathbf{p})$, la solución será la asociada al *menor autovalor no negativo*.

Existen diferentes formas de resolver este problema, cada una con sus ventajas y desventajas respecto a velocidad y estabilidad numérica. Para el caso de circunferencias y elipses, si la aplicación necesita estimar sus parámetros con un conjunto de datos de partida poco representativo de la curva, es recomendable utilizar un método de resolución que sea especialmente cuidadoso con las particularidades numéricas de las operaciones algebraicas involucradas. Entre las muchas páginas en

Internet con este tipo de programas de acceso libre, puede obtenerse el código MATLAB de bastantes de las técnicas de estimación que comentaremos en este capítulo en (Chernov 2012).

15.1.2. La regresión ortogonal de la recta

A partir de la ecuación implícita general (15.2), eliminando los términos cuadráticos y renombrando los parámetros a la notación universalmente utilizada, la ecuación de una recta viene dada por:

$$Ax + By + C = 0 \tag{15.11}$$

En (15.11) debe además cumplirse $A^2 + B^2 > 0$ para que $A = B = 0$ no conduzca a resultados absurdos. Además, para evitar que el vector de parámetros tome el valor $\mathbf{p} = (A, B, C)^t = \mathbf{0}$ debe emplearse alguna restricción.

La restricción $\mathbf{p^t N p} = \mathbf{p^t I p} = A^2 + B^2 + C^2 = 1$ es una opción factible, pero lo lógico es escoger:

$$\mathbf{p^t N p} = A^2 + B^2 = 1 \tag{15.12}$$

que no sólo garantiza $A^2 + B^2 > 0$, sino que de forma natural hace que la solución algebraica sea también geométrica. Veámoslo a continuación.

El error geométrico dado por la distancia de un punto a una recta es:

$$\varepsilon = d = \frac{|Ax + By + C|}{\sqrt{A^2 + B^2}} \tag{15.13}$$

por lo que la función $\mathrm{S}(A, B, C)$ a minimizar, suma de los cuadrados d_i^2 es:

$$\mathrm{S}(A, B, C) = \frac{1}{A^2 + B^2} \sum_{i=1}^{n} (Ax_i + By_i + C)^2 = \sum_{i=1}^{n} (Ax_i + By_i + C)^2 \tag{15.14}$$

donde utilizamos la restricción $A^2 + B^2 = 1$. Ahora, por tanto, el problema de minimización de distancias geométricas (15.13) se ajusta a la solución general planteada en (15.9).

Para minimizar debemos derivar respecto a los parámetros, siendo fácil eliminar el parámetro C:

$$\frac{\partial S(A, B, C)}{\partial C} = 2 \sum_{i=1}^{n} (Ax_i + By_i + C) = 0 \tag{15.15}$$

y despejando C en (15.15):

$$C = -A \frac{1}{n} \sum_{i=1}^{n} x_i - B \frac{1}{n} \sum_{i=1}^{n} y_i = -Ax_g - By_g \tag{15.16}$$

donde (x_g, y_g) es la media o centro de gravedad del conjunto de puntos. Esto no es una sorpresa, dado que la recta buscada es el eje de inercia de momento mínimo del conjunto de puntos (x_i, y_i).

El problema puede entonces reformularse trasladando el origen a (x_g, y_g), utilizando las nuevas coordenadas $x' = (x - x_g)$ e $y' = (y - y_g)$.

En las nuevas coordenadas la ecuación implícita (15.11) sujeta a la restricción (15.12) queda:

$$Ax' + By' = 0 \tag{15.17}$$

Identificando términos de la solución general (15.9) tenemos:

$$\begin{aligned}
&\mathbf{p} = (A, B)^t \\
&\mathbf{z} = (x', y')^t \\
&\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n)^t = \begin{bmatrix} x'_1 & y'_1 \\ \dots & \dots \\ x'_n & y'_n \end{bmatrix} \\
&\mathbf{M} = \mathbf{Z}^t\mathbf{Z} = \begin{bmatrix} \sum_{i=1}^{n} x_i'^2 & \sum_{i=1}^{n} x'_i y'_i \\ \sum_{i=1}^{n} x'_i y'_i & \sum_{i=1}^{n} y_i'^2 \end{bmatrix} = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix} \qquad \mathbf{N} = \mathbf{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
\end{aligned} \tag{15.18}$$

donde $\mathbf{M}$ es el tensor de los momentos de inercia. El problema planteado por (15.9), $\mathbf{Mp} = \lambda\mathbf{Np}$, teniendo en cuenta que $\mathbf{N}$ es la matriz unitaria queda:

$$\begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \lambda \begin{bmatrix} A \\ B \end{bmatrix} \tag{15.19}$$

Por tanto, la solución es el autovector del tensor de los momentos inercia $\mathbf{M}$ asociado al menor autovalor. Para ver algunas situaciones *patológicas* excepcionales, véase (Chernov, 2010).

Una solución más rápida a la inducida por (15.19) y numéricamente más estable al problema la encontramos haciendo uso de la descomposición en valores singulares (SVD) de la matriz $\mathbf{Z}$.

Para cualquier matriz $\mathbf{Z}$, cuya descomposición en valores singulares (SVD) es $\mathbf{Z} = \mathbf{U^t\Sigma V}$ se cumple:

$$\mathbf{M} = \mathbf{Z^tZ} = (\mathbf{V\Sigma^tU^t})(\mathbf{U\Sigma V^t}) = \mathbf{V}(\mathbf{\Sigma^t\Sigma})\mathbf{V^t} = \mathbf{VDV^t} \tag{15.20}$$

Multiplicando por $\mathbf{V}$ por la derecha en (15.20) obtenemos:

$$\mathbf{MV} = \mathbf{VD} \tag{15.21}$$

En (15.21) los autovectores de la matriz $\mathbf{M}$ son los vectores singulares por la derecha de la matriz $\mathbf{Z}$, es decir, los vectores columna de $\mathbf{V}$. Habitualmente, los paquetes de software ordenan los valores singulares en la diagonal $\mathbf{D}$ en orden decreciente; en ese supuesto, la solución buscada será el segundo vector columna de $\mathbf{V}$.

Nótese que no es necesario calcular la matriz $\mathbf{M}$ (ahorro de tiempo) y que el cálculo de la descomposición SVD es más estable numéricamente que el cálculo de los autovectores.

15.1.3. Ajuste a una circunferencia

Al igual que hemos visto para el caso de la recta, la opción a priori lógica para el problema de ajuste en el caso de una circunferencia es resolver el problema de regresión ortogonal basado en minimizar distancias geométricas al cuadrado. Los parámetros a estimar son el centro de la circunferencia (c_x, c_y) y su radio R. El modelo del error geométrico ε_G vendrá dado por:

$$\varepsilon_G = d_G = \sqrt{(x-c_x)^2+(y-c_y)^2} - R \tag{15.22}$$

La función $S_G(R, c_x, c_y)$ a minimizar es la suma de los cuadrados de las distancias geométricas:

$$S_G(R, c_x, c_y) = \sum_{i=1}^{n} {\varepsilon_G}_i^2 = \sum_{i=1}^{n} \left(\sqrt{(x_i-c_x)^2+(y_i-c_y)^2} - R \right)^2 \tag{15.23}$$

Lamentablemente, no existe una solución cerrada a la minimización de la función no lineal $S_G(R, c_x, c_y)$ en (15.23), debiendo recurrirse a soluciones iterativas basadas en métodos tales como el de Gauss-Newton o el de Levenberg-Marquardt (Madsen y col., 2004).

Este tipo de soluciones requieren de una solución inicial, típicamente alguna obtenida mediante un método algebraico. Por ello, vamos a enfocar el problema buscando una solución de tipo lineal usando métodos algebraicos. Así, revisaremos dos de los métodos algebraicos más conocidos: el método de Kasa y el método de Pratt.

La ecuación implícita general $Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$ presentada en (15.2) puede adaptarse a la descripción de una circunferencia siguiendo la parametrización de Pratt (1987):

$$A(x^2+y^2) + Bx + Cy + D = 0 \tag{15.24}$$

donde debe cumplirse la restricción $A \neq 0$ (si no tendríamos la ecuación de una recta) y también:

$$B^2 + C^2 - 4AD > 0 \tag{15.25}$$

La restricción (15.25) se pone claramente de manifiesto si reescribimos (15.24) en la forma clásica $(x-c_x)^2 + (y-c_y)^2 = R^2$:

$$\left(x + \frac{B}{2A}\right)^2 + \left(y + \frac{C}{2A}\right)^2 = \frac{B^2+C^2-4AD}{4A^2}$$
$$c_x = -\frac{B}{2A} \qquad c_y = -\frac{C}{2A} \qquad R^2 = \frac{B^2+C^2-4AD}{4A^2} \tag{15.26}$$

Método algebraico de Kasa (Kasa, 1976)

Es quizás uno de los más conocidos y utilizados. El modelo de error viene dado por:

$$\varepsilon_K = d_A = r^2 - R^2 = (x-c_x)^2 + (y-c_y)^2 - R^2 \tag{15.27}$$

donde $d_A = r^2 - R^2$ es la llamada *distancia algebraica*, y donde $r = R + d_G$ es obviamente la distancia del punto al centro de la circunferencia. El modelo de error ε_K es:

$$\varepsilon_K = d_A = A(x^2+y^2) + Bx + Cy + D \tag{15.28}$$

con la restricción $A^2 = 1$ para no degenerar en una recta. Por tanto, tenemos un problema de minimización cuya solución está dada por (15.9) con:

$$\begin{aligned}
&\mathbf{p} = (A, B, C, D)^t \\
&\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_\mathrm{n})^t = \begin{bmatrix} x_1^2 + y_1^2 & x_1 & y_1 & 1 \\ \ldots & \ldots & \ldots & \ldots \\ x_n^2 + y_n^2 & x_n & y_n & 1 \end{bmatrix} \\
&\mathbf{M} = \mathbf{Z}^t\mathbf{Z} \qquad \mathbf{N} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
\end{aligned} \tag{15.29}$$

El método de Kasa no da buenos resultados cuando los datos disponibles están concentrados en un arco pequeño de la circunferencia, dado que tiende a subestimar el radio y, por tanto, *preferir* circunferencias pequeñas. Esto puede verse mediante el siguiente análisis del modelo de error (15.27):

$$\varepsilon_K = r^2 - R^2 = (r + R)(r - R) = (d_G + R + R) \cdot d_G \approx 2R \cdot d_G \tag{15.30}$$

donde la aproximación se basa en suponer $|d_G| \ll 2R$. En (15.30) vemos claramente la influencia directa del valor R en el modelo del error. El resultado (15.30) sugiere utilizar como modelo del error $\varepsilon = \varepsilon_K/(2R) \approx d_G$ para eliminar la influencia del término $2R$, idea subyacente del método de Pratt.

Método algebraico de Pratt, (Pratt, 1987)

Tiene como modelo de error:

$$\varepsilon_P = \frac{r^2 - R^2}{2R} \tag{15.31}$$

quedando la función a minimizar $S_P(A, B, C, D) = \sum_{i=1}^{n} {\varepsilon_P}_i^2$:

$$S_P(A, B, C, D) = \frac{1}{B^2 + C^2 - 4AD} \sum_{i=1}^{n} (A(x_i^2 + y_i^2) + Bx_i + Cy_i + D)^2 \tag{15.32}$$

La clave consiste en imponer la restricción $B^2 + C^2 - 4AD = 1$, eliminando el denominador y retornando a un problema cuya solución es del tipo propuesto en (15.9) con:

$$\begin{aligned}
&\mathbf{p} = (A, B, C, D)^t \\
&\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_\mathrm{n})^t = \begin{bmatrix} x_1^2 + y_1^2 & x_1 & y_1 & 1 \\ \ldots & \ldots & \ldots & \ldots \\ x_n^2 + y_n^2 & x_n & y_n & 1 \end{bmatrix} \\
&\mathbf{M} = \mathbf{Z}^t\mathbf{Z} \qquad \mathbf{N} = \begin{bmatrix} 0 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -2 & 0 & 0 & 0 \end{bmatrix}
\end{aligned} \tag{15.33}$$

donde la única diferencia, aunque determinante, respecto al método de Kasa estriba en la matriz de restricciones **N**. Nótese que **N** en (15.33) es invertible, por lo que (15.9) se reduce a un problema normal de autovectores:

$$\mathbf{N^{-1}Mp} = \lambda \mathbf{N^{-1}Np} = \lambda \mathbf{p} \tag{15.34}$$

cuya solución es el autovector asociado al menor autovalor positivo de $\mathbf{N^{-1}M}$.

La resolución numérica del problema de autovectores (15.34) puede causar problemas numéricos (Chernov, 2010). Es mejor acometer el problema usando la descomposición SVD, $\mathbf{Z = U\Sigma V^t}$.

Sustituyendo términos en (15.34):

$$\begin{gathered} \mathbf{N^{-1}Mp} = \lambda \mathbf{p} \\ \mathbf{N^{-1}(V\Sigma^t U^t)(U\Sigma V^t)p} = \mathbf{N^{-1}V(\Sigma^t \Sigma)V^t p} = \lambda \mathbf{p} \\ \mathbf{(\Sigma V^t)N^{-1}(V\Sigma^t)(\Sigma V^t)p} = \lambda \mathbf{(\Sigma V^t)p} \\ \mathbf{Y^t N^{-1} Y q} = \lambda \mathbf{q}, \text{con} \qquad \mathbf{Y = V\Sigma^t} \quad \text{y} \quad \mathbf{q = \Sigma V^t p} \end{gathered} \tag{15.35}$$

Tras resolver el problema de autovectores $\mathbf{Y^t N^{-1} Y q} = \lambda \mathbf{q}$ en (15.35) la solución se obtiene simplemente como $\mathbf{p = V\Sigma^{-1} q}$, donde $\mathbf{q}$ será el autovector ligado al menor autovalor no negativo.

Por razones de espacio, no presentamos ningún método para el ajuste de elipses. En cualquier caso, la estrategia para la resolución algebraica sigue el patrón aquí expuesto, basado en la solución general (15.9). El lector interesado puede encontrar en las referencias Taubin (1991), Gander (1994), Fitzgibbon y col. (1995), Halif y Flusser (1998) y Kanatani y Rangarajan (2010) abundante información tanto para el ajuste algebraico de circunferencias como de elipses.

15.2. Detección de patrones geométricos con RANSAC

El ajuste por mínimos cuadrados es muy sensible a la presencia de puntos que no se ajustan al modelo, conocidos como puntos espurios o atípicos (*outliers* en inglés), proporcionando unos resultados poco precisos o incluso totalmente inservibles en estas situaciones. En procesamiento de imágenes es fundamental tener en cuenta este hecho porque, además de los píxeles del patrón que queremos aproximar, normalmente aparecerán otros. Estos últimos deben ser descartados a la hora de calcular el modelo geométrico para obtener unos resultados mínimamente fiables. El método de estimación robusta RANSAC es un procedimiento que precisamente minimiza la influencia de los *outliers*.

RANSAC es una abreviatura de RANdom SAmple and Consensus, que podría traducirse como "consenso de la muestra tomada al azar". RANSAC es un algoritmo de estimación robusta que permite hallar un modelo matemático a partir de datos contaminados con numerosos valores espurios que no se ajustan al modelo. El algoritmo fue publicado por Fischler y Bolles en 1981 y desde entonces se ha aplicado profusamente en el análisis de imágenes (Fischler y Bolles, 1981). RANSAC presenta una extraordinaria capacidad para proporcionar un buen ajuste a partir de datos contaminados con grandes proporciones de *outliers*, muy superiores incluso al 50%.

El algoritmo se basa en suponer que entre el conjunto de datos existe un porcentaje suficiente de estos, los *inliers*, cuyo uso permite estimar los parámetros del modelo con la precisión requerida. La idea consiste en seleccionar aleatoriamente muestras mediante un procedimiento probabilista evaluando su ajuste al modelo.

El Algoritmo 15.1 presenta el pseudocódigo de RANSAC. La función ***RANSAC()*** busca el mejor modelo considerando los *n* puntos de contorno *puntos*. Para ello, de forma iterativa lleva a cabo dos etapas: *hipótesis* y *verificació*n.

- *Hipótesis*: Mediante la función ***SELECCIONALEATORIA()*** se obtiene aleatoriamente del conjunto de *n* puntos de contorno una muestra *m* de tamaño *dim*, siendo *dim* los puntos necesarios para establecer los parámetros del modelo: (2 para una recta, 3 para una circunferencia, 5 para una elipse…). Para evitar escoger puntos que, siendo inliers, proporcionen estimaciones poco precisas, es habitual introducir un parámetro *dist_minima*, que garantice que los puntos seleccionados estén suficientemente alejados unos de otros. Los *parametros* del modelo (A, B, C para una recta, R, c_x, c_y para una circunferencia, etc.) se evalúan para la *muestra* usando la función ***CALCULAPARAMETROS()***.
- *Verificación*: Una vez calculado el vector de parámetros, la función ***CALCULACONSENSO()*** obtiene un vector *conjunto_consenso*, el número de *inliers* asociado *num_inliers* y el vector con los *errores* en cada punto. El vector *conjunto_consenso* es un vector de tamaño *n*. En la posición *i*, *conjunto_consenso(i)* será 1 si *puntos(i,:)* pertenece al conjunto de consenso, es decir, su error respecto al modelo descrito por *parametros* es menor que la tolerancia prefijada *tol.* Será 0 en caso contrario. Si *num_inliers* es superior a *min_num_inliers*, un valor prefijado de antemano para considerar válido el modelo obtenido, asignamos una *puntuacion* al conjunto de consenso mediante la función ***CALCULAPUNTUACION()***. La puntuación en el algoritmo clásico suele ser el propio valor *num_inliers*, pero como comentaremos existen mejores opciones. Si la nueva puntuación es mejor que la anterior, entonces se reemplazan los datos asociados al mejor conjunto de consenso hasta este momento por el nuevo.
-

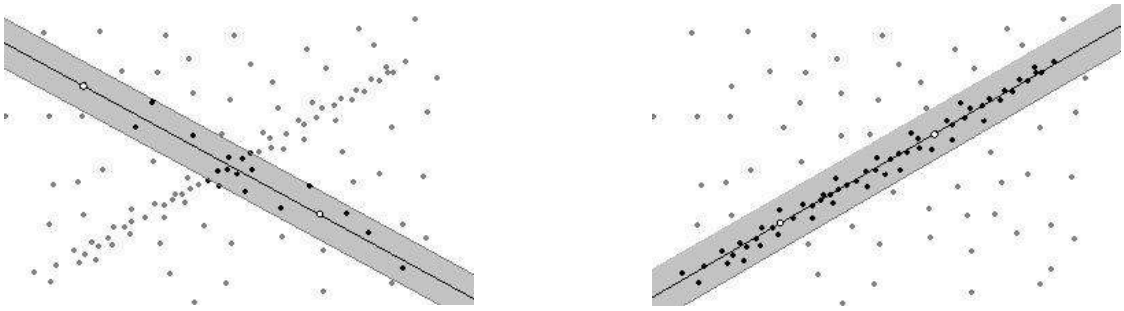


Figura 15.1. En el caso de la recta, RANSAC selecciona aleatoriamente 2 puntos (en blanco en la figura) y calcula el conjunto de consenso formado por los puntos situados a una distancia inferior a una tolerancia *tol* prefijada). Tras un número prefijado de iteraciones, el modelo elegido es aquel que logra un mayor conjunto de consenso (derecha).

→ *puntos*: Matriz de *n* puntos de contorno (x_i, y_i) de tamaño *(n,2)*
dim: Dimensión del modelo (2 rectas, 3 circunferencias, 5 elipses,…)
tol: Tolerancia de ajuste al modelo
min_num_inliers: Número mínimo de *inliers* para considerar válido el modelo
max_iter Número máximo de iteraciones
dist_minima Distancia mínima entre puntos en la selección aleatoria
← *mejor_conjunto_consenso*: Vector de *n* valores: 0 el punto no es del consenso, 1 si lo es
fallo: No se ha detectado un conjunto de consenso válido

RANSAC *(puntos,dim,tol,min_num_inliers,max_iter,dist_minima)*
fallo=1
num_iter=0
mejor_puntuacion=$-n \cdot tol^2$
mejor_ conjunto_consenso=**0**
Mientras *num_iter<max_iter* %Puede añadirse también un TIMEOUT como condición de salida
num_iter=*num_iter*+1
%HIPOTESIS
muestra=***SELECCIONALEATORIA(****puntos****,****dim,dist_minima****)*** %Selecciona muestra *m* de tamaño *dim*
parametros=***CALCULAPARAMETROS(****muestra****)***
%VERIFICACIÓN
(conjunto_consenso,num_inliers,errores)=***CALCULACONSENSO(****puntos,parametros,tol****)***
Si *num_inliers*≥*min_num_inliers*
fallo=0
puntuacion=***CALCULAPUNTUACION(****conjunto_consenso,errores,tol****)***
Si *puntuacion>mejor_puntuacion*
mejor_puntuacion=puntuacion
mejor_conjunto_consenso=conjunto_consenso
Fin
Fin
Fin
Devolver *mejor_conjunto_consenso, fallo*

Algoritmo 15.1. Pseudocódigo del algoritmo RANSAC.

El algoritmo necesita tres parámetros para controlar el proceso de estimación del modelo: la tolerancia *tol*, el número mínimo de *inliers min_num_inliers* y el número máximo de iteraciones *max_iter*.

La tolerancia *tol* se establece en base a los requerimientos específicos de cada aplicación. El valor de *tol* no debe ser excesivamente bajo para poder asumir los errores del ruido y la digitalización en los puntos de contorno, pero tampoco muy elevado de forma que puntos espurios sean tomados como *inliers*.

El umbral *min_num_inliers* dependerá en gran medida de la aplicación en particular. Si estamos estimando los parámetros de una circunferencia a partir de un arco pequeño, el umbral deberá ser también pequeño. Si sabemos que la circunferencia aparece completa en la imagen y tenemos información *a priori* de su radio en píxeles, el umbral deberá tener un valor relativamente próximo a la longitud de la circunferencia en píxeles.

El cálculo del número *max_iter* es clave en el algoritmo y lo describiremos a continuación.

15.2.1. Determinación del número máximo de iteraciones

El número máximo de iteraciones *max_iter* puede determinarse teóricamente a partir de la probabilidad de encontrar al menos una muestra de puntos no contaminada por *outliers*. RANSAC no es un algoritmo determinista y, por tanto, el modelo que proporciona tendrá más probabilidades de ser válido a medida que se lleven a cabo más iteraciones.

Sea q la probabilidad de que al elegir aleatoriamente una muestra *m* formada por *dim* datos, esta esté completamente formada por *inliers*. Por tanto, la probabilidad de elegir una muestra con al menos un outlier será $(1-q)$. Si tomamos k muestras aleatorias, la probabilidad de que todas estén contaminadas por *outliers* será $(1-q)^k$.

Dado que cuando $k \to \infty$ entonces $(1-q)^k \to 0$, la idea es escoger un valor de k lo suficientemente alto para que $(1-q)^k < \varepsilon$, donde ε, la probabilidad de que el algoritmo acabe proporcionando un resultado erróneo, es un umbral suficientemente pequeño.

Tomando logaritmos y despejando concluimos que:

$$max\ \_iter = \boldsymbol{CEIL}\,(k) = \boldsymbol{CEIL}\left(\frac{\log(\varepsilon)}{\log(1-q)}\right) \tag{15.36}$$

El valor de q depende del número real de *inliers* $n_{inliers}$ presentes en el conjunto de *n* datos, que obviamente es también desconocido. Puesto que las muestras tienen tamaño *dim* y suponiendo que todas las muestras tienen la misma probabilidad de ser escogidas, la probabilidad *q* vendrá dada por:

$$q(n_{inliers}) = \frac{\binom{n_{inliers}}{dim}}{\binom{n}{dim}} = \prod_{i=0}^{dim-1} \frac{n_{inliers}-i}{n-i} \approx \left(\frac{n_{inliers}}{n}\right)^{dim} \tag{15.37}$$

donde en la aproximación (15.37) suponemos $n_{inliers} \gg dim$. Es decir, el valor de q es aproximadamente igual a la probabilidad de escoger con reinserción *dim inliers* de forma consecutiva.

Por tanto, inicialmente el valor *max_iter*, entrada a la función RANSAC(), deberá ser calculado a partir del porcentaje de *inliers* estimado $(n_{inliers}/n)$ y de la tasa ε, que determina la probabilidad de que el algoritmo acabe dando un resultado erróneo.

Aunque por claridad no se ha reflejado en el pseudocódigo, si durante el proceso se descubre un conjunto de consenso con $num\_inliers > n_{inliers}$ podemos actualizar el valor *max_iter*, reduciendo así su valor.

El algoritmo puede incorporar otros criterios de parada, además del número máximo de iteraciones. Por ejemplo, exigir un número suficiente de *inliers* a partir del cual consideramos bueno el modelo. En aplicaciones con tiempos de ciclo muy exigentes, puede establecerse un intervalo de tiempo prefijado (*timeout*), a partir del cual se da el mejor modelo disponible hasta el momento.

Por el contrario, supongamos que, una vez calculado el valor de *max_iter,* el tiempo de cálculo asociado al algoritmo RANSAC t_{ransac} es inferior al t_{ciclo} disponible en nuestra aplicación. En ese caso, podemos aumentar el margen de seguridad del algoritmo incrementando *max_iter*.

15.2.2. Cálculo de la puntuación

Este es el cometido de la función ***CALCULAPUNTUACION()*** del Algoritmo 15.1. En el algoritmo clásico RANSAC la puntuación obtenida viene dada por *num_inliers* en el conjunto de consenso.

A continuación vamos a ver cómo el algoritmo RANSAC puede reformularse como un M-estimador. Un M-estimador (Huber, 1981) sustituye el funcional clásico de minimización mediante mínimos cuadrados $\sum_{i=1}^{n} \mathrm{e}_i^2$ por un funcional del tipo $\sum_{i=1}^{n} \rho(\mathrm{e}_i^2)$ donde la función $\rho()$ es una función simétrica definida positiva y con un único mínimo en 0. Nótese que el método de mínimos cuadrados es también un M-estimador. El objetivo es usar una función $\rho()$ que amplifique menos los errores debido a los *outliers* que lo que hace la función cuadrática $(\cdot)^2$.

Así, el algoritmo RANSAC clásico puede verse como la minimización de la función de coste $S_R(\mathbf{e})$:

$$S_R(\mathbf{e}) = \sum_{i=1}^{n} \rho_R(\mathrm{e}_i^2) \qquad \text{con } \rho_R(\mathrm{e}^2) = \begin{cases} 0 & \text{si } \mathrm{e}^2 \leq tol^2 \\ 1 & \text{en caso contrario} \end{cases} \tag{15.38}$$

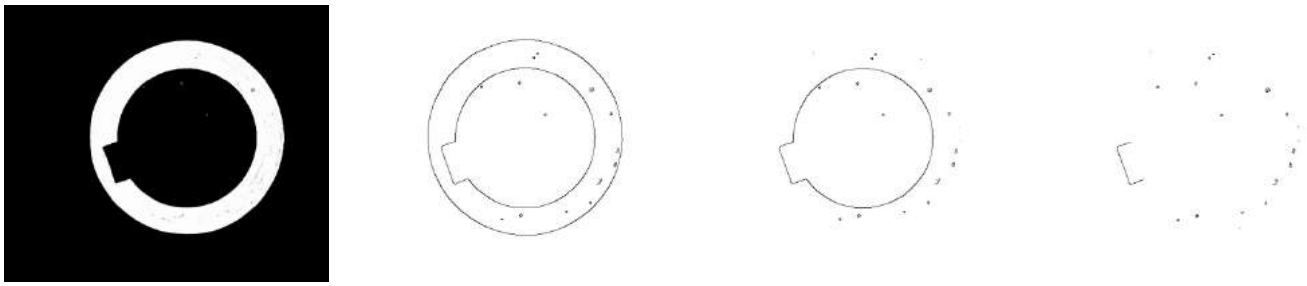
La propuesta introducida en (Torr y Zisserman, 2000) consiste en usar el M-estimador $\rho_T()$:

$$S(\mathbf{e}) = \sum_{i=1}^{n} \rho_T(\mathrm{e}_i^2) \qquad \text{con } \rho_T(\mathrm{e}^2) = \begin{cases} \mathrm{e}^2 & \text{si } \mathrm{e}^2 \leq tol^2 \\ tol^2 & \text{en caso contrario} \end{cases} \tag{15.39}$$

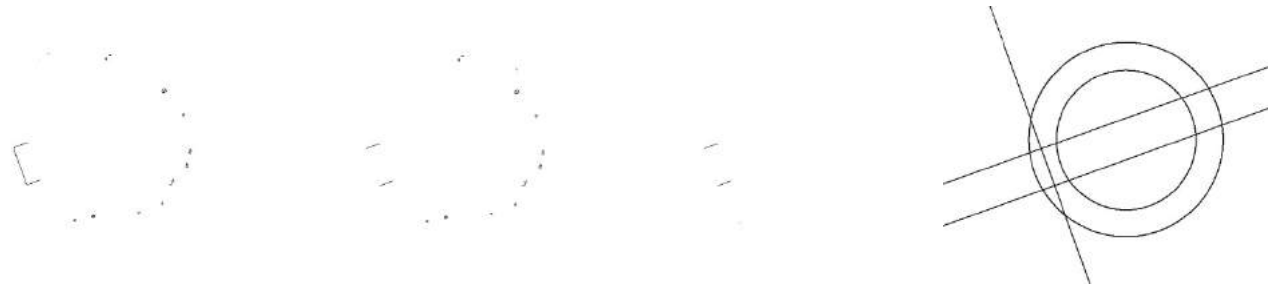
dando lugar a la variante del RANSAC conocida como MSAC, M-estimator SAmple and Consensus. Ahora los *inliers* contribuyen en la función a minimizar con su propio error, mientras que los *outliers* contribuyen con una magnitud fija. Esta formulación no supone prácticamente ningún sobrecoste respecto a la tradicional, pero permite escoger de una forma más lógica el conjunto de consenso.

Nótese que en el pseudocódigo estamos realmente maximizando la puntuación, por lo que basta con escoger $-\rho_T()$. Véase que el valor inicial *mejor_puntuacion* de la función RANSAC() es $-n \cdot tol^2$ que correspondería a una muestra cuyo conjunto de consenso está vacío.

Los parámetros estimados con RANSAC pueden no ser muy precisos, por lo que se recalculan inyectando los *inliers* del mejor conjunto de consenso a un procedimiento de mínimos cuadrados.



(a) (b) (c) (d)



(e) (f) (g) (h)

Figura 15.2. (a) Imagen original. (b) Puntos de contorno detectados. De (c) a (g) se muestran los puntos de contorno tras varios filtrados. En (c) se ha suprimido el conjunto de consenso de la circunferencia externa. En (d) se ha eliminado el conjunto de consenso de la circunferencia interna. En (e) se muestran solo los píxeles en la corona que definen estas dos circunferencias. En (f) se ha suprimido el conjunto de consenso del segmento de chaveta de mayor longitud. La recta detectada divide la imagen en dos semiplanos y en (g) solo se han mantenido los píxeles que aparecen en el mismo semiplano que el centro y a una distancia inferior a un umbral preestablecido. En (h) aparecen los modelos detectados en el proceso (véase Algoritmo 15.2).

15.3. Aplicación práctica

Veamos un ejemplo práctico de ajuste a rectas y circunferencias con RANSAC. Supongamos que tenemos la imagen de la figura 15.2(a) correspondiente a una arandela con chavetero. En una hipotética inspección dimensional de esta pieza tendríamos que determinar los modelos de las circunferencias interior y exterior de la arandela así como los tres segmentos que definen el chavetero.

En este tipo de aplicaciones, donde son varios los modelos que hay que detectar en la imagen, lo más conveniente es empezar buscando la primitiva geométrica que tiene una mayor probabilidad de ser detectada, es decir, aquella que presenta un mayor número de puntos en la imagen. A continuación se eliminarán de la imagen los puntos que se han ajustado a este modelo y se procederá con la siguiente primitiva con más presencia de puntos en la imagen.

En el ejemplo que nos ocupa, detectaremos en primer lugar las circunferencias, a continuación el segmento largo del chavetero y, finalmente, los segmentos cortos del chavetero.

El Algoritmo 15.2 muestra en pseudocódigo el proceso de ajuste para cada una de las figuras. Según se van detectando las primitivas, los *inliers* correspondientes son eliminados del conjunto de puntos de contorno inicial (Figuras 15.2(b) a 15.2(g)). Por otro lado, a partir de cada uno de los conjuntos de *inliers* detectados, se obtienen los parámetros definitivos de las figuras mediante un procedimiento de mínimos cuadrados como los descritos más arriba.

→*puntos*: Matriz de *n* puntos de contorno (x_i, y_i) de tamaño *(n,2)*
param_dim_tol: Parámetros de dimensiones y tolerancias admisibles
param_circ: Parámetros de entrada algoritmo RANSAC de las circunferencias. Véase Algoritmo RANSAC.
param_rect1: Parámetros de entrada algoritmo RANSAC de segmento chaveta largo. Véase Algoritmo RANSAC.
param_rect2: Parámetros de entrada algoritmo RANSAC de segmento chaveta corto. Véase Algoritmo RANSAC.
←*(xc1,yc1,R1)*: Vector con los parámetros de la circunferencia externa
(xc2 yc2,R2): Vector con los parámetros de la recta del circunferencia interna
(A1,B1,C1): Vector con los parámetros del segmento chaveta largo
(A2,B2,C2): Vector con los parámetros del segmento chaveta corto 1
(A3,B3,C3): Vector con los parámetros del segmento chaveta corto 2
error: Variable con código indicando tipo de error producido

ARANDELACONCHAVETA*(puntos,param_dim_tol ,param_circ, param_rect1, param_rect2)*
cont=1
fallo=0
Mientras *cont<=2* **Y** *fallo==0* %Buscamos 2 circunferencias
*(conjunto_consenso,fallo)=**RANSACCIRCUNFERENCIA**(puntos, param_circ)*
Si *fallo==0*
*(puntos,puntos_consenso)=**OBTIENEYBORRAPUNTOSCONSENSO** (puntos,conjunto_consenso)*

*(xc(cont),yc(cont),R(cont))=**MINIMOSCUADRADOSCIRCUNFERENCIA**(puntos_consenso)*
cont=cont+1
Fin
Fin
*(xc1,yc1,R1, xc2,yc2,R2,error)=**VALIDACIRCUNFERENCIAS**(cont,xc,yc,R,param_dim_tol)*
Si *error>0*
Devolver *error*
Fin
*(puntos)=**FILTRAPUNTOSCORONA** (puntos, xc1,yc1,R1, xc2,yc2,R2)*
*(conjunto_consenso,fallo)=**RANSACRECTA**(puntos,param_rect1)*
Si *fallo==0*
*(puntos,puntos_consenso)=**OBTIENEYBORRAPUNTOSCONSENSO** (puntos,conjunto_consenso)*
*(A1,B1,C1)=**MINIMOSCUADRADOSRECTA**(puntos_consenso)*
Fin
*(A1,B1,C1,error)=**VALIDASEGMENTOCHAVETALARGO** (fallo,A1,B1,C1,xc1,yc1,R1,param_dim_tol)*
Si *error>0*
Devolver *error*
Fin
%Eliminamos los puntos del semiplano de la recta chaveta larga contrario al de los centros de la circunferencias
%y aquellos situados a una distancia superior a la marcada en los parámetros de tolerancia
*(puntos)=**FILTRAPUNTOSSEMIPLANOSEGMENTOCHAVETALARGO** (puntos,A1,B1,C1,xc2,yc2,R2, param_dim_tol)*
cont=1
Mientras *cont<=2* **Y** *fallo==0* %Buscamos los 2 segmentos cortos de la chaveta
*(conjunto_consenso,fallo)= **RANSACRECTA** (puntos,param_rect2)*
Si *fallo==0*
*(puntos,puntos_consenso)=**OBTIENEYBORRAPUNTOSCONSENSO** (puntos,conjunto_consenso)*
*(A(cont),B(cont),C(cont))=**MINIMOSCUADRADOSRECTA**(puntos_consenso)*
cont=cont+1
Fin
Fin
*(A2,B2,C2,A3,B3,C3,error)=**VALIDASEGMENTOSCHAVETACORTOS** (cont, A,B,C,param_dim_tol)*

Algoritmo 15.2. Pseudocódigo del algoritmo de detección de Arandela con Chaveta.

	Num. Puntos contorno n	Núm. *inliers* $n_{inliers}$	Porcentaje aprox. $\frac{n_{inliers}}{n}$ (para cualquier imagen)	*max_iter* (15.36)	*max_iter* fijado en programa
Circunf. Exterior	2362	1227	40%	140	200
Circunf. Interior	1135	805	60%	38	50
Segmento Mayor	330	279	75%	12	25
Segmento Menor1	51	25	40%	53	75
Segmento Menor2	26	26	60%	21	40

Tabla 15.1. Número total de puntos de contorno y de *inliers* en las estimaciones de las distintas primitivas para el ejemplo de la Figura 15.2. Experimentalmente se ha obtenido el porcentaje aproximado $n\_inliers/n$ que aparecerá en cualquier imagen de la aplicación. Los valores $max\ \_iter$ se han hallado a partir de la ecuación (15.36). Los valores finalmente prefijados en el programa aplicando unos márgenes de seguridad "*ad hoc*" se muestran en la última columna.

A modo de ejemplo, para la circunferencia externa se determina experimentalmente que en las imágenes adquiridas la proporción de *inliers* es aproximadamente del 40%. La probabilidad q de que una muestra de tamaño $dim = 3$ esté formada completamente por *inliers*, ecuación (15.37), es:

$$q(n_{inliers}) \approx \left(\frac{n_{inliers}}{n}\right)^{dim} = 0.4^3 = 0.064$$

Si escogemos que la probabilidad de que el algoritmo acabe proporcionando un resultado erróneo sea $\varepsilon = 10^{-4}$, el máximo número de iteraciones $max\ \_iter$, ecuación (15.36), será:

$$max\ \_iter = \boldsymbol{CEIL}\left(\frac{\log(\varepsilon)}{\log(1-q)}\right) = \boldsymbol{CEIL}\left(\frac{\log(10^{-4})}{\log(1-0.064)}\right) = 140$$

No obstante, escogeremos $max\ \_iter$ =200 para tener un margen de seguridad aún mayor. El número máximo de iteraciones para el resto de figuras se calcula análogamente teniendo en cuenta el número total de puntos que aún permanecen tras diferentes filtrados (Figura 15.2 y Tabla 15.1).

A partir de las dimensiones obtenidas (radios, centros y parámetros A, B y C de las tres rectas) podemos analizar la figura de la pieza y determinar si cumple los requisitos dimensionales prefijados en la aplicación, tales como cotas dentro de tolerancias, circularidad, excentricidad, anchura y profundidad de la chaveta, etc.

15.4. Conclusiones

Las aplicaciones industriales de visión artificial frecuentemente requieren de la detección de determinados patrones en la imagen. La detección de rectas, circunferencias o elipses no es una tarea trivial. Las técnicas tradicionales de mínimos cuadrados solo permiten el ajuste a un modelo determinado siempre que los puntos estén convenientemente aislados, puesto que la estimación de mínimos cuadrados es notoria por su falta de robustez frente a valores atípicos (*outliers*). Si aparecen valores atípicos en los datos, es preciso recurrir a métodos de regresión robusta como el método RANSAC.

RANSAC permite hallar un modelo matemático a partir de datos contaminados con numerosos valores que no se ajustan al modelo. RANSAC presenta una extraordinaria capacidad para proporcionar un buen ajuste con un porcentaje de *outliers* superior incluso al 50%. El algoritmo selecciona aleatoriamente muestras de puntos con los que genera hipótesis que son posteriormente verificadas evaluando el conjunto de consenso (número de puntos de la imagen próximos al modelo calculado). Si el número de puntos en el conjunto de consenso es superior a un umbral predeterminado, el modelo se asume como válido.

15.5. Bibliografía

Bookstein, F.L. (1979). Fitting Conic Sections to Scattered Data, *Computer Graphics and Image Processing*, 9, 56-71.

Chernov, N. (2010). *Circular and linear regression: Fitting circles and lines by least squares*. Chapman & Hall/CRC Monographs on Statistics and Applied Probability, 117.

Chernov, N. (2012). Código MATLAB para circunferencias y cónicas [online]. Obtenido el 20 sept 2012 de http://people.cas.uab.edu/~mosya/cl/index.html

Fischler M.A. y Bolles R.C. (1981). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Comm. of the ACM* 24, 6, 381–395.

Fitzgibbon, A.W. Pilu, M. y Fisher, R. (1999). *Direct least-squares fitting of ellipses, IEEE Transactions on Pattern Analysis and Machine Intelligence,* 21, 5, 476-480.

Gander, W., Golub, G. H. y Strebel, R. (1994). Least squares fitting of circles and ellipses, *BIT*, 34, 558–578.

Halif , R. y Flusser, J. (1998). Numerically stable direct least squares fitting of ellipses, in *Proc. Sixth Int'l Conf.Computer Graphics and Visualization*, 1, 125 –132.

Huber, P. J. (1981). *Robust statistics*. Wiley. New York.

Kanatani, K. y Rangarajan, P. (2010). Hyperaccurate Ellipse Fitting without Iterations *Proc. Int. Conf. Computer Vision Theory and Applications (VISAPP 2010)*, 2, 5–12.

Kasa, I. (1976). A curve fitting procedure and its error analysis. *IEEE Transactions on Instrumentation and Measurement*, 25, 8–14.

Madsen, K. Nielsen, H.B. y Tingleff, O. (2004). Methods for Non-Linear Least Squares Problems, 2nd edition. *Lecture Note IMM Department of Mathematical Modelling.* Technical University of Denmark.

Pratt. V. (1987). Direct least-squares fitting of algebraic surfaces. *Computer Graphics*, 21, 145–152.

Taubin, G. (1991). Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13, 1115–1138.

Torr, P.H.S. y Zisserman, A. (2000). MLESAC: A new robust estimator with application to estimating image geometry, *Computer Vision and Image Understanding* 78, 1, 138–156.

CAPÍTULO 16

CONTROL VISUAL

Jorge POMARES, Gabriel J. GARCÍA, Javier PÉREZ, Pablo GIL, Fernando TORRES

Universidad de Alicante, Departamento de Física, Ingeniería de Sistemas y Teoría de la Señal, Alicante, España

En este capítulo se describen los fundamentos de los sistemas de control visual así como las diferentes tipologías que presentan atendiendo a la ubicación del sistema de visión. Se describirá la principal clasificación realizada comúnmente en los sistemas de control visual: basados en posición y basados en imagen. Se indicará otra clasificación que separa los sistemas de control visual en función del diseño del lazo de control: indirecto y directo. De entre todos estos tipos, se prestará mayor atención a los controladores indirectos basados en imagen. Se detallará su formulación y se indicarán las principales consideraciones a tener en cuenta para su implementación. Además, se hará uso de una interfaz basada en Matlab para la evaluación de estos controladores. Esta interfaz, desarrollada por los autores, permite especificar la inicialización y condiciones de funcionamiento de una tarea de control visual. Finaliza el capítulo citando algunas de las principales aplicaciones prácticas que tienen en la actualidad los controladores visuales basados en imagen.

16.1. Introducción

El término control visual o su correspondiente en inglés '*visual servoing'* es empleado en la literatura para hacer referencia a la utilización de visión artificial para controlar el movimiento de un robot. Uno de los primeros trabajos en los que se usa el término '*visual servoing'* fue el desarrollado por Hill y Park (1979), donde se plantea la utilización del control en bucle cerrado introduciendo información visual en la realimentación. Sanderson y Weiss (1980) establecen una clasificación de los sistemas de control visual en basados en posición y en imagen, describiendo asimismo el primer sistema de control visual basado en imagen. Desde la aparición de estos primeros sistemas de control visual a principios de los 80, su evolución ha sido lenta. Sin embargo, en la última década han aparecido nuevos desarrollos debidos principalmente a las capacidades de procesamiento de los nuevos ordenadores y sistemas de visión artificial que permiten procesar una escena a una frecuencia cada vez mayor.

Los componentes más importantes que conforman un sistema de control visual son los que se representan en la Figura 16.1 y que a continuación se detallan dentro de una aplicación genérica de seguimiento y manipulación.

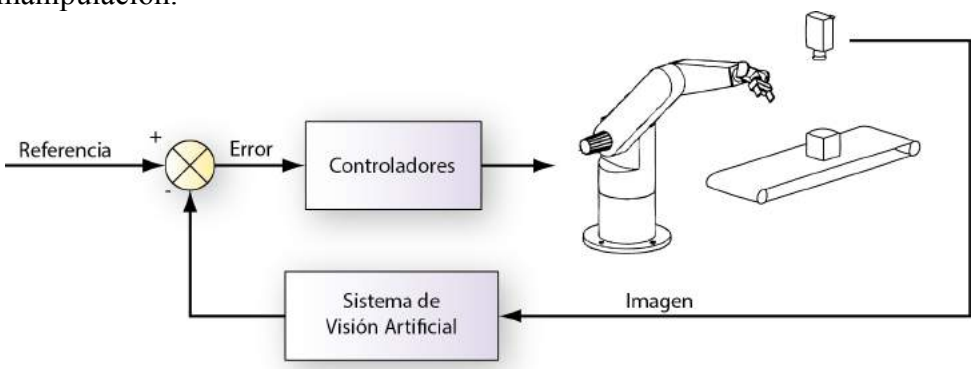


Figura 16.1. Esquema genérico de un sistema de control visual.

Los principales elementos que aparecen en la Figura 16.1 son los siguientes:

- *Referencia.* Se trata de la configuración final que se desea que alcance el robot. El tipo de referencia utilizada depende de si se emplea un controlador basado en posición (la referencia es una posición 3D a alcanzar) o si se emplea un controlador basado en imagen (la referencia es un conjunto de características visuales deseadas).
- *Controlador*. Tiene como función realizar el guiado del robot, haciendo uso de la información visual, hasta conseguir alcanzar la referencia. Para el caso indicado en la Figura 16.1, el controlador se encargará de guiar al robot hasta conseguir que se realice el agarre del objeto. El tipo de controlador dependerá, entre otras cosas, de si se realiza un control basado en posición, en imagen o si el control es directo o indirecto. Las principales características de estos controladores se indicarán en el Apartado 16.2.
- *Sistema de Visión Artificial.* Este bloque representa la realimentación del sistema de control visual y se encarga de realizar la extracción de la información visual requerida para guiar al robot. En el caso de un controlador basado en imagen, este componente simplemente extrae características visuales. Sin embargo, cuando se emplea control basado en posición, este bloque debe determinar la posición 3D del objeto implicado en la tarea a partir de la información visual capturada.

Para terminar con el estudio de los diferentes componentes que conforman los sistemas de control visual es necesario profundizar en uno de los elementos más importantes: la ubicación del sistema de visión. La información visual puede ser adquirida por una cámara montada directamente sobre el robot manipulador, en cuyo caso el movimiento del robot induce movimiento en la cámara; o la cámara puede ser fijada en el espacio de trabajo de manera que se puede observar el movimiento del robot a partir de una configuración estacionaria. Ambas configuraciones, que se representan en la Figura 16.2, se denominan respectivamente configuración en el extremo del robot (eye-in-hand) y configuración externa al robot (eye-to-hand). Cuando se emplea una cámara en el extremo del robot los objetos del espacio de trabajo se encuentran dentro de su campo visual, y se tiene una relación constante entre el sistema de coordenadas de la cámara y el del extremo del robot. Cuando se utiliza una cámara externa

al robot, la relación de posiciones entre la cámara y el sistema de coordenadas situado en la base del robot es fija. Por el contrario, no existe una relación constante entre las posiciones de la cámara y el extremo del robot. En este capítulo se empleará una configuración *'eye-in-hand'*, de forma que el objeto del cual se extraen las características se encuentra en el campo de visión de la cámara. Además, la relación entre los sistemas de coordenadas de la cámara y del extremo del robot se mantendrá constante a lo largo de la tarea. En cualquier caso, todos los desarrollos pueden extenderse fácilmente a una configuración *'eye-to-hand'*.

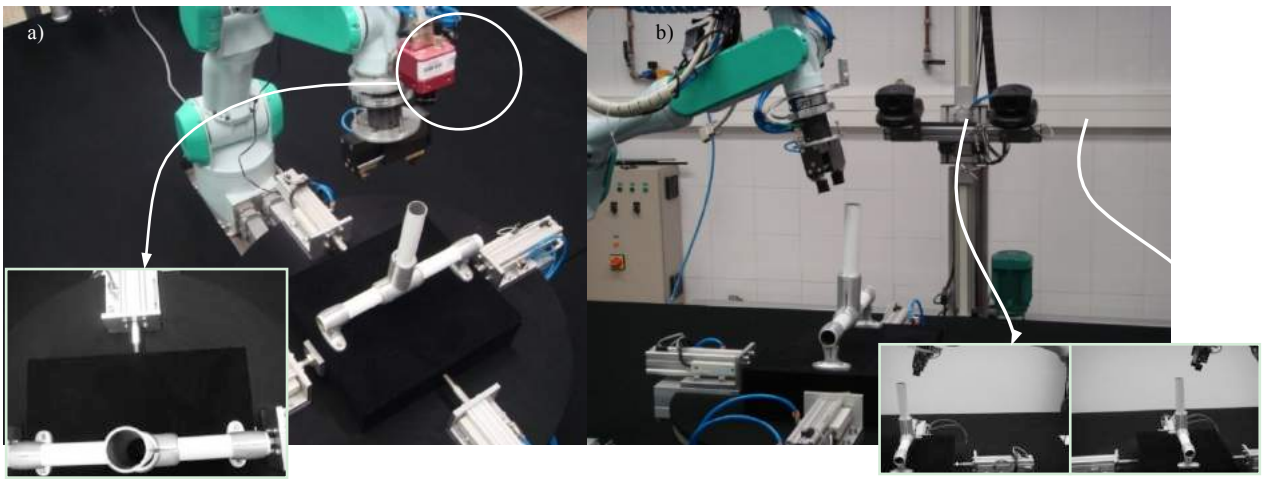


Figura 16.2. Posibles configuraciones de cámara. (*a*) cámara en el extremo del robot. (*b*) Cámara externa al robot.

16.2. Tipos de control visual

Este apartado tiene como objetivo realizar una revisión de los diversos tipos de sistemas de control visual. En primer lugar, se clasificarán los sistemas de control visual en basados en posición y basados en imagen. Los basados en posición son aquéllos en los que la referencia del sistema de control es una localización tridimensional deseada. Sin embargo, en los basados en imagen la referencia viene dada directamente en el espacio de la imagen. Esta clasificación inicial se realizará considerando una implementación "clásica" del controlador. Estos controladores clásicos se corresponden también con lo que suele denominarse como controladores indirectos. En este caso, la acción de control viene dada como velocidades a aplicar al robot, asumiendo la dinámica del mismo como despreciable. Cuando se aplica un controlador indirecto, se tienen en cuenta únicamente aspectos cinemáticos del robot. En los controladores directos la acción de control viene dada habitualmente como pares a aplicar a las articulaciones. En el Apartado 16.2.3 se indicarán las principales propiedades de los controladores directos basados en imagen. La mayoría de las aproximaciones existentes hasta la actualidad consideran una implementación indirecta de los controladores basados en imagen.

16.2.1. Control visual indirecto basado en posición

En este tipo de configuración, la entrada del regulador corresponde a la diferencia entre la localización deseada del objeto observado y la estimada a partir de las características extraídas por el

sistema de visión. Al trabajar el controlador en el espacio Cartesiano, el robot suele describir una trayectoria lineal entre la posición inicial y la deseada. Habitualmente, para reconstruir la posición y orientación 3D del objeto implicado en la tarea ($\boldsymbol{p}^C$,$\boldsymbol{\varphi}^C$) se ha de disponer de un modelo del espacio de trabajo, así como de un modelo calibrado de la cámara empleada. En este tipo de control visual, la referencia del sistema será la posición y orientación deseada del robot, o de un objeto existente en el espacio de trabajo.

En la Figura 16.3 se representa el esquema básico de un controlador basado en posición. Este esquema, que se corresponde con un controlador indirecto, presenta un primer nivel de realimentación proporcionada por los controladores de las articulaciones, mientras que el segundo nivel corresponde a la realimentación visual. En el esquema se indica la necesidad de reconstruir la localización 3D del objeto implicado en la tarea, ($\boldsymbol{p}^C$,$\boldsymbol{\varphi}^C$), a partir del conjunto de características extraídas, **s**. Más información de estos controladores puede consultarse en (Torres y col., 2002).

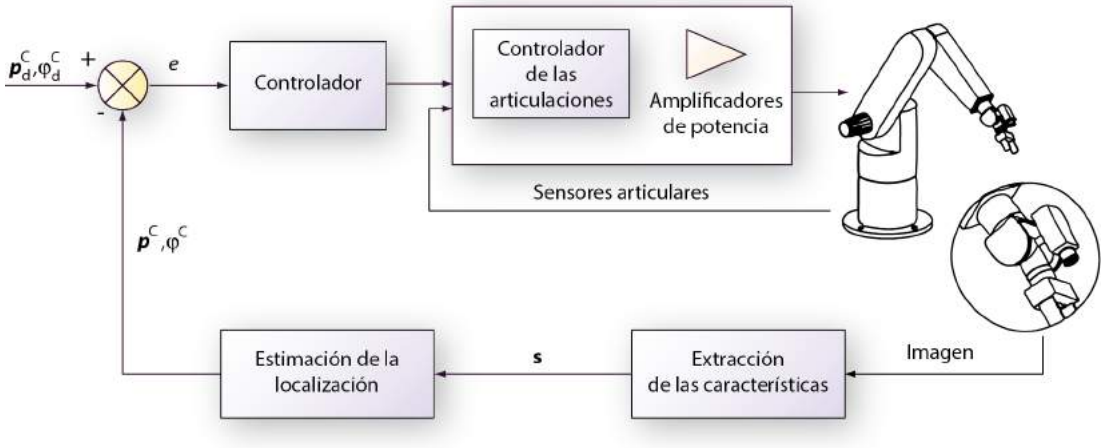


Figura 16.3. Controlador indirecto basado en posición.

16.2.2. Control visual indirecto basado en imagen

Cuando se emplea control visual basado en imagen, el regulador tiene como finalidad generar acciones de control de forma que las características visuales de las imágenes obtenidas vayan convergiendo progresivamente a las deseadas. En la Figura 16.4 se muestra el esquema de un controlador indirecto basado en imagen. Como se observa en esta figura, en la realimentación del bucle de control se realiza un proceso de extracción de características visuales, **s**, que en este caso se corresponde con cuatro puntos en la imagen. Por lo tanto, $\mathbf{s} = (\boldsymbol{f}_{1x}, \boldsymbol{f}_{1y}, \boldsymbol{f}_{2x}, \boldsymbol{f}_{2y}, \boldsymbol{f}_{3x}, \boldsymbol{f}_{3y}, \boldsymbol{f}_{4x}, \boldsymbol{f}_{4y})$, siendo $(\boldsymbol{f}_{ix}, \boldsymbol{f}_{iy})$ las coordenadas en la imagen de cada uno de los 4 puntos. A diferencia de los controladores basados en posición, no es necesario calcular la posición 3D del objeto implicado en la tarea. De esta manera, el control se realiza directamente en el espacio imagen. En este caso, el controlador genera un movimiento sobre el robot de forma que las características visuales describen una línea recta en el plano imagen entre las iniciales y las deseadas, no conociendo de antemano la trayectoria 3D que describirá el robot.

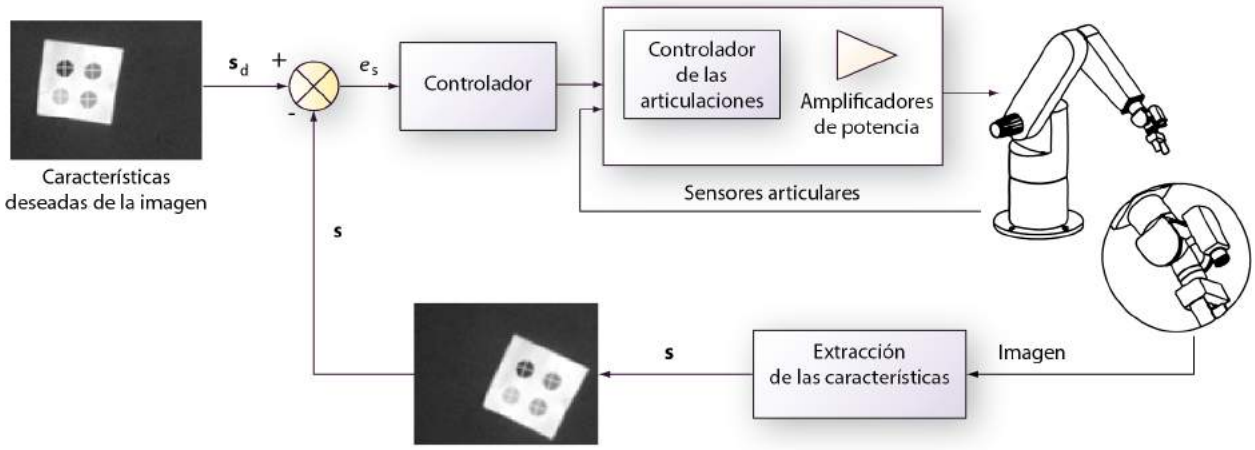


Figura 16.4. Controlador indirecto basado en imagen.

16.2.3. Control visual directo basado en imagen

Por último, para finalizar la clasificación de los sistemas de control visual, en la Figura 16.5 se indican los principales componentes de un sistema de control visual directo basado en imagen. En esta configuración, la referencia está compuesta por un conjunto de características visuales deseadas que habrían de observarse una vez que la tarea haya finalizado. El error $\mathbf{e}_s$, especificado en el espacio imagen, es la entrada al controlador. Este último se encarga de determinar los pares articulares necesarios (o corriente a aplicar a cada uno de los motores) para hacer converger las características extraídas hacia la deseadas. En los sistemas de control visual indirectos el controlador calcula la velocidad que se debe aplicar al extremo del robot, o la velocidad articular, para conseguir posicionar la cámara respecto a un objeto de referencia. Para ello, se hace uso del controlador interno del robot, que traduce esas velocidades a pares y fuerzas sobre las articulaciones. Sin embargo, empleando un esquema de control directo como el que se muestra en la Figura 16.5, se elimina el bucle interno de realimentación de los servomotores de forma que la información visual es empleada directamente para generar las corrientes o pares articulares a aplicar a cada uno de los motores del robot. Estos controladores integran no sólo la cinemática del robot sino que también tienen en cuenta el comportamiento dinámico. El resultado es un controlador más rápido que consigue reaccionar mejor ante cambios en la referencia o ante movimientos del objeto observado.

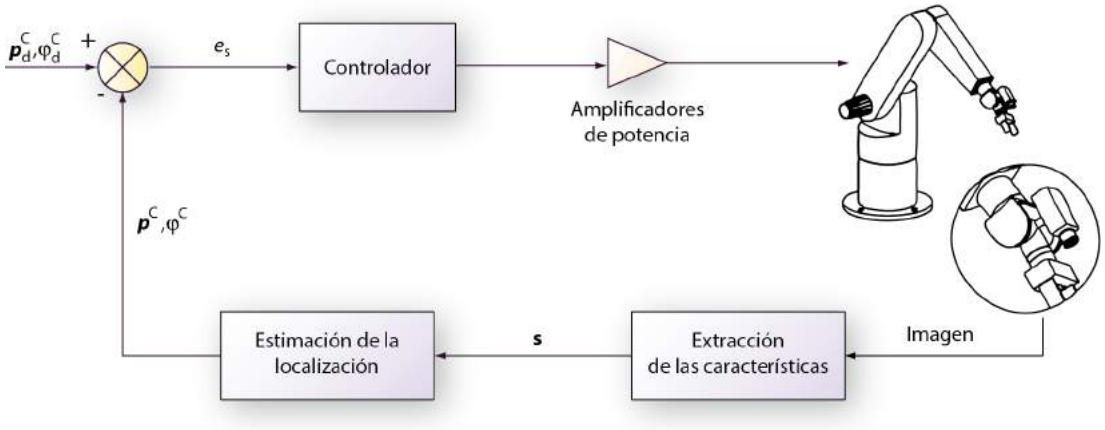


Figura 16.5. Controlador directo basado en imagen.

16.3. Control basado en imagen

En este apartado se describen las principales consideraciones a tener en cuenta para la implementación de un sistema de control visual indirecto basado en imagen.

16.3.1. Matriz de interacción

La matriz de interacción, $\mathbf{L}_s$, es empleada por los controladores visuales basados en imagen con el objetivo de relacionar la velocidad de un punto en el espacio tridimensional con la velocidad del punto correspondiente en el espacio imagen. Considerando una cámara que se encuentra observando un punto ubicado en el extremo de un robot en movimiento, la matriz de interacción ofrecería información acerca de cómo cambia la característica en el plano de la imagen (punto del extremo del robot observado en la imagen) cuando se produce un cambio en el extremo del robot en movimiento:

$$\begin{bmatrix} \dot{f}_x \\ \dot{f}_y \end{bmatrix} = \mathbf{L}_s\left(\boldsymbol{p}_E^C\right) \cdot \begin{bmatrix} \boldsymbol{v}_{tE}^C \\ \boldsymbol{w}_E^C \end{bmatrix} \tag{16.1}$$

Donde $\boldsymbol{p}_E^C$ se trata de un punto en el espacio 3D que se moverá con una velocidad de rotación $w_E^C(\dot{\alpha}_E^C, \dot{\beta}_E^C, \dot{\gamma}_E^C)$, así como también, una velocidad de traslación $v_{tE}^C(\dot{x}_{tE}^C, \dot{y}_{tE}^C, \dot{z}_{tE}^C)$, ambas con respecto al sistema de coordenadas de la cámara. Además $(\dot{f}_x, \dot{f}_y)$ es la variación de las coordenadas de la característica extraída en la imagen.

En un caso general, considerando $\boldsymbol{p}^C$ la posición de un punto de un objeto observado por la cámara en un espacio de dimensión j, $\boldsymbol{v}^C = \begin{bmatrix} \boldsymbol{v}_t^C & \boldsymbol{w}^C \end{bmatrix}$ será su velocidad respecto al sistema de referencia de la cámara (compuesta por velocidad de traslación y rotación). Se va a representar con **s** un vector de k características observadas en la imagen (medidas en píxeles), mientras que $\dot{s}$ será la variación de estas características en la imagen. La matriz de interacción se representará como $\mathbf{L}_s$ y definirá la siguiente transformación:

$$\dot{s} = L_s v^c \tag{16.2}$$

Desarrollando la expresión anterior, se obtiene la expresión que permite calcular esta matriz de interacción:

$$\mathbf{s} = \frac{\partial \mathbf{s}}{\partial \mathbf{r}} \cdot \frac{\partial \mathbf{r}}{\partial \mathrm{t}}, \dot{\mathbf{s}} = \begin{bmatrix} \dot{f}_{1\mathrm{x}} \\ \dot{f}_{1\mathrm{y}} \\ \vdots \\ \dot{f}_{\mathrm{kx}} \\ \dot{f}_{\mathrm{ky}} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_{1\mathrm{x}}}{\partial \mathbf{r}_1} & \cdots & \frac{\partial f_{1\mathrm{x}}}{\partial \mathbf{r}_\mathrm{j}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{\mathrm{ky}}}{\partial \mathbf{r}_1} & \cdots & \frac{\partial f_{\mathrm{ky}}}{\partial \mathbf{r}_\mathrm{j}} \end{bmatrix} \cdot \boldsymbol{v}^C \tag{16.3}$$

Como se observa en la expresión (16.3), el número de columnas en la matriz de interacción variará dependiendo de la tarea (se ha considerado un espacio de dimensión *j*). En general, la matriz tendrá 2k filas y j columnas y no se puede asegurar que sea cuadrada.

El valor de la matriz de interacción para el caso de que las características extraídas sean puntos ha sido determinado en numerosos trabajos previos, véase por ejemplo los trabajos de Chaumette y Hutchinson (2006). En cualquier caso, siguiendo la notación empleada, esta matriz de interacción adquiere el siguiente valor para el caso de que únicamente se extraiga un punto característico de la imagen con coordenadas $\left(f_\mathrm{x}, f_\mathrm{y}\right)$:

$$\dot{\mathbf{s}} = \begin{bmatrix} \frac{f_u}{z_P^C} & 0 & -\frac{\left(f_x - u_0\right)}{z_P^C} & -\frac{\left(f_x - u_0\right)\left(f_y - v_0\right)}{f_v} & \frac{\left(f_x - u_0\right)^2 + f_u^2}{f_u} & -\frac{f_u\left(f_y - v_0\right)}{f_v} \\ 0 & \frac{f_v}{z_P^C} & -\frac{\left(f_y - v_0\right)}{z_P^C} & -\frac{\left(f_y - v_0\right)^2 + f_v^2}{f_v} & \frac{\left(f_x - u_0\right)\left(f_y - v_0\right)}{f_u} & \frac{f_v\left(f_x - u_0\right)}{f_u} \end{bmatrix} \begin{bmatrix} \dot{x}_t^C \\ \dot{y}_t^C \\ \dot{z}_t^C \\ \dot{\alpha}^C \\ \dot{\beta}^C \\ \dot{\gamma}^C \end{bmatrix} \tag{16.4}$$

Como se observa en la Ecuación (16.4), la matriz de interacción depende de parámetros intrínsecos como el punto principal $\left(u_0, v_0\right)$ y f_u, f_v, siendo $\mathrm{f}_u = \mathrm{f} \cdot s_x$ y $\mathrm{f}_v = \mathrm{f} \cdot s_y$ donde f es la focal y $\left(s_x, s_y\right)$ es el escalado del plano imagen con respecto al sensor. Por último, z_P^C es la profundidad o distancia de la cámara a la característica.

Supóngase el caso en el que el sistema de visión es capaz de extraer k puntos característicos de la imagen. En este caso, la matriz de interacción obtenida se puede generalizar para varios puntos de la siguiente manera:

$$\mathbf{L}_s = \begin{bmatrix} \mathbf{L}_{s1}(\boldsymbol{p}_1) \\ \mathbf{L}_{s2}(\boldsymbol{p}_2) \\ \vdots \\ \mathbf{L}_{sk}(\boldsymbol{p}_k) \end{bmatrix} \tag{16.5}$$

siendo $\mathbf{L}_{si}$ la matriz de interacción para cada uno de los k puntos característicos extraídos.

16.3.2. Controlador indirecto basado en imagen

En este apartado se describen las principales consideraciones a tener en cuenta a la hora de implementar un controlador indirecto basado en imagen cuando las características visuales extraídas son un conjunto de puntos. Este conjunto de puntos se representarán con la variable **s**. Se desea que el controlador visual aplique las acciones de control oportunas para que el conjunto **s** vaya progresivamente alcanzando el valor de las características deseadas $\mathbf{s}_d$. Para ello, se define una función de error que se desea ir reduciendo progresivamente:

$$\boldsymbol{e}(\boldsymbol{r},t) = \hat{\mathbf{L}}_s^+ \left(\mathbf{s}(\boldsymbol{r},t) - \mathbf{s}_d\right), \tag{16.6}$$

La función de error, ***e***, indicada en la Ecuación (16.6) representa el error que se desea disminuir progresivamente en el espacio tridimensional. Este error se obtiene a partir del error medido en el espacio imagen $\boldsymbol{e}_s(\boldsymbol{r},t) = \mathbf{s}(\boldsymbol{r},t) - \mathbf{s}_d$. En esta expresión del error se ha indicado explícitamente que tanto el error como las características extraídas en cada iteración del bucle de control dependen de la posición de la cámara del extremo del robot, ***r***, y del tiempo, t. La relación entre ambos errores viene dada por la estimación de la pseudoinversa de la matriz de interacción, $\hat{\mathbf{L}}_s^+$. Se podría haber utilizado la inversa bajo la suposición de que la matriz de interacción sea cuadrada (2k = j), no singular y por tanto exista la inversa. Para que ocurra esto el número de puntos característicos extraídos en la imagen debe ser 3, con lo que el Jacobiano de la imagen o matriz de interacción será de 6 x 6.

Si se observan k características en la imagen de manera que 2k > j, se utiliza la siguiente pseudoinversa:

$$\mathbf{L}_s^+ = \left(\mathbf{L}_s^T \cdot \mathbf{L}_s\right)^{-1} \cdot \mathbf{L}_s^T. \tag{16.7}$$

En el caso de que 2k < j, la matriz pseudoinversa utilizada tiene el siguiente valor:

$$\mathbf{L}_s^+ = \mathbf{L}_s^T \cdot \left(\mathbf{L}_s \cdot \mathbf{L}_s^T\right)^{-1}. \tag{16.8}$$

Con $\hat{\mathbf{L}}_s$ se representa una estimación de la matriz de interacción ya que, como se comentó anteriormente, depende de la distancia de la cámara al objeto seguido, y en muchas ocasiones se considera que el parámetro de profundidad en esta matriz es constante (igual a la correspondiente a la posición final deseada). Además, los parámetros intrínsecos no son calculados con total exactitud sino que se emplea una aproximación a los mismos.

Se desea que la función de tarea decrezca de manera exponencial, de forma que se cumpla:

$$\dot{e} = -\lambda e \tag{16.9}$$

siendo λ una ganancia positiva. Por otro lado, como $\boldsymbol{e}$ es función de la localización de la cámara, $\boldsymbol{r}$, y del tiempo, t, su derivada se puede expresar de la siguiente manera:

$$\dot{e} = \left(\frac{\partial e}{\partial r}\right) v^C + \frac{\partial e}{\partial t} \tag{16.10}$$

donde v^C es la velocidad de la cámara, y $\frac{\partial \boldsymbol{e}}{\partial t}$ representa las variaciones en el error debidas al movimiento del objeto. Por otro lado, a partir de la definición de la función de tarea se puede afirmar que $\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{r}}$ viene dado por:

$$\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{r}} = \hat{\mathbf{L}}_s^+ \mathbf{L}_s. \tag{16.11}$$

A partir de las Ecuaciones (16.9) y (16.10) es posible obtener la siguiente expresión para la velocidad de la cámara:

$$\boldsymbol{v}^C = \left(\frac{\partial \hat{\boldsymbol{e}}}{\partial \boldsymbol{r}}\right)^+ \left(-\lambda \boldsymbol{e} - \frac{\partial \hat{\boldsymbol{e}}}{\partial t}\right), \tag{16.12}$$

donde:

- $\left(\frac{\partial \hat{\boldsymbol{e}}}{\partial \boldsymbol{r}}\right)$ puede ser igualado a la matriz identidad ya que:

$$\frac{\partial \hat{\boldsymbol{e}}}{\partial \boldsymbol{r}} = \hat{\mathbf{L}}_s^+ \hat{\mathbf{L}}_s = \mathbf{I}_6, \tag{16.13}$$

- $\frac{\partial \hat{\boldsymbol{e}}}{\partial t}$ es una estimación del movimiento del objeto en la imagen.

Se puede concluir finalmente que la ley de control obtenida para un controlador visual indirecto basado en imagen sería la siguiente:

$$\boldsymbol{v}^C = -\lambda \hat{\mathbf{L}}_s^+ \boldsymbol{e} - \frac{\partial \hat{\boldsymbol{e}}}{\partial t} \tag{16.13}$$

Ejemplo 16.1. *Se pretende realizar el posicionamiento de un robot Mitsubishi PA10 de 6 grados de libertad (g.d.l.) utilizando un esquema de control visual basado en imagen. El Mitsubishi PA-10 es un*

robot cuyo diseño está orientado al campo de la investigación en el área de robótica industrial. Se equipa el robot con una cámara en su extremo: una Smart Camera de National Instruments NI1744. Una calibración previa de la cámara permite conocer sus parámetros intrínsecos: focal de 8mm, escalado 4.65 x 4.65 µm, resolución de 1280x1024 píxeles y punto principal (640, 512) píxeles. Para el posicionamiento del robot se utilizará un patrón de 4 puntos que conformarán las características visuales del controlador. Se pide: obtener la evolución de la señal enviada por el controlador; la evolución de las características visuales en el plano imagen; la trayectoria 3D del extremo del robot; así como la evolución del error del controlador.

Solución: Para la resolución del ejercicio se va a utilizar ViSeC_gui, un GUI de Matlab que utiliza las toolbox Robotics y Machine Vision de Peter Corke (Perez, 2015). El entorno visual guía al usuario en la definición de la tarea de control visual. El resultado del proceso es la simulación de la tarea de posicionamiento que ha definido el usuario. A partir de esta simulación se pueden obtener distintas gráficas que permitirán observar el funcionamiento del controlador.

En primer lugar, el entorno ofrece la posibilidad de elegir el robot sobre el que se desea situar la cámara. El entorno ofrece la posibilidad de elegir dos robots de 6 g.d.l. que están precargados: el Mitsubishi PA10 y el Puma 560 (ver Figura 16.6). La aplicación permite también crear un nuevo robot, de hasta 7 g.d.l. para posicionar la cámara. En el ejemplo que se propone aquí se selecciona directamente el robot Mitsubishi PA10.

Figura 16.6. Selección del robot para la tarea de control visual.

A continuación, se elige la cámara que se situará en el extremo del robot seleccionado. Dado que en el enunciado del ejemplo se indican los parámetros intrínsecos y resolución de la cámara, se puede seleccionar en el entorno visual la creación de una nueva cámara con esos parámetros (ver Figura 16.7).

Figura 16.7. Definición de parámetros para cámara NI 1744.

El siguiente paso consiste en seleccionar el patrón que servirá como referencia para el controlador. A partir de este patrón se obtendrán las características visuales que el controlador necesita para obtener el error. En el entorno visual se puede utilizar un patrón de 4 puntos que forma un cuadrado de 0.5m de lado centrado en la posición (1,0,-0.5) m del sistema de referencia de la base del robot. Este es el patrón por defecto. También se puede definir un nuevo patrón formado por una malla de N puntos, indicando la separación entre puntos y la translación y orientación del sistema de referencia del patrón respecto del sistema de referencia de la base del robot, Figura 16.8.

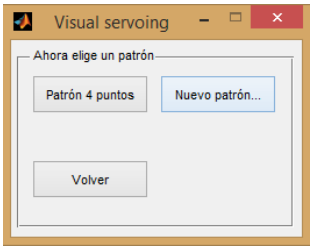


Figura 16.8. Selección del objeto para control visual.

Una vez se han definido los parámetros básicos de la tarea de control (el robot, la cámara y el objeto o patrón), es el momento de definir la posición deseada del robot y exportar dicha posición utilizando el botón "Posición deseada" (Figura 16.9). Para ello, se puede proceder de dos formas distintas. El robot se puede posicionar a través de sus coordenadas articulares, o con movimientos relativos cartesianos del extremo. El movimiento articular se refiere a los valores en grados o radianes de cada una de las articulaciones que componen el robot. También es posible importar una variable del Workspace de MATLAB que describa una determinada posición del robot utilizando el botón "Cargar posición" del panel central "Opciones posiciones". Si durante el transcurso de esta operación se desea devolver al robot a la posición articular inicial, bastará con hacer clic en la opción "Posición home". El movimiento cartesiano relativo permite movimientos de translación relativos del extremo en cualquier eje del sistema de referencia de la base.

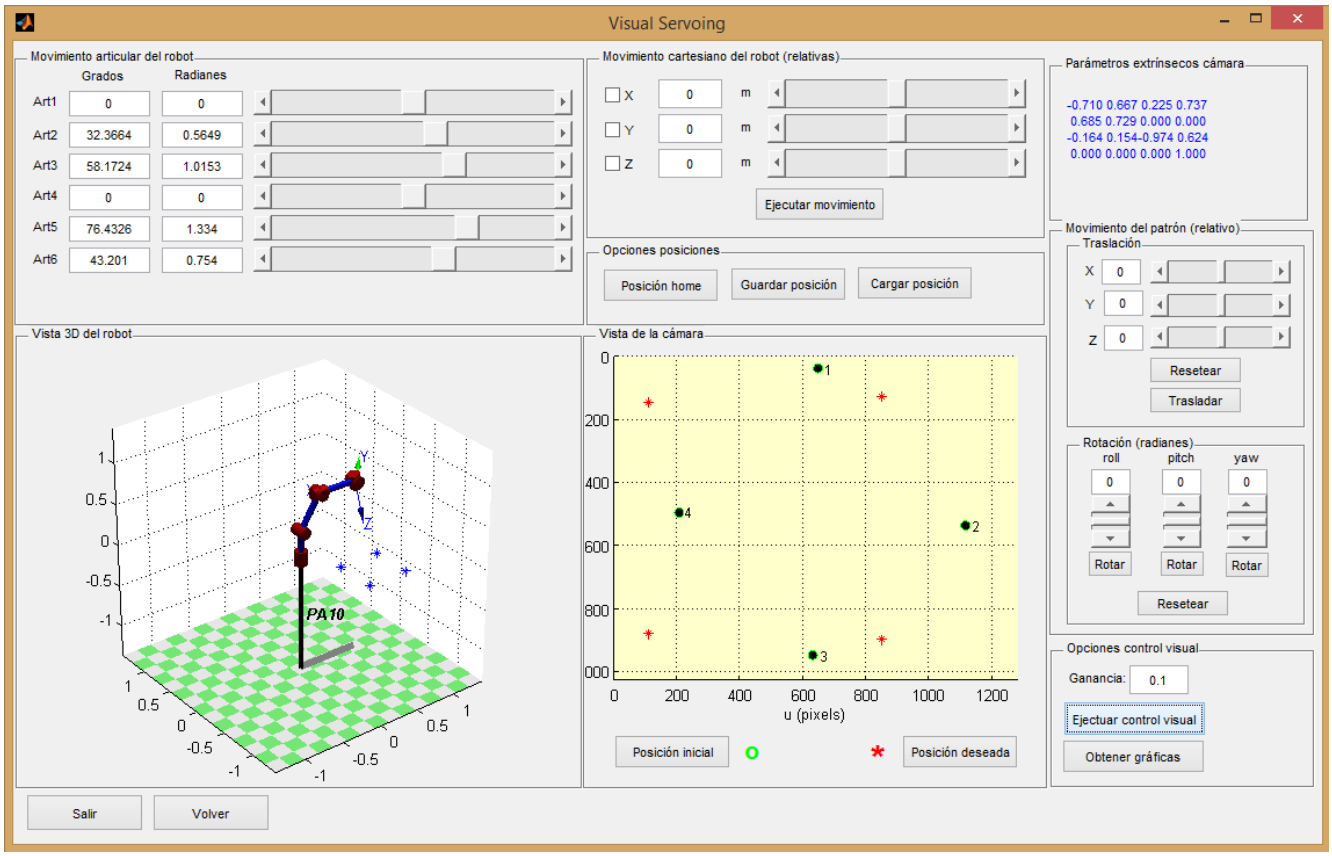


Figura 16.9. Herramienta de simulación de control visual.

La vista de la cámara situada en el extremo del robot permite observar dónde se localizan las características visuales en el plano imagen. Al pulsar sobre el botón "Posición deseada" se almacena la posición de cada característica en el plano imagen con la posición actual del robot y se quedará marcado con un asterisco en rojo. Ésta será la referencia del controlador. A continuación se ha de situar el extremo del robot en una posición distinta que se denominará como "Posición inicial", que quedará entonces marcada con círculos verdes. El controlador aplicará diferentes valores de velocidad del extremo del robot para desplazar la cámara desde esta posición inicial hasta la posición deseada en imagen que se estableció previamente. De esta forma queda totalmente definida la tarea de control visual.

Hasta este momento se han preparado los elementos básicos que permiten realizar la simulación de control visual del robot Mitsubishi PA-10 con la cámara NI1744. Ahora es el momento de ejecutar el controlador visual basado en imagen. Para ello, basta con hacer clic en el botón "Ejecutar control visual". Así, se ejecuta la tarea de posicionamiento mediante control visual basado en imagen. Aparecerá una ventana que permite visualizar cómo varía lo que está capturando la cámara acoplada al robot durante su desplazamiento. En otra ventana se puede observar el movimiento de la cámara en el espacio tridimensional, Figura 16.10.

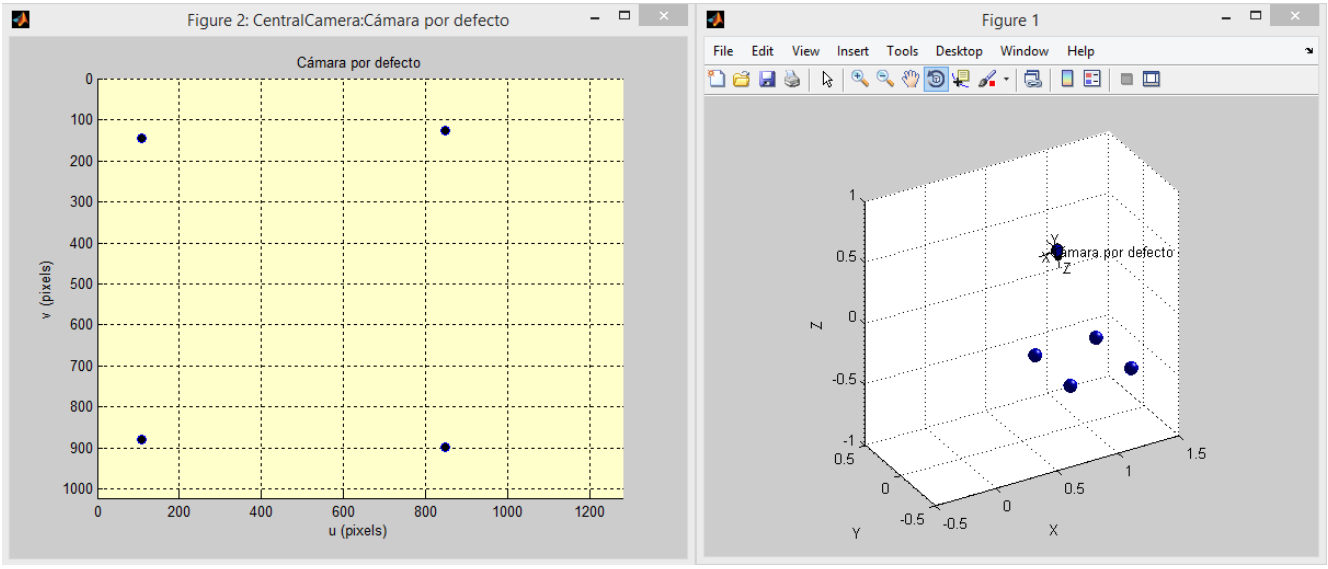


Figura 16.10. Ventanas que permiten observar la evolución de las características en imagen (izquierda) y la evolución de la cámara en el espacio Cartesiano 3-D durante la ejecución de la tarea de control visual (derecha).

En caso de generarse un error durante la simulación, quiere decir que el robot no ha podido alcanzar su objetivo por algún motivo. En estos casos, será necesario revisar los parámetros definidos para garantizar que el robot no pasa por algún tipo de configuración singular durante la trayectoria.

Una vez concluida la simulación, el usuario podrá obtener una serie de gráficas que describen el funcionamiento del controlador. Para ello, el entorno proporciona un botón para generar las gráficas. Se podrán obtener gráficas de la evolución de la velocidad enviada por el controlador, es decir, la velocidad del extremo del robot. También se puede obtener una gráfica de la evolución del error de las características visuales. Una gráfica muy interesante para comprobar el correcto funcionamiento del controlador durante la tarea de posicionamiento es la que permite ver la evolución de las características visuales en el plano imagen. Al definir la tarea de control visual con un patrón de cuatro puntos, esta gráfica mostrará la evolución en imagen de cada uno de esos cuatro puntos. Por último, también es posible obtener una gráfica que muestra la evolución de la cámara situada en el extremo del robot durante la tarea de control visual, Figura 16.11.

Estas gráficas permiten comprobar el correcto funcionamiento del controlador. Al tratarse de un controlador proporcional, el error del controlador debe presentar un decrecimiento exponencial, como se puede ver en la Figura 16.11. Además, al ser un controlador basado en imagen, la evolución de las características visuales en el plano imagen siguen una línea recta.

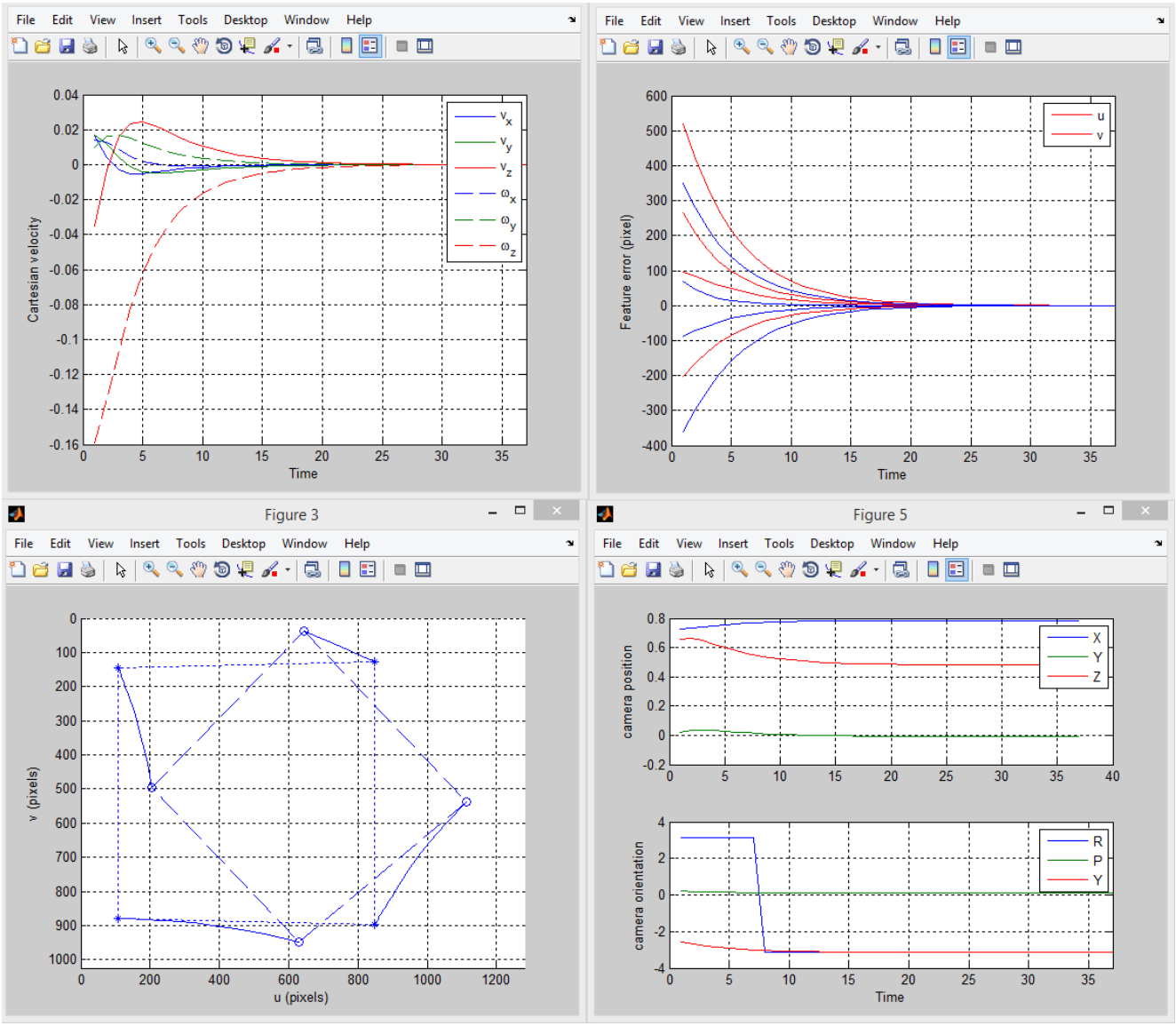


Figura 16.11. Gráficas obtenidas a través del interfaz visual de simulación de control visual basado en imagen: velocidad de la cámara (arriba izquierda); error en imagen (arriba derecha); evolución de las características en el plano imagen (abajo izquierda); y posición en el espacio Cartesiano 3-D de la cámara (abajo derecha).

16.4. Aplicaciones

Actualmente, es posible enumerar un gran número de aplicaciones de los sistemas de control visual dentro del ámbito industrial. Cabe mencionar, por ejemplo, tareas como el ensamblado/desensamblado, mecanizado automatizado, inspección de placas de circuito impresos, soldadura, manejo de materiales peligrosos, guiado de robots en el espacio, vigilancia, cirugía a distancia, industria textil, minería, etc. Aunque es posible encontrar un gran número de trabajos teóricos, investigaciones y aplicaciones industriales dentro del ámbito de los sistemas de control visual, no son tantos los grupos de investigación que han desarrollado sistemas de control visual en aplicaciones de interacción robot-humano. Cabe destacar, por ejemplo, el trabajo expuesto en Tsui y col., (2011) para la ayuda de personas discapacitadas. En este caso, el usuario selecciona un objeto y el robot lo alcanza de forma

autónoma para acercárselo al discapacitado. En este tipo de aplicaciones las principales investigaciones van desde la interfaz robot-humano hasta el sistema de control del robot. Relacionados con este último punto se vienen desarrollando en los últimos años '*trackers*' visuales para escenas reales, capaces de adaptarse a las condiciones de funcionamiento de los sistemas de control visual. Como se describirá a lo largo de este apartado, los sistemas de control visual presentan aplicaciones potenciales en áreas muy diversas: en robótica del espacio, para la aplicación de manipuladores en tareas de mantenimiento de satélites; en robótica de servicio, para el diagnóstico e intervención empleando imágenes de ultrasonidos; en robótica aérea, para el seguimiento y control de posición, etc. Con el objetivo de realizar la aplicación práctica de este tipo de sistemas, se vienen diseñando sistemas de control visual semi-autónomos donde el robot y el operador humano colaboran conjuntamente para alcanzar un determinado objetivo. El principio de esta aproximación consiste en que el robot se encarga de realizar el posicionamiento preciso mientras que el humano se encarga de inicializar, especificar y monitorizar la tarea de control visual.

A lo largo de los últimos años, se ha desarrollado una gran cantidad de investigación relacionada con los sistemas de control visual y todavía es un área activa de investigación con muchos problemas abiertos todavía a resolver (Chaumette y Hutchinson, 2006; Chaumette y Hutchinson, 2007; Chesi y Hashimoto, 2010). La principal aplicación del control visual es el posicionamiento de precisión de un brazo robótico dada una ubicación deseada. Sin embargo, esta función simple es crucial en muchas tareas de nivel superior. Una de estas tareas de alto nivel es realizar el agarre y/o manipulación de un objeto. En un entorno industrial, donde el ambiente de trabajo es conocido y adaptado al robot, agarrar un objeto puede llegar a ser bastante trivial. En este tipo de aplicaciones la localización del objeto y la mano robótica son conocidos de antemano y se obtienen buenas precisiones durante el agarre (de hecho, en general el uso de visión artificial en este tipo de aplicaciones no está justificado). Sin embargo, supóngase el uso de un robot humanoide en un entorno no estructurado como pueda ser una cocina en la que debe realizar el agarre de una botella. Este escenario presenta una gran cantidad de problemas que pueden solucionarse con la utilización del bucle cerrado y realimentación visual. En esta aplicación sería necesario utilizar visión artificial para detectar el objeto a agarrar, mover el robot a una localización cercana a la botella, localizar la cámara para que tanto la mano como la botella se encuentren en el campo de visión de la cámara y, por último, utilizar control visual para posicionar correctamente la mano y posteriormente agarrar y manipular la botella.

Dentro del ámbito del guiado de vehículos autónomos también es posible encontrar un gran número de aplicaciones de los sistemas de control visual. En este caso, la realimentación visual es utilizada para llevar a cabo el guiado del vehículo empleando referencias visuales. La navegación viene expresada como un conjunto de características a ser observadas en cada momento y no localizaciones tridimensionales a alcanzar. Para ello se emplea como estrategia básica los sistemas de control basados en imagen con el objetivo de ir alcanzando las características visuales deseadas y, de esta manera, conseguir el guiado. No sólo es posible encontrar aplicaciones para el guiado de robots móviles con ruedas o brazos robots, sino que comienzan a surgir en los últimos años los sistemas de control visual

para el guiado de robots submarinos y aéreos. Relacionados con los primeros existen aplicaciones para el posicionamiento de los mismos respecto a estructuras submarinas o tuberías. Relacionado con los segundos, en los últimos años han surgido distintas aplicaciones dentro del ámbito del guiado de los denominados drones, así como los sistemas de control visual para el guiado y localización de aviones. Así, cabe mencionar aplicaciones de control visual para ayuda durante el aterrizaje o repostaje en vuelo. En algunas ocasiones, para conseguir la localización de aviones se emplea una técnica denominada control visual virtual. El principio de funcionamiento de los sistemas de control visual virtual es modificar la posición de una cámara virtual de forma iterativa con el fin de alcanzar las características deseadas extraídas por una cámara real. Se emplea una ley de control basada en imagen para ir modificando la localización de la cámara virtual. Las características extraídas por dicha cámara son obtenidas por proyección conocido el modelo del objeto observado y los parámetros intrínsecos y extrínsecos de la cámara virtual. Cuando el error se haya anulado los parámetros extrínsecos de la cámara virtual coincidirán con la localización tridimensional del objeto observado. Más información acerca de estas técnicas puede obtenerse en Marchand y Chaumette (2002).

Uno de los ámbitos en los que más han crecido las aplicaciones de los sistemas de control visual en los últimos años es el de la robótica médica y, más en concreto, en el campo de la cirugía mínimamente invasiva. En este tipo de cirugía los dispositivos robóticos se insertan en el cuerpo humano o son utilizados como herramientas quirúrgicas. Estos dispositivos robóticos requieren de un método de posicionamiento y guiado preciso. Esto último es crítico cuando una herramienta robótica es insertada dentro del cuerpo humano y más aún cuando se emplean herramientas flexibles como catéteres, agujas o endoscopios. Como es bien conocido, las imágenes médicas son empleadas por los médicos para obtener información tanto antes como durante la cirugía. De hecho, el procesamiento de estas imágenes es una técnica eficiente para proporcionar una realimentación rápida y precisa de la posición que puede emplearse para el control de robots manipuladores. Esta información es empleada por los algoritmos de control visual usados en robótica médica. En estos casos, la realimentación visual a menudo se obtiene usando cámaras habituales o endoscópicas. También existen aplicaciones que emplean otro tipo de información visual útil en determinados ámbitos para detectar ciertas características que no pueden observarse haciendo uso de información visual convencional. Así, cabe citar el uso de imágenes tomográficas, ultrasonidos, rayos X, resonancia magnética, etc. Un estudio acerca de las aplicaciones médicas de los sistemas de control visual puede consultarse en Azizian y col., 2014 y Azizian y col., 2015).

16.5. Bibliografía

Azizian, M.; Khoshnam, M.; Najmaei, N.; Patel. R. V. (2014). “Visual servoing in medical robotics: a survey. Part I: endoscopic and direct vision imaging - techniques and applications”. Int J Med Robot. Vo. 10, no. 3, pp. 263-74. 2014

Azizian M.; Najmaei N.; Khoshnam M.; Patel R. (2015) Visual servoing in medical robotics: a survey. Part II: tomographic imaging modalities--techniques and applications. *Int J Med Robot*, vol. 11, no. 1, pp. 67-79.

Chaumette, F.; Hutchinson, S. (2006). Visual Servo Control, Part I: Basic Approaches. *IEEE Robotics and Automation Magazine*, *13*(4), 82-90.

Chaumette, F.; Hutchinson, S. (2007). Visual servo control, Part II: Advanced approaches. *IEEE Robotics and* Automation Magazine, 14(1), 109-118.

Chesi, G.; Hashimoto, K. (2010). Visual Servoing via Advanced Numerical Methods. London, UK: Springer Link.

Hill, J.; Park, W. (1979). Real Time Control of a Robot with a Mobile Camera. En J. Wallace (eds.), Procedings of the 9th International Syposium on Industrial Robots. Washington DC, USA: Society of Manufacturing Engineers, 233-246

Marchand, E.; Chaumette. F. (2002) Virtual Visual Servoing: a framework for real-time augmented reality. En EUROGRAPHICS 2002 Conference Proceeding, G. Drettakis, H.-P. Seidel (eds.), Computer Graphics Forum, 21(3), Saarebrücken, Germany, September.

Perez, J. (2015). ViSeC_gui. http://www.aurova.ua.es/index.php/es/laboratorio-virtual/visec-matlab. Última vez visitado el 28 de septiembre de 2015.

Sanderson, A.; Weiss, L. (1980). Image-Based Visual Servo Control Using Relational Graph Error Signals. En Proceedings of the IEEE International Conference on Cybernetics and Society, Cambridge, USA: Institute of electrical and Electronic Engineers (IEEE), 1074-1077

Tsui, K. M.; Kim, D. J.; Behal, A.; Kontak, D.; Yanco, H. A. (2011). I want that: Human-in-the-loop control of a wheelchairmounted robotic arm, Journal of Applied Bionics and Biomechanics, 8(1), 127–147.

Torres, F.; Pomares, J.; Gil, P.; Puente, S. T.; Aracil, R. (2002). Robots y Sistemas Sensoriales. Editorial Pearson Educación.

CAPÍTULO 17

VISIÓN POR COMPUTADOR EN DISPOSITIVOS MÓVILES

Óscar DÉNIZ,[1,3] Gloria BUENO[1,3] y Jesús SALIDO[2,3]

[1] Depto. de Ing. Eléctrica, Electrónica, Automática y Comunicaciones (IEEAC)
ETSII, Universidad de Castilla-La Mancha (UCLM), Ciudad Real (España)
[2] Depto. IEEAC / Escuela Superior de Informática, UCLM, Ciudad Real, (España)
[3] Grupo de Visión y Sistemas Inteligentes (VISILAB)

En este capítulo se aporta al usuario la información esencial para realizar aplicaciones de visión por computador en las plataformas móviles Android y, en menor medida, iOS (Apple). Se explicará, para ambos casos, cómo usar la librería OpenCV de visión por computador. El capítulo comienza con una breve introducción a los dispositivos móviles en los que se resaltan las peculiaridades hardware y software de estos sistemas computacionales que condicionan el desarrollo de aplicaciones de visión en ellas, diferenciándolo de la implementación de las mismas aplicaciones sobre computadores de propósito general. El capítulo se completa con secciones específicas en las que se explican las herramientas específicas de desarrollo y su utilización para programar aplicaciones de visión tanto en el sistema operativo Android como iOS.

17.1. Introducción a la computación en dispositivos móviles

De acuerdo a cifras publicadas recientemente en los medios especializados, el volumen de ventas de *smartphones* (teléfonos inteligentes) durante el año 2014 ha crecido el 28 % respecto al año anterior llegando a una cifra de 1.200 millones de terminales vendidos en todo el mundo, muy por encima de los aproximadamente 230 millones de *tablets* (tabletas electrónicas) y los 308 millones de computadores personales (de escritorio y portátiles). Estas cifras aportan una idea de la presencia de los dispositivos móviles en nuestras vidas y en nuestros hábitos. En la actualidad tanto los *smartphones* como las *tablets* están equipados con una o dos cámaras que son responsables de aproximadamente ¡1 billón! de capturas durante el año 2014 (en parte debido a la popularidad de los *selfies*). Estos datos

proporcionan una idea aproximada de la importancia que tienen las técnicas de procesamiento de imagen y vídeo en los dispositivos móviles mencionados anteriormente.

En este capítulo, por dispositivo móvil se entiende aquellos sistemas de computación personal que permiten el procesamiento de imagen y vídeo en cualquier lugar y momento ya que pueden ser llevados por el propio usuario en sus desplazamientos cotidianos. De este modo, con esta denominación nos referiremos a los *smartphones* y los *tablets*.

La movilidad de los dispositivos afecta a tres aspectos de los mismos determinando sus posibilidades de aplicación:

1. **Comunicaciones** entre dispositivos a través de diferentes redes empleando distintas tecnologías inalámbricas y protocolos. Entre las tecnologías actualmente más empleadas están: 3G/4G, Wi-Fi, Bluetooth y NFC.
2. **Hardware**. Implica las características relacionadas con la unidad principal de procesamiento (CPU), los procesadores gráficos dedicados (GPU), la memoria, las interfaces de entrada salida (pantallas táctiles, micrófono y altavoces), los sistemas sensoriales (GPS, acelerómetros, giróscopo, campo magnético, proximidad, luminosidad, etc.), cámaras y batería.
3. **Software**. Determinado por el sistema operativo que emplea el dispositivo móvil (Android, iOS y otros).

Figura 17.1. Teléfono inteligente equipado con cámara

Las tecnologías relativas a las comunicaciones empleadas en los dispositivos móviles no se distinguen de las que aparecen en los sistemas de computación de propósito general. Sin embargo, tanto en el hardware como en el software de los dispositivos móviles aparecen diferencias respecto a los computadores de propósito general que condicionan el modo en que se implementan las aplicaciones de visión y procesamiento de imagen en estos dispositivos.

Debido a su relativa simplicidad y bajo consumo (del orden de 1 mW en estado de bajo consumo) en los dispositivos móviles es predominante el empleo de la arquitectura ARM. El empleo de esta arquitectura, muy diferente a la utilizada en los computadores de propósito general dificulta la portabilidad de aplicaciones entre estos sistemas y los dispositivos móviles. Mientras que en una arquitectura PC los diferentes circuitos integrados (entre los que se encuentra el microprocesador) se

agregan en una placa base para constituir el sistema computador, en un sistema de computación móvil se recurre a la tecnología SoC (*System on a Chip*).

La arquitectura SoC reúne en un único circuito integrado tanta funcionalidad como sea posible con el objetivo de reducir el espacio necesario y el consumo requerido para su funcionamiento. De este modo los procesadores basados en SoC integran las capacidades tanto computacionales (CPU) como gráficas (GPU). Por ejemplo, los procesadores Hummingbird de Samsung integran una CPU ARM Cortex A8 junto a un núcleo gráfico PowerVR SGX 535; en el caso de los procesadores Tegra 2 de NVIDIA se dispone de una CPU ARM Cortex con una GPU desarrollada por NVIDIA. La heterogeneidad de los procesadores SoC dificulta las comparaciones de desempeño entre ellos ya que se debe analizar conjuntamente tanto sus cualidades de cómputo como su capacidad gráfica.

Debido a la importancia que adquiere la autonomía en los dispositivos móviles, en ellos cobra especial importancia los aspectos relacionados con el almacenamiento de energía y su consumo. De este modo las baterías han ido mejorando su relación de energía por unidad de masa y reduciendo su tiempo de carga. Por este motivo es muy importante que al desarrollar aplicaciones de visión por computador en los dispositivos móviles se realicen análisis de rendimiento que permitan su optimización. En este sentido es importante considerar las técnicas de depuración y análisis de rendimiento (*debugging* y *profiling*) para que las aplicaciones finales se optimicen en cuanto a los aspectos de consumo, maximizando la autonomía de los dispositivos para unos recursos dados.

17.1.1. Hardware de captura de imagen y otros sistemas sensoriales

Los dispositivos móviles actuales están equipados con dos cámaras (frontal y trasera) para la captura de imagen y vídeo con dos propósitos diferenciados. La cámara frontal, con características menos exigentes, está destinada a tareas de vídeo conferencia y auto capturas (los populares *selfies*). La cámara trasera, con unas características más similares a las de una cámara fotográfica compacta dedicada, se destina a capturas de imagen y vídeo propiamente dichas. En los *smartphones* actuales incluso es posible la captura simultánea con ambas cámaras (captura dual).

Las cámaras de los dispositivos móviles están basadas en tecnología CMOS y abarcan un rango casi tan amplio como el de fabricantes de *smartphones*, llegando en modelos más recientes, como el Samsung S6, a resoluciones de 16 MP (5.312x2.988 píxeles) en la cámara principal y 5 MP (2.592x1.458 píxeles) en la frontal. En este mismo equipo la resolución de la captura de vídeo llega hasta UHD 4K 2160p a 30 fps en su cámara trasera y a 1440p a 30 fps en su cámara frontal.

Generalmente las hojas de características de las cámaras móviles suministradas por los fabricantes aportan poca información y hacen énfasis en las especificaciones que son más fácilmente asumidas por el consumidor final, a menudo erróneamente, como sinónimo de calidad. Algunas de las características menos publicitadas pero que más influencia tienen sobre la captura de imagen son:

- **Tamaño del sensor de imagen**: Es un valor que rara vez se suministra pero de gran importancia en el desempeño de la cámara ya que para la misma resolución, cuanto mayor es el tamaño del sensor, mejores son el rango dinámico y la relación señal/ruido de las

capturas. En la actualidad, el valor del tamaño del sensor para este tipo de cámaras están en el rango 1/3" y llegan a 1/2,3" en los sensores de altas prestaciones (Sony). Comparando el tamaño del sensor de fotograma completo (*full frame*) de 36 x 24 mm, este ofrece una superficie de captura de luz de 864 mm^2 mientras que en una cámara con un sensor de 1/2,3" de 5,76 x 4,29 mm la superficie expuesta a la luz es de 25 mm^2.

- **Óptica y enfoque**. La calidad de la óptica de estas cámaras ha mejorado mucho en los últimos años. Se trata siempre de ópticas fijas (sin zoom), con distancia focal fija en torno a los 25 mm, pero en las que se incorpora el autoenfoque y la estabilización óptica de la imagen (OIS) en los modelos más recientes (Samsung S6). Tanto la velocidad del autoenfoque como la OIS ha mejorado la calidad de las capturas de sujetos en movimiento y en condiciones de baja luminosidad con tiempos de exposición reducidos.
- **Apertura**. En las cámaras móviles, la apertura es fija y la tendencia es emplear aperturas grandes (*f* pequeños, p. ej. *f* 1,9 en el Samsung S6) persiguiendo mejorar la sensibilidad en condiciones de baja luminosidad.

Estas cámaras están equipadas con sistemas sensoriales (acelerómetros) y procesadores dedicados que proporcionan capacidad de estabilización óptica de imagen, autoenfoque, auto HDR y detección de rostros entre otras. Las técnicas HDR (*High Dynamic Range*) permiten obtener una captura con distintos niveles de exposición (al menos dos) para conseguir el valor más apropiado dependiendo de la cantidad de luz en cada zona de la imagen, de este modo se obtienen colores más brillantes y un rango dinámico más amplio de la imagen.

El hecho de que los dispositivos móviles estén equipados con sistemas sensoriales adicionales (GPS, acelerómetros, giróscopo, campo magnético, proximidad, luminosidad, etc.) que habitualmente no están presentes en los computadores de propósito general añade posibilidades adicionales a las aplicaciones de usuario final en las que también se realiza procesamiento de imagen o vídeo.

17.1.2. Los sistemas operativos de los dispositivos móviles

Según se observa en la Fig. 17.2 el sistema operativo (SO) dominante en el mercado de los dispositivos móviles es Android seguido por iOS a bastante distancia. Las diferencias entre los SO móviles y los predominantes en los computadores de propósito general (Windows, Mac OSX y GNU/Linux) son la causa de que una aplicación desarrollada en estos últimos no sea compatible en un dispositivo móvil. Por el momento, el desarrollo de aplicaciones, incluyendo las de visión por computador, en los SO móviles es más complejo que en los SO más «tradicionales».

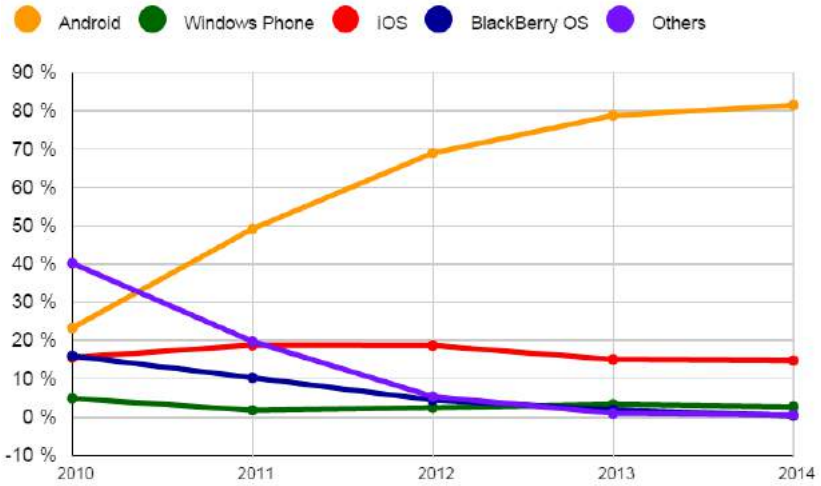


Figura 17.2. Cuota de mercado mundial de los *smartphones* por sistema operativo (fuente: IDC)

iOS es un SO desarrollado por Apple Inc. en 2007 que actualmente está en su versión 8.3. Este SO gobierna los iPhone, así como las *tablets* iPad. A diferencia del SO Android y Windows Phone, Apple no licencia el uso de iOS para hardware de otras compañías. iOS deriva de Mac OS X cuyos componentes básicos proceden de Darwin, un SO tipo Unix de código abierto conforme al estándar POSIX liberado por Apple Inc. en el año 2000.

Google adquirió en el año 2005 la compañía de software Android Inc. embarcada en el proyecto de desarrollo del SO Android. En 2007 Google liberó bajo licencia Apache el código de Android, un SO para dispositivos móviles, basado en Linux, para que todo aquel fabricante de hardware que lo deseara pudiera emplearlo en sus productos. Ese mismo año se constituyó la Open Handset Alliance (OHA), un consorcio de más de 80 empresas, liderado por Google, dedicado al impulso de estándares abiertos para dispositivos móviles. Desde su inicio la OHA ha abanderado el proyecto de desarrollo e implantación del SO móvil abierto Android.

17.1.3. Aplicaciones de visión en dispositivos móviles

En general, el objetivo de las aplicaciones de visión por computador es capturar imágenes o vídeo y procesarlos mediante software que obtenga automáticamente información a partir de ellas, Figura 17.3. En los dispositivos móviles las aplicaciones de visión adquieren una nueva dimensión ya que además de poseer cámaras capaces de capturar la imagen/vídeo poseen la movilidad inherente a estos dispositivos. Las aplicaciones que pueden desarrollarse abarcan el reconocimiento de rostros, objetos, edificios, aplicaciones médicas, de asistencia, etc. El reto en estos casos es la programación de aplicaciones fiables, capaces de ejecutarse en tiempo real.

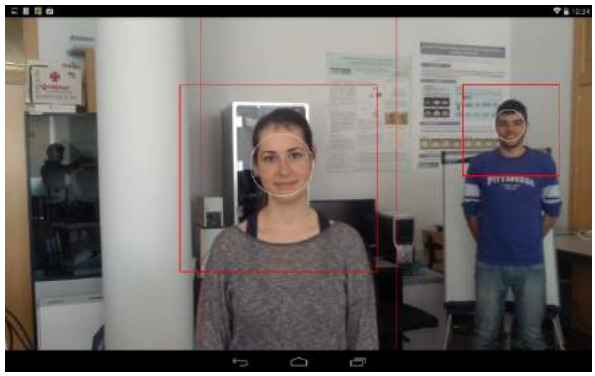


Figura 17.3. Salida gráfica de aplicación móvil de detección y posición de personas en escena (fuente: Grupo Visilab-Uclm)

Existen dos modelos de computación aplicables en el desarrollo de aplicaciones móviles de visión por computador:

1. **Modelo distribuido cliente-servidor**. En este modelo el dispositivo móvil (cliente) captura la imagen y, mediante conexión inalámbrica a una red de datos, transfiere dicha imagen a un servidor que procesa la imagen (puede incluir también la consulta a un servicio de base de datos) y devuelve los resultados al cliente. Éste fue el modelo elegido en las primeras aplicaciones de visión por computador en dispositivos móviles cuando su capacidad de cómputo era muy limitada. En la actualidad se recurre a este modelo cuando, por ejemplo, la aplicación cliente precisa la consulta a un servicio de base de datos. Bajo este modelo las aplicaciones no requieren requisitos de respuesta en tiempo real.
2. **Modelo de procesamiento local**. Con el aumento de la capacidad de cómputo de los dispositivos, las aplicaciones de visión por computador llevan a cabo el procesamiento de imagen/vídeo de modo local, ofreciendo en muchos casos características de respuesta de tiempo real. No obstante algunas aplicaciones pueden requerir la conexión a un servidor remoto cuando la capacidad del servidor no puede ser implementada de modo local (p. ej. consulta a grandes bases de datos).

17.2. Herramientas de desarrollo en iOS

Para desarrollar aplicaciones de visión por computador en iOS sólo es necesario tener instalado XCode, el entorno de desarrollo gratuito de Apple (podríamos decir que es el equivalente Apple a Microsoft Visual Studio). XCode puede compilar Objective-C, C, C++ y Java. Con él podemos desarrollar software para ordenadores Mac, teléfonos iPhone, reproductores iPod y tabletas iPad. XCode es un completo entorno de desarrollo que incluye editor de código, diseñador de interfaces de usuario, depurador y simulador iOS. De forma similar a otros entornos de desarrollo, en XCode se trabaja con proyectos. Un proyecto contiene toda la información de nuestra aplicación, incluyendo ficheros de código fuente, diseños de interfaces de usuario (las pantallas de nuestra aplicación) y opciones de compilación.

Las capacidades y opciones de XCode quedan fuera del ámbito de este capítulo. En cualquier momento podemos consultar la completísima ayuda del programa, que incluye manuales de referencia

del SDK iOS. En cualquier caso, el proceso básico que seguiremos con XCode para crear nuestras aplicaciones pasará por los siguientes pasos:

1. Editar ficheros de código fuente,
2. Diseñar la interfaz de usuario,
3. Ejecutar la aplicación (en el emulador o en un dispositivo, esto último obligatorio si usamos la cámara), y
4. Distribuir la aplicación.

Para poder ejecutar la aplicación en un dispositivo es necesario realizar una serie de pasos que habilitan en el dispositivo lo que se conoce como 'perfil de aprovisionamiento'. Este proceso, que requiere adquirir una suscripción al programa de desarrolladores iOS de Apple (la suscripción individual por un anual cuesta 99€ en 2015) y conectar físicamente el dispositivo al ordenador Mac de desarrollo, se explica con detalle en muchos textos como Déniz y col. (2013) y en recursos en Internet, por lo que lo omitiremos aquí.

Para incorporar OpenCV a nuestros proyectos iOS, básicamente tenemos dos posibilidades. Para los más avezados siempre está la opción de recompilar la librería para iOS. Es un proceso largo y propenso a errores, pero tiene la ventaja de poder disponer de las funciones más recientes incorporadas en las últimas versiones de OpenCV. Sin embargo, en la mayoría de los casos es suficiente con incorporar a nuestro proyecto la librería ya precompilada. Actualmente desde la web de OpenCV se puede descargar el 'framework' de OpenCV 3. En XCode un framework o marco de trabajo es prácticamente equivalente a una librería. Una de las mayores ventajas de los marcos de trabajo es que podemos incorporarlos a nuestros proyectos fácilmente, basta con arrastrar la carpeta del marco de trabajo descargado (opencv2.framework) sobre el grupo Frameworks dentro del Navegador de Proyecto de XCode. En ese momento se nos da la posibilidad de copiar los ficheros del marco de trabajo a la carpeta destino del proyecto. Si estamos compartiendo el marco de trabajo entre múltiples proyectos tal vez queramos dejar esa opción desactivada.

Una vez se ha añadido el marco de trabajo, nuestro proyecto enlazará automáticamente con la librerías de OpenCV precompiladas. También están desde ese momento disponibles los ficheros cabecera, que debemos incluir en nuestros ficheros fuente (deben referenciarse con ruta relativa al marco de trabajo). Para poder usar el marco de trabajo OpenCV en nuestro proyecto es necesario añadir otros marcos de trabajo de Apple. Los marcos de trabajo necesarios son: AVFoundation.framework, ImageIO.framework y libz.dylib. Si queremos hacer captura de vídeo deberemos también añadir: CoreVideo.framework y CoreMedia.framework.

Como se mencionó anteriormente, para utilizar OpenCV en nuestros ficheros de código fuente hemos de incluir las cabeceras correspondientes (podemos ver los ficheros cabecera desplegando la entrada de OpenCV dentro del grupo Frameworks del Navegador de Proyecto). La forma más sencilla de hacer esto es editar el fichero X-Prefix.pch, donde X es el nombre que hemos dado a nuestro proyecto. Este fichero se encuentra en el grupo *Supporting Files* del Navegador de Proyecto. Debemos añadir tres líneas al principio del fichero (esto último es importante), como se muestra en la siguiente figura.

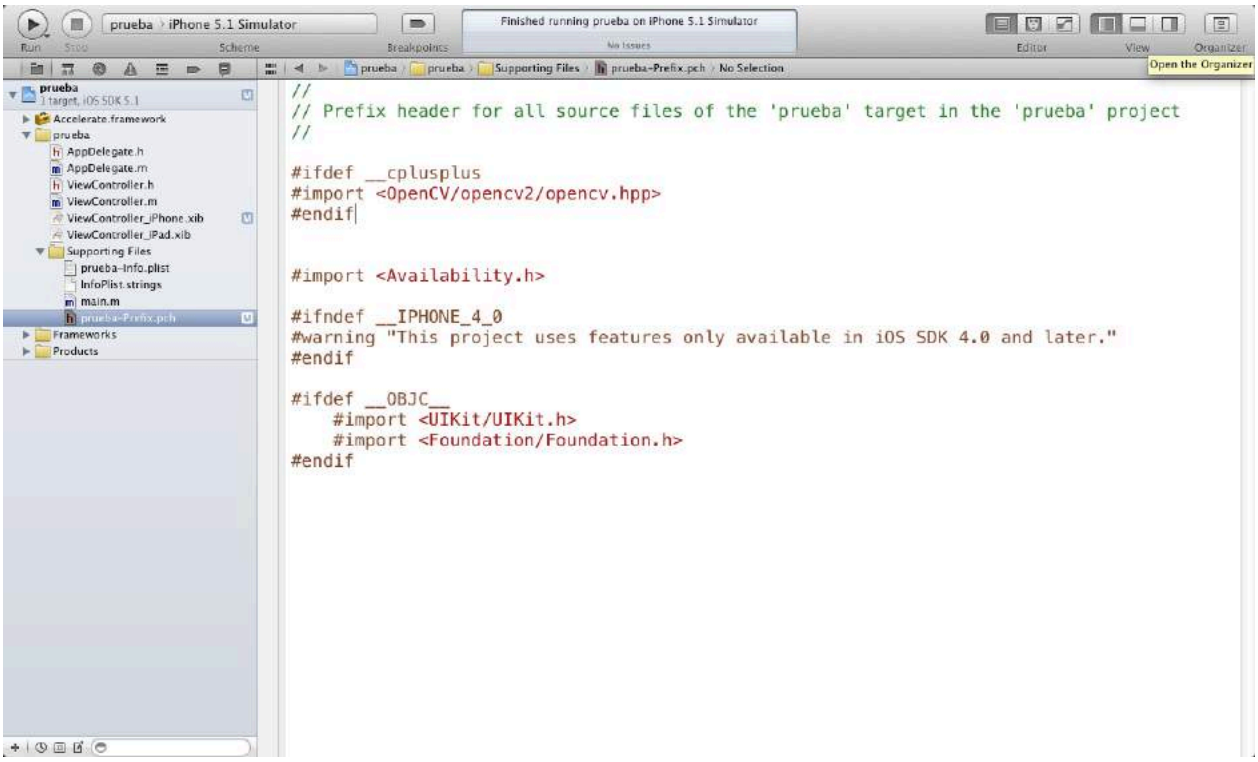


Figura 17.4. Inclusión de las cabeceras de OpenCV en nuestro proyecto

Los proyectos por defecto en XCode usan el lenguaje Objective-C. Los ficheros de código fuente en este lenguaje tienen la extensión .m. En los ficheros de código en que usemos OpenCV hemos de cambiar la extensión a .mm (basta con hacer doble clic sobre el fichero en el Navegador de Proyecto), que indica al compilador que el fichero fuente mezcla código Objective-C y C++.

En OpenCV se usa el tipo IplImage para contener una imagen (en versiones de OpenCV posteriores a la 2 se puede usar el tipo Mat). Hay que tener en cuenta, sin embargo, que en iOS se utiliza, dentro del marco de trabajo UIKit (el marco básico para crear y manejar una interfaz de usuario en la aplicación), la clase UIImage para mostrar imágenes. Se hace por tanto necesario disponer de funciones de conversión. La idea es que podamos trabajar con las funciones de OpenCV habitualmente y solo usemos las funciones de conversión cuando sea necesario. Recomendamos usar las funciones de conversión escritas por A. Myagkov, (2012). Este código puede ser usado libremente en aplicaciones tanto académicas como comerciales. Las funciones están implementadas en los ficheros CVImageConverter.h y CVImageConverter.mm. Para usar estas funciones basta con añadir ambos ficheros a nuestro proyecto e incluir CVImageConverter.h donde sea necesario con la directiva #import. Si lo hacemos arrastrando los ficheros al Navegador de Proyecto, hemos de asegurarnos de activar la opción Add to Targets. Por otro lado, es posible que estos ficheros den error de compilación si nuestro proyecto ha sido creado con la opción de conteo automático de referencias activada (el conteo automático de referencias, Automatic Reference Counting (ARC), se introdujo en la versión 4.2 de XCode con el fin de facilitar el manejo de memoria al programador. En este texto se asume que se usa ARC). Para evitarlo podemos desactivar el ARC en el fichero CVImageConverter.mm, editando las opciones de compilación del fichero para añadir la opción -fno-objc-arc (podemos acceder a las opciones del compilador en la pestaña Build Phases de las propiedades de proyecto, lista Compile

sources, columna Compiler Flags). En el siguiente apartado se mostrará un ejemplo que usa estas funciones de conversión.

17.3. Visión por computador en iOS, primeros pasos

En este apartado daremos un sencillo ejemplo que utiliza OpenCV para detectar caras en una imagen .jpg. Puede ser ejecutado tanto en el emulador como en un dispositivo (luego no es necesario tener un iPhone o iPad para ejecutarlo).

Lo primero que debemos hacer es crear un proyecto de tipo Single View y añadirle las funcionalidades de OpenCV (ver apartado anterior). Tras esto, hemos de añadir el fichero de la imagen al proyecto. Para ello crearemos un grupo *Imagenes* en el Navegador de Proyecto y arrastraremos la imagen dentro de la sección. En un proyecto de tipo Single View la aplicación carga inicialmente una única vista vacía que ocupa toda la pantalla del dispositivo. La vista es un objeto de la clase UIView (pertenece al framework UIKit). El método viewDidLoad de este objeto, que se ejecuta cuando la vista se ha cargado, nos servirá para cargar la imagen, intentar detectar la cara y dibujar el recuadro en su caso.

Insertaremos en la vista un nuevo objeto UIImageView, ocupando toda la pantalla del dispositivo (UIImageView es una clase hija de UIView que sirve para mostrar una imagen o secuencia animada de imágenes). Para que podamos referirnos a esta nueva vista en nuestro código hemos de declararla como una nueva variable miembro de la clase (un *referencing outlet*, en terminología de Objective-C). Otra posibilidad que no requiere crear un *referencing outlet* es crear el objeto UIImageView directamente dentro del método viewDidLoad.

Siguiendo esta opción, todo nuestro ejemplo quedaría dentro del método viewDidLoad:

```
- (void) viewDidLoad
{
   [super viewDidLoad];

   // Carga una imagen y la inserta en la vista actual...
   UIImageView *myView = [[UIImageView alloc] initWithFrame:self.view.frame];
   UIImage *testImage = [UIImage imageNamed:@"Yo.jpg"];
   myView.image = testImage;
   [self.view addSubview:myView];

   // Carga la cascada Haar de los recursos del proyecto...
   cv::CascadeClassifier _faceCascade;
   NSString *faceCascadePath = [[NSBundle mainBundle]
         pathForResource:@"haarcascade_frontalface_alt2" ofType:@"xml"];
   if (!_faceCascade.load([faceCascadePath cStringUsingEncoding:NSASCIIStringEncoding])) {
            NSLog(@"No pude cargar la Haar cascade: %@", faceCascadePath);
   }
```

```
    // Detectar caras...
    std::vector<cv::Rect> faces;
    cv::Mat mat;
    [CVImageConverter CVMat:mat FromUIImage:testImage error:NULL];
    _faceCascade.detectMultiScale(mat, faces, 1.1, 2, CV_HAAR_FIND_BIGGEST_OBJECT |
CV_HAAR_DO_ROUGH_SEARCH, cv::Size(60, 60));

    // Dibujar resultados...
    if (faces.size())
    {
      // recuadrar la cara detectada...
      cv::rectangle(mat, cv::Point(faces[0].x, faces[0].y),
cv::Point(faces[0].x+faces[0].width,faces[0].y+faces[0].height), CV_RGB(255, 0, 0), 2);
    }
    else {
      // escribir texto 'no face'
      int fontFace = cv::FONT_HERSHEY_SIMPLEX;
      double fontScale = 12;
      cv::putText(mat, "no face", cv::Point(10, mat.rows/2), fontFace, fontScale, CV_RGB(255, 0,
0), 20);
    }

    // Mostrar imagen resultado...
    myView.image = [CVImageConverter UIImageFromCVMat:mat error:NULL];
  }
```

Primero se carga la imagen y se añade a la vista actual, siguiendo los procedimientos habituales en iOS. Seguidamente se declara el objeto cascada de clasificadores y se carga con el detector de caras frontal (que está en el fichero haarcascade_frontalface_alt2.xml que se distribuye con OpenCV). La detección en sí se intenta en el método detectMultiScale. Las caras detectadas son devueltas en el vector STL faces. Las opciones que se han puesto en la llamada a detectMultiScale hacen que solo se devuelva la cara más grande encontrada. Finalmente se muestra los resultados de la detección. Si se detectó alguna cara se dibuja un recuadro sobre la propia imagen. Si no se detectó, se escribe el texto 'no face' en la imagen. Nótese que las funciones de conversión entre UIImage y Mat realmente hacen una copia de la imagen, por lo que es conveniente restringir su uso al mínimo. En este ejemplo hemos hecho una conversión al principio (la imagen de entrada) y otra al final para mostrar la imagen resultante. La Figura 17.5 muestra el resultado del programa cuando se ejecuta en el simulador.

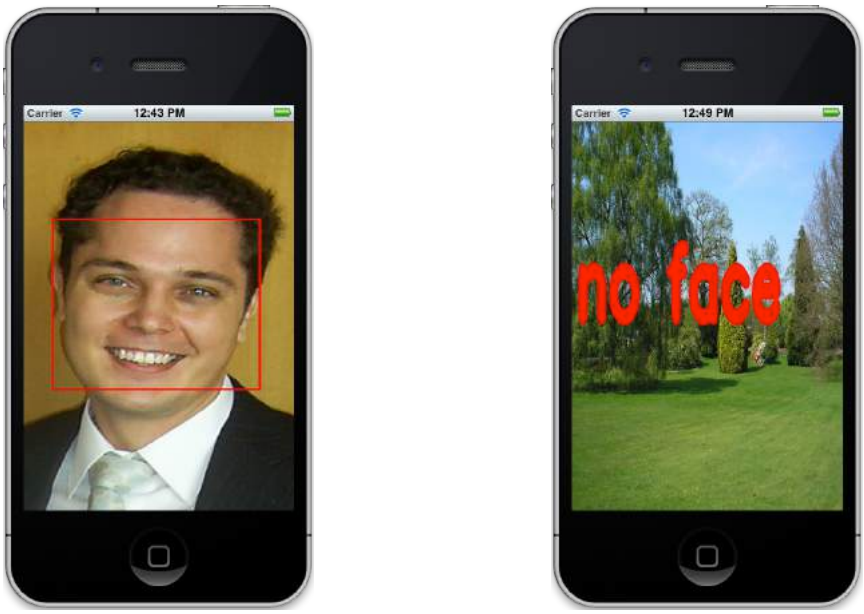


Figura 17.5. Resultado del ejemplo ejecutándose con una imagen con una cara (izquierda) y sin caras (derecha)

17.4. Herramientas de desarrollo Android

Para el desarrollo de aplicaciones Android, Google proporciona gratuitamente un kit de desarrollo (SDK) compuesto de varias herramientas para diversos propósitos en el flujo de trabajo del desarrollo de aplicaciones con Android (Google, 2015). Estas herramientas se pueden obtener en varias modalidades:

- *Plugin* ADT (*Android Developer Tools*) para IDE Eclipse. Este *plugin* proporciona un acceso sencillo y rápido a las herramientas necesarias para desarrollar aplicaciones Android. Aunque se desaconseja oficialmente, ya que su soporte cesará en favor del Android Studio, se recomienda su uso inicial ya que de este modo es más sencillo trabajar con los tutoriales de ejemplo de aplicaciones de visión por computador contenidos en la distribución estable de OpenCV para Android.
- *Android Studio. Actualmente es el IDE oficial propuesto para el desarrollo de aplicaciones móviles en las plataformas Android. Está basado en el IDE Java Intellij IDEA de JetBRAINS. Este entorno emplea un sistema de construcción de aplicaciones basado en Gradle que automatiza los pasos para construir, gestionar las dependencias y facilitar el despliegue de proyectos software, aplicado en este caso a los proyectos Android.*
- Modo independiente (*standalone*). Las herramientas se pueden emplear con el entorno de desarrollo (IDE) preferido (p. ej. Intellij IDEA, Eclipse, NVIDIA TADP, etc.).

Si bien las aplicaciones Android están basadas en el lenguaje Java (Eckel, 2006) hay que tener presente que Android emplea la máquina virtual Dalvik (DVM), sustituida por ART (Android Runtime) en la última versión de Android (Lollipop), que ejecuta un *bytecode* diferente al convencional de Java.

La construcción de una aplicación Android se puede resumir en los pasos siguientes:

1. Compilación de los recursos e interfaces de la aplicación para su uso en el proyecto.
2. Compilación de la aplicación junto a los recursos y las interfaces para crear las clases que componen el proyecto.
3. Conversión de las clases y librerías externas en un fichero *.dex de *bytecode* para la máquina virtual Dalvik.
4. Empaquetado, con la herramienta apkbuilder, del *bytecode* junto a los recursos no compilables (p. ej. imágenes) para obtener el fichero *.apk de la aplicación.
5. Finalmente el fichero *.apk debe ser firmado y alineado quedando listo para su instalación en un dispositivo.

Todos los pasos del desarrollo de la aplicación se llevan a cabo de modo transparente desde el IDE facilitando mucho el trabajo del programador. Como se ha señalado más arriba el lenguaje más habitual de desarrollo de aplicaciones Android es Java, sin embargo con el «middleware» apropiado (NDK, Native Development Kit) es posible crear aplicaciones nativas en C/C++. Este proceso lleva implícita la compilación cruzada de los programas hacia la plataforma destino constituida por la familia de procesadores ARM. Debido a la complejidad añadida a este proceso, su uso sólo se recomienda en casos que justifiquen la mejora del desempeño final de la aplicación.

La depuración de la aplicación desarrollada se puede hacer empleando un emulador o un dispositivo físico conectado mediante USB al computador en el que se realiza el desarrollo. El SDK de Android proporciona un emulador de dispositivo que facilita la depuración de los programas. También es posible recurrir a otros emuladores privativos de terceros, como el proporcionado gratuitamente por BlueStacks (2015). Aunque no es su propósito principal, este emulador se puede emplear en el desarrollo de aplicaciones de visión por computador pues su ejecución es bastante «ágil» y tiene acceso a la cámara del computador en el que se ejecuta.

En ocasiones la depuración de programas sobre un dispositivo físico es más recomendable ya que la aplicación se prueba en el dispositivo final y en general esto acorta los tiempos de desarrollo. No obstante, cuando se emplea un dispositivo real para las pruebas es importante tener los siguientes aspectos en mente:

1. La aplicación debe ser declarada en su fichero de manifiesto (AndroidManifest.xml) como «depurable» (*debuggable*). Esto es, dicho fichero debe añadir android:debuggable="true" al elemento <application>. Al emplear Eclipse para el desarrollo las declaraciones se actualizan de modo automático.
2. Activar la opción de depuración USB en el dispositivo físico Ajustes -> Aplicaciones -> Opciones de desarrollo -> Depuración USB. Debe recordarse que las opciones de desarrollo no están activadas por defecto y que su activación se consigue desde Ajustes -> Información del dispositivo y pulsando siete veces sobre el número de compilación.
3. Instalar el driver USB apropiado para que el dispositivo móvil sea reconocido por el computador anfitrión. El driver USB se distribuye junto con el SDK y puede localizarse en el SDK Manager de Android para su instalación local. Una vez conectado el dispositivo físico, el flujo de trabajo es el mismo al que se seguiría con el emulador.

17.5. Visión por computador en Android, primeros pasos

Como se ha comentado en la sección anterior las aplicaciones en Android se programan en Java aunque es posible desarrollar aplicaciones incluyendo código nativo en C/C++ mediante el empleo de NDK el cual permite compilar el código correspondiente en la aplicación de usuario. Con esta posibilidad en mente, a continuación se enumeran las posibles aproximaciones al desarrollo de aplicaciones de visión por computador en la plataforma Android.

17.5.1. Desarrollo directo en Java con API de Android

Usando la API android.hardware.camera2 (Google, 2015) es posible acceder desde una aplicación de usuario a las cámaras del dispositivo. Aunque puede ser una tarea ardua, el usuario puede desarrollar una aplicación «desde cero» para el procesamiento de las imágenes y vídeos capturados. Sin embargo, una estrategia para desarrollar de modo más rápido aplicaciones de visión por computador consiste en emplear librerías Java específicas de visión por computador.

17.5.2. Librerías Java de desarrollo de aplicaciones de Visión por Computador

La librería más popular entre los desarrolladores de aplicaciones de visión por computador es OpenCV (Bueno, 2015; Déniz, 2014; Déniz, 2013). Desde la versión 2.3.0 y hasta la última versión estable (3.0) esta librería de código abierto (*opensource*) proporciona una API Java para su empleo en Android (Itseez, 2015), trasladando a este entorno una parte muy importante de la funcionalidad presente en la librería original programada en C/C++. Existe alguna otra alternativa a OpenCV para el desarrollo sobre Android como es FastCV. Se trata de una librería de visión por computador liberada por Qualcomm optimizada para su ejecución en dispositivos Android aunque se trabaja en una versión para iOS. Esta librería se distribuye en forma de código binario con una única API que presenta dos implementaciones: una para arquitectura ARM (*FastCV for ARM*) y otra específicamente para SoC de Qualcomm SnapDragon (*Fast for SnapDragon*).

17.5.3. OpenCV para Android

La última versión estable liberada de OpenCV para Android se distribuye con varios ejemplos de aplicaciones (Itseez, 2015). Estos ejemplos son proyectos para Eclipse con ADT listos para usar. Para ejecutar estos ejemplos empleando el IDE Eclipse se comienza por importar la librería y los ejemplos en el espacio de trabajo actual (*workspace*). Para ello se ejecuta la opción File -> Import y en el panel de importación se selecciona General -> Existing Projects into Workspace.

Una vez importados los proyectos de ejemplo es muy posible que se produzcan errores en el proceso de compilación (*build*) de estos proyectos. Algunos de estos errores podrían desaparecer al reiniciar Eclipse, pero si persisten es necesario subsanarlos comenzando por aquellos relacionados con la propia librería OpenCV (Déniz, 2013):

- Error de inconsistencia de la plataforma de destino del proyecto. Este error se produce cuando en el sistema no está instalada la plataforma Android de destino para la cual se creó

el proyecto inicial importado. Para corregirlo basta con ir al panel de Propiedades del proyecto para elegir la plataforma de destino deseada (debe coincidir o ser inferior a la empleada por el emulador o dispositivo físico en el que se pruebe la aplicación).

- Error en la compilación con NDK en los proyectos que hacen uso de código nativo C/C++. Este error se produce en Eclipse cuando no se accede correctamente al comando de compilación ndk-build. Si se desea que la compilación sea automática desde Eclipse es preciso que la variable de entorno NDKROOT esté correctamente configurada.
- Error de ubicación del proyecto de librería OpenCV. Todos los nuevos proyectos OpenCV deben enlazar con dicha librería haciendo referencia a ella en su ubicación correcta. Si la importación de los proyectos de ejemplo se hace con copia completa al *workspace*, la referencia al *path* de la librería queda errónea y debe corregirse en las propiedades del nuevo proyecto.
- Errores sintácticos de reconocimiento de símbolos en el código nativo C/C++ incluido en los proyectos. Este error se produce cuando no se especifican (o se hace de modo incorrecto) las rutas de búsqueda de los ficheros de cabecera incluidos en el código nativo. Ajustando las propiedades del proyecto puede especificarse las rutas apropiadas incluyendo siempre la ruta <OPENCV4ANDROID>/sdk/native/jni/include. Siendo <OPENCV4ANDROID> la ruta donde se desempaquetó la versión de OpenCV para Android.

Una vez subsanados los errores se puede ejecutar el proyecto Android siguiendo el procedimiento convencional de cualquier aplicación. Es muy posible que en la primera ejecución se obtenga un mensaje señalando que el dispositivo no dispone del OpenCV Manager, Figura 17.6. Esta aplicación se puede instalar desde Google Play pero cuando no se dispone de conexión a Internet o se realizan pruebas con un emulador, es conveniente realizar la instalación manual de dos paquetes, suministrados con el SDK OpenCV4Android (ubicados en el directorio /apk):

- OpenCV_3.0.0_Manager_3.00_armeabi-v7a.apk
- OpenCV_3.0.0_Manager_3.00_armeabi.apk

La instalación manual de dichos paquetes se realiza con la herramienta adb. Es posible ejecutar los ejemplos de OpenCV tanto en un emulador como en un dispositivo real.

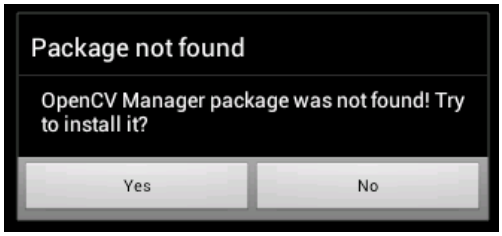


Figura 17.6. Ventana de error cuando OpenCV Manager no está instalado

En el tutorial-1-camerapreview suministrado en la distribución de OpenCV para Android se implementa el acceso a dicha librería mediante la API suministrada. El mecanismo de acceso a la librería se denomina de inicialización asíncrona en la que el gestor OpenCV Manager, instalado previamente en el dispositivo, actúa de intermediario. En este caso los pasos a seguir para crear un proyecto Android que haga uso de OpenCV son:

- Paso 1: Importación de proyecto de librería. Antes de crear el proyecto Android hay que importar en el espacio de trabajo la API Java de la librería para que pueda accederse a ella en el nuevo proyecto.
- Paso 2: Creación de nuevo proyecto Android. Se crea un nuevo proyecto Android cuya plataforma de destino sea la apropiada.
- Paso 3: Inclusión de la referencia a la librería. El proyecto Android debe incluir las referencias a las librerías que emplea y en concreto a la librería previamente importada al espacio de trabajo.
- Paso 4: Inclusión del código de inicialización de la librería en la programación del proyecto Android.

El tutorial-2-mixedprocessing muestra como emplear código nativo C++ de OpenCV en un proyecto Android. El procedimiento de acceso a la librería es idéntico al comentado más arriba pero en este caso la llamada a las funciones de la librería se hace mediante la interfaz proporcionada por JNI (Java Native Interfaz) para acceder al código nativo de la librería OpenCV. Para poder hacer uso de este tutorial se debe seguir los pasos siguientes:

- Paso 1: Importación de proyecto de librería. Este paso es necesario para todos los proyectos Android que usen la librería OpenCV usen código nativo o no.
- Paso 2 Importación del proyecto tutorial 2 al *workspace* creando una copia local. Se recomienda hacer una copia local porque de este modo se puede alterar el código del proyecto sin modificar el proyecto original de partida. Sin embargo, esto obliga a modificar algunas referencias empleadas en el proyecto según se indica en los pasos siguientes.
- Paso 3: Inclusión de la referencia a la librería OpenCV (Properties->Android). El proyecto Android debe incluir las referencias a las librerías que emplea y en concreto a OpenCV.
- Paso 4: Edición del fichero ./jni/Android.mk. En este fichero se debe incluir la referencia correcta a la ubicación del fichero OpenCV.mk incluido en la distribución de OpenCV para Android, ubicado en <OPENCV4ANDROID>/sdk/native/jni/.
- Paso 5: Inclusión de los ficheros de cabecera para la correcta compilación con NDK del código nativo (Properties->C/C++ General->Path and Symbols->[Includes]).

La Figura 17.7 muestra la salida del tutorial 2 ejecutado sobre un dispositivo físico.

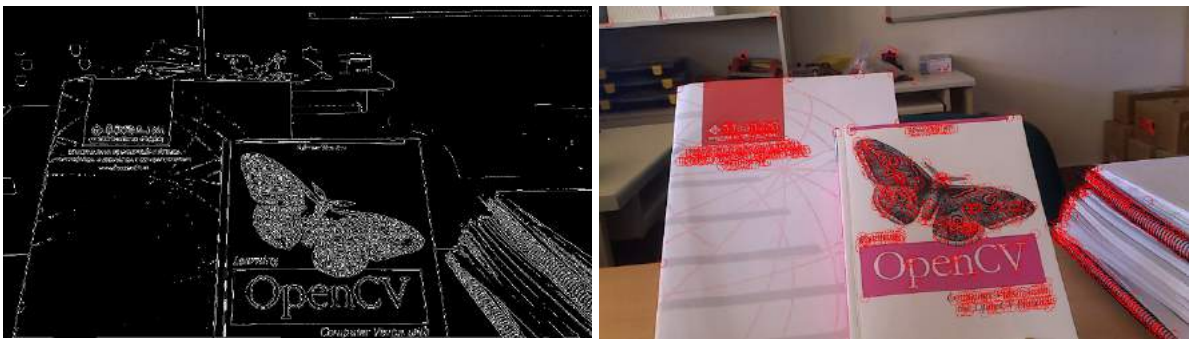


Figura 17.7. Salida del tutorial 2 ejecutando los algoritmos *Canny* y *Find Features*

17.5. Bibliografía

BlueStacks. (2015) Emulador Android. Disponible on-line: http://www.bluestacks.com/ (accedido 20 Abril 2015).

Bueno, G.; Déniz, O. y col. (2015) *Learning Image Processing with OpenCV*. Packt Publishing. ISBN: 978-17-832-8765-9.

Déniz, O.; Fernández, M. M. y col. (2014) *OpenCV Essentials*. Packt Publishing. ISBN: 978-17-839-8424-4.

Déniz, O.; Salido, J.; Bueno, G. (2013) *Programación de Apps de Visión Artificial*; Bubok Publishing S.L.: Madrid, España. ISBN: 978-84-686-3801-0. Disponible online: http://www.bubok.es/libros/224862/Programacion-de-Apps-de-Vision-Artificial (accedido 20 Abril 2015).

Eckel, B. (2006) *Thinking in Java*, 4ª ed.; Prentice-Hall. ISBN: 978-01-318-7248-6.

Howse, J. (2013) *Android Application Programming with OpenCV*. Packt Publishing. ISBN: 978-18-496-9520-6.

Itseez (2015). OpenCV for Android Platform. Disponible on-line: http://opencv.org/platforms/android.html (accedido 20 de Abril de 2015).

Google Inc. (2015) *Android Developers Site*. Disponible on-line: http://developer.android.com/ (accedido 20 de Abril de 2015).

Myagkov, A. (2012) Porting OpenCV to iOS. Disponible on-line: https://code.google.com/p/opencv-on-ios/source/browse/#svn%2Ftrunk%2Futilities (accedido 20 de Abril de 2015).

Shishkov, A. Kornyakov, K. (2013) *Instant OpenCV for iOS*. Packt Publishing. ISBN: 978-17-821-6384-8.

CAPÍTULO 18

ALGEBRA LINEAL BÁSICA PARA VISIÓN POR COMPUTADOR

Rafael C. GONZÁLEZ[1], José A. CANCELAS[1], Ignacio ÁLVAREZ[1], José M. ENGUITA[1]

[1] Profesores Titulares de Universidad: Departamento de Ingeniería Eléctrica, Electrónica, de Computadores y Sistemas, Universidad de Oviedo, Gijón, España

El álgebra lineal juega un papel importante en la visión por computador. Dentro del procesamiento de imagen es importante en aplicaciones como las transformaciones geométricas de la imagen. Sin embargo, cobra especial importancia en las aplicaciones de visión 3D. En este campo, el álgebra lineal es la herramienta necesaria para realizar la proyección de puntos, estimar la posición y orientación de la cámara, estimar parámetros del sistema o calcular la posición 3D de un punto entre otras muchas aplicaciones.

En este capítulo se repasan los conceptos, operaciones y transformaciones algebraicas más importantes desde el punto de vista del cálculo vectorial y matricial, así como su interpretación geométrica.

18.1. Introducción

El álgebra lineal es una herramienta básica para la realización de cálculos en geometría o en tratamiento de datos. En este capítulo se hace un breve repaso al cálculo vectorial y matricial, para facilitar el manejo posterior de los modelos matemáticos empleados en visión 3D. Se comienza con una descripción de qué son los vectores y las matrices, y de las operaciones que se pueden realizar con ellos. Se hace especial énfasis en el manejo de las operaciones mediante bloques, ya que el manejo de estas técnicas facilita el entendimiento de muchas de las operaciones que se realizan. En el caso del cálculo matricial, se describen las principales características de las aplicaciones lineales como método para transformar el espacio de trabajo. Se presta especial atención a la interpretación geométrica de muchas de las operaciones descritas. La descomposición en valores singulares de una matriz y sus usos dentro de la visión por computador reciben una especial atención. El capítulo finaliza con la

descripción de un algoritmo para factorizar una matriz en el producto de una matriz triangular superior por una matriz ortonormal que representa una rotación, ya que este método es fundamental en la factorización de la matriz de calibración de la cámara.

18.2. Vectores y matrices

Un vector de coeficientes reales de dimensión *n*, $\mathbf{x} \in \mathbb{R}^n$, es una secuencia de *n* números reales $[x_1, x_2, \ldots, x_n]^T$, organizados como una columna. Cada componente de la tupla se denomina componente. El vector nulo es aquel que tiene todas sus componentes iguales a cero.

En visión por computador los vectores se emplean:

- Para describir diferentes entidades geométricas. Por ejemplo, un punto del espacio se representa mediante un vector cuyas componentes son las coordenadas del punto en dicho espacio. Gráficamente lo podemos visualizar como una flecha cuyo origen es el origen del sistema de coordenadas y su extremo está en el propio punto.
- Para representar cualquier conjunto de datos ordenado, como por ejemplo un histograma o el descriptor de un punto característico.

Una matriz real de dimensión *m*×*n*, $\mathbf{A} \in \mathbb{R}^{m \times n}$, es una secuencia de *m*×*n* valores numéricos organizados en *m* filas y *n* columnas. También puede interpretarse como una secuencia de *n* columnas de vectores de dimensión *m*. Para referirse a un elemento individual de la matriz se utilizan el número de fila seguido del número de la columna, así el elemento que está en la fila *i*, columna *j* se representa como a_{ij}. Se denomina diagonal principal de la matriz (o simplemente diagonal) a los elementos cuyo número de fila es igual al número de columna. También podemos interpretar una matriz como una secuencia de vectores fila $\mathbf{A} = \begin{bmatrix} \mathbf{a}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{a}_{\mathrm{m}}^{\mathrm{T}} \end{bmatrix}, \mathbf{a}_i \in \mathbb{R}^n$, o como una secuencia de vectores columna $\mathbf{A} = [\mathbf{a}_1 \quad \cdots \quad \mathbf{a}_n], \mathbf{a}_i \in \mathbb{R}^m$.

En visión por computador las matrices se utilizan para almacenar la imagen, aunque también se utilizan en el modelo matemático de la cámara o por ejemplo para describir la relación existente entre dos imágenes de una misma escena.

Una matriz 1×*n* puede considerarse como un vector (o vector columna), mientras que una matriz *m*×1 puede considerarse como la traspuesta de un vector (o vector fila). Una matriz cuadrada es aquella que tiene el mismo número de filas que de columnas. La matriz nula es aquella cuyos elementos son todos nulos. La matriz identidad es aquella que tiene sus elementos de la diagonal iguales a 1 y el resto cero.

18.3. Operaciones con vectores

La **traspuesta** de un vector es otro vector cuyos elementos se consideran dispuestos en forma de una fila en lugar de una columna. Si una matriz es igual a su traspuesta, se dice que es una matriz simétrica. En el caso de que la matriz sea igual a su traspuesta cambiada de signo, se dice que es antisimétrica.

El **módulo** de un vector se define como $\|\mathbf{x}\| = \sqrt{\sum_i x_i^2}$. El módulo es siempre una cantidad positiva. Un vector cuyo módulo es la unidad, se denomina vector unitario.

La **multiplicación de un vector x por un escalar** λ es otro vector $\lambda\mathbf{x} = [\lambda x_1, \lambda x_2, \dots, \lambda x_n]^T$ donde $\lambda \in \mathbb{R}$ y $\mathbf{x} \in \mathbb{R}^n$. La multiplicación de un vector no modifica su dirección, sólo afecta a su módulo. Para obtener el vector unitario en una dirección, es suficiente con multiplicar el vector por el inverso de su módulo.

La **suma** de dos vectores es otro vector de la forma $\mathbf{z} = \mathbf{x} + \mathbf{y} = [x_1 + y_1,\ x_2 + y_2,\ \dots, x_n + y_n]$.

Un vector $\mathbf{z}$ de la forma $\mathbf{z} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i$, con $\alpha_i \in \mathbb{R}$, se dice que es una **combinación lineal** de los vectores $\mathbf{x}_\mathrm{i} \in \mathbb{R}^n$. Los vectores $\mathbf{x}_\mathrm{i}$ son linealmente independientes si la única forma expresar el vector nulo como combinación lineal de dichos vectores, es que $\alpha_i = 0\ \forall i$.

El **producto escalar** de dos vectores $\mathbf{x}$ e $\mathbf{y}$**,** es un escalar tal que $d = \mathbf{x} * \mathbf{y} = \mathbf{x}^\mathrm{T}\mathbf{y} = \sum_i x_i y_i$. El producto escalar también puede expresarse en función del módulo de los vectores y del ángulo que forman $\mathbf{x} * \mathbf{y} = \|\mathbf{x}\|\ \|\mathbf{y}\| \cos(\alpha)$. Desde el punto de vista geométrico, el producto escalar representa la proyección de un vector en la dirección representada por el otro, o dicho de otra manera, qué parte de la información contenida en el vector $\mathbf{x}$ se puede explicar mediante el vector $\mathbf{y}$. Si el producto escalar de dos vectores es cero, se dice que son ortogonales o perpendiculares. En ese caso, no existe ninguna relación entre la información que portan ambos vectores. El producto escalar admite la propiedad conmutativa y distributiva.

El **producto vectorial** de dos vectores $\mathbf{x}$ e $\mathbf{y}$ es otro vector perpendicular a los otros dos vectores y cuyo módulo es igual al producto de los módulos $\mathbf{x}$ e $\mathbf{y}$ por el seno del ángulo que forman. En consecuencia, el producto vectorial de dos vectores paralelos es el vector nulo. Geométricamente representa la dirección de la normal al plano definido por $\mathbf{x}$ e $\mathbf{y}$, y su módulo es igual al área del paralelogramo que definen dichos vectores. El producto vectorial de dos vectores en el espacio de dimensión 3 es $\boldsymbol{z} = \boldsymbol{x}\times\boldsymbol{y} = [(x_2y_3 - x_3y_2), (x_3y_1 - x_1y_3), (x_1y_2 - x_2y_1)\ ]^T$. El producto vectorial puede expresarse como el producto de una matriz por un vector mediante el uso de la matriz antisimétrica asociada a un vector $\mathbf{x}$:

$$[\mathbf{x}]_\times = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$$

Entonces, se verifica que $\boldsymbol{x}\times\boldsymbol{y} = [\mathbf{x}]_\times\mathbf{y} = [\mathbf{y}]_\times^\mathrm{T}\mathbf{x}$. Cumple la propiedad distributiva y anticonmutativa.

El **producto externo** de los vectores $\mathbf{x} \in \mathbb{R}^m$ e $\mathbf{y} \in \mathbb{R}^n$ se define como $\mathbf{A} = (\mathbf{x}\mathbf{y}^T) \in \mathbb{R}^{mxn}$. El resultado es una matriz *m*×*n* tal que la columna *j* es una copia del vector $\mathbf{x}$ escalado según la componente *j* del vector $\mathbf{y}$, es decir $\mathbf{A} = [y_1\mathbf{x} \quad y_2\mathbf{x} \quad \cdots \quad y_n\mathbf{x}]$. También puede verse la matriz $\mathbf{A}$ como una matriz cuyas fila *i* es una copia del vector $\mathbf{y}^T$ escalado según la componente *i* del vector $\mathbf{x}$. Se verifica que $a_{ij} = x_i y_j$.

18.4. Operaciones con matrices

La **traspuesta** de una matriz $\mathbf{A} = [a_{\mathrm{ij}}]$ es la matriz que se obtiene al intercambiar sus filas por sus columnas, con lo que resulta $\mathbf{A}^{\mathrm{T}} = [a_{ji}]$**.**

La **traza** de una matriz se define como la suma de los elementos de su diagonal principal: $\mathrm{tr}(\mathbf{A}) = \sum_i a_{\mathrm{ii}}$**.**

La **multiplicación de una matriz A por un escalar** $\boldsymbol{\lambda}$ es otra matriz de las mismas dimensiones que **A**, tal que $\mathbf{B} = \lambda\mathbf{A} = [\lambda a_{ij}]$, con $\lambda \in \mathbb{R}$.

Se define la **suma de dos matrices** $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m\times n}$, como la matriz $\mathbf{C} = \mathbf{A} + \mathbf{B} \mid c_{ij} = a_{ij} + b_{ij}$. La matriz **C** tiene las mismas dimensiones que **A** y **B**. La suma de matrices cumple es asociativa y conmutativa, y el elemento nulo es la matriz nula. Se verifica que $(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$**.**

Se define el **producto de la matriz** $\mathbf{A} \in \mathbb{R}^{\boldsymbol{m\times n}}$ **y el vector** $\mathbf{x} \in \mathbb{R}^{\boldsymbol{n}}$, como un vector $\mathbf{y} = \mathbf{A}\mathbf{x} \in \mathbb{R}^m$ cuyos elementos son el producto escalar de la fila correspondiente de la matriz por el vector: $y_i = \mathbf{a}_i^T\mathbf{x}$. Se verifica que $\mathbf{y} = \sum_j x_j\mathbf{a}_j$, por lo que **y** puede interpretarse como una combinación lineal de las columnas de **A** en la que los pesos vienen dados por las componentes del vector **x**. Una aplicación interesante de esta operación es que para extraer la fila *i* de una matriz, es suficiente con multiplicar la matriz por un vector cuyas componentes son todas cero a excepción de la componente *i* que vale 1.

Se define el **producto del vector** $\mathbf{x} \in \mathbb{R}^{\boldsymbol{m}}$ **por la matriz** $\mathbf{A} \in \mathbb{R}^{\boldsymbol{m\times n}}$ **y**, como un vector $\mathbf{y} = \mathbf{x}^T\mathbf{A} \in \mathbb{R}^n$ cuyos elementos son el producto escalar del vector por la columna correspondiente de la matriz: $y_i = \mathbf{x}^T\mathbf{a}_i$. Se verifica que $\mathbf{y}^T = \sum_i x_i\mathbf{a}_i^T$, por lo que puede interpretarse como una combinación lineal de las filas de **A**, en la que los pesos vienen dados por las componentes del vector **x**. Una aplicación interesante de esta operación es que para extraer la columna *i* de una matriz, es suficiente con multiplicar un vector cuyas componentes son todas cero a excepción de la componente *i* que vale 1 por la matriz.

Se define el **producto de dos matrices** $\mathbf{A} \in \mathbb{R}^{m\times p}$ y $\mathbf{B} \in \mathbb{R}^{p\times n}$, como la matriz $\mathbf{C} = \mathbf{AB} \mid c_{ij} = \mathbf{a}_i^T\mathbf{b}_j = \sum_p a_{ip}b_{pj}$. La matriz **C** tiene dimensiones *m*×*n*. El producto de matrices cumple la propiedad asociativa y la distributiva, pero no cumple la propiedad conmutativa. Se verifica que $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$. Es conveniente conocer cómo se puede desarrollar el producto de matrices cuando éstas están expresadas en forma de bloques:

$$\mathbf{C} = \mathbf{AB} = \mathbf{A}[\mathbf{b}_1\mathbf{b}_2 \ldots \mathbf{b}_{\mathrm{n}}] = [\mathbf{A}\mathbf{b}_1\ \mathbf{A}\mathbf{b}_2 \cdots \mathbf{A}\mathbf{b}_{\mathrm{n}}]$$

$$\mathbf{C} = \mathbf{AB} = \begin{bmatrix}\mathbf{a}_1^{\mathrm{T}}\\ \vdots \\ \mathbf{a}_{\mathrm{m}}^{\mathrm{T}}\end{bmatrix}\mathbf{B} = \begin{bmatrix}\mathbf{a}_1^{\mathrm{T}}\mathbf{B}\\ \vdots \\ \mathbf{a}_{\mathrm{m}}^{\mathrm{T}}\mathbf{B}\end{bmatrix}$$

$$\mathbf{C}_{m\times n} = \mathbf{A}_{(m\times p)}\mathbf{B}_{p\times n} = \left[\mathbf{A}_{m\times p_1}\middle|\mathbf{A}_{m\times p_2}\right]\begin{bmatrix}\mathbf{B}_{p_1\times n}\\ - \\ \mathbf{B}_{p_2\times n}\end{bmatrix} = \mathbf{A}_{m\times p_1}\mathbf{B}_{p_1\times n} + \mathbf{A}_{m\times p_2}\mathbf{B}_{p_2\times n}$$

Dada una matriz cuadrada **A**, su **inversa** $(\mathbf{A}^{-1})$ si existe, es una matriz tal que el producto de ambas es igual a la matriz unidad $\mathbf{I} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1}$. Si una matriz cuadrada no admite inversa se dice que es **singular**. Para que una matriz admita inversa, todas sus filas (o sus columnas) deben ser linealmente independientes. Se verifica que $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$**.**

En el caso de matrices rectangulares se define el concepto de matriz pseudoinversa. Dada una matriz $\mathbf{B} \in \mathbb{R}^{m\times n}(m \geq n)$ con todas sus columnas independientes (ran(**B**)=n), entonces la matriz $\mathbf{B}^T\mathbf{B}$

es invertible y la pseudoinversa de $\mathbf{B}$ es $\mathbf{B}^{+} = (\mathbf{B}^{T}\mathbf{B})^{-1}\mathbf{B}^{T}$. Para una matriz $\mathbf{B} \in \mathbb{R}^{m\times n}$ $(m < n)$ con todas sus filas independientes (ran(**B**)=m), entonces la matriz $\mathbf{B}\mathbf{B}^{T}$es invertible y la pseudoinversa de **B** es $\mathbf{B}^{+} = \mathbf{B}^{T}(\mathbf{B}\mathbf{B}^{T})^{-1}$. Una matriz se dice que es ortogonal si su inversa es igual a su traspuesta.

El **determinante** de una matriz cuadrada ($\det(\mathbf{A})$ o $|\mathbf{A}|$) es un escalar que se calcula a partir de los elementos de la matriz. Una de formas para hacer el cálculo es aplicar la fórmula de Laplace:

$$\det(\mathbf{A}) = \sum_{i=1}^{n}(-1)^{i+k}a_{ik}|\mathbf{M}_{ik}| = \sum_{j=1}^{n}(-1)^{k+j}a_{kj}|\mathbf{M}_{kj}|$$

donde $k \in [1 \ldots n]$, $\mathbf{M}_{ij}$ es la matriz que se obtiene al eliminar la fila *i* y la columna *j* de la matriz **A**. Para que una matriz admita inversa, su determinante debe ser distinto de cero. El valor del determinante está relacionado con el volumen del paralelepípedo formado por las columnas (o las filas) de la matriz. Se verifica que:

$$\det(\lambda\mathbf{A}) = \lambda^{n}\det(\mathbf{A}), \quad \det(\mathbf{A}\mathbf{B}) = \det(\mathbf{A})\det(\mathbf{B}), \quad \det(\mathbf{A}^{-1}) = \frac{1}{\det(A)}$$

Se define el **producto de Kronecker** de dos matrices $\mathbf{A} \in \mathbb{R}^{m\times n}$ y $\mathbf{B} \in \mathbb{R}^{q\times p}$,$\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$, como una matriz *mn*×*pq* tal que:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix}$$

Aplicaciones lineales

Una matriz $\mathbf{A} \in \mathbb{R}^{m\times n}$ define una función que transforma un vector de $\mathbb{R}^{n}$ en un vector de $\mathbb{R}^{m}$ según la ecuación $\mathbf{y} = \mathbf{A}\mathbf{x}$. El vector **y** es pues una combinación lineal de las columnas de la matriz **A** en la que los coeficientes de la combinación son las distintas componentes de **x**. Esta operación se conoce como aplicación o transformación lineal.

Las aplicaciones lineales se usan en visión por computador para hacer transformaciones geométricas en el plano imagen o en el espacio, tal y como se representa en la figura 17.1. La estructura de la matriz **A** define las características que tendrá la transformación. Una transformación lineal siempre mantiene las propiedades de incidencia de los puntos que transforma, es decir, si tres puntos están alineados, sus transformadas también determinarán una recta. De la misma forma, si transformamos dos rectas que se cortan, las rectas trasformadas también se cortarán.

La imagen de una transformación lineal definida por la matriz $\mathbf{A} \in \mathbb{R}^{m\times n}$, Im(**A**), es el conjunto de puntos $\mathbf{y} \in \mathbb{R}^{m}$ para los que existe al menos un punto $\mathbf{x} \in \mathbb{R}^{n}$ tal que $\mathbf{y} = \mathbf{A}\mathbf{x}$. Cualquier combinación lineal de puntos de Im(**A**) es punto de Im(**A**). Se define el núcleo de una aplicación lineal, Ker(**A**), como el conjunto de puntos $\mathbf{x} \in \mathbb{R}^{n}$, cuya transformada es el vector nulo de $\mathbb{R}^{m}$. El núcleo de una aplicación lineal también se denomina kernel o subespacio nulo. Se denomina rango de la aplicación lineal definida por **A**, ran(**A**), a la dimensión del subespacio imagen. Se denomina nulidad de la aplicación lineal, nul(**A**), a la dimensión del subespacio nulo. Se cumple que si $\mathbf{A} \in \mathbb{R}^{m\times n}$, entonces ran(**A**)+nul(**A**)=n. Dado que el punto imagen se puede interpretar como una combinación lineal de las filas de la matriz, el rango es igual al número de columnas independientes que tiene la matriz **A**.

A un conjunto de vectores linealmente independientes entre sí y que permite expresar cualquier otro vector del espacio como una combinación lineal de los mismos, se le denomina base de dicho espacio

vectorial. Una base se dice ortogonal cuando todos los vectores son perpendiculares entre sí. Si además todos los vectores de la base son unitarios, la base se denomina ortonormal. Se denomina base canónica de un espacio $\mathbb{R}^n$ a la formada por los vectores $Bc = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n\}$ tales que el vector $\mathbf{e}_i$ tiene todas sus componentes iguales a cero excepto la componente i que vale 1.

Dada una base del espacio $\mathbb{R}^n$, $B = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$, cualquier otro vector del espacio puede expresarse de forma única como combinación lineal de los vectores de la base $\mathbf{x} = \sum_i x_i^B \mathbf{v}_i$. Los valores $[x_i^B]$ son las coordenadas del vector $\mathbf{x}$ en la base B.

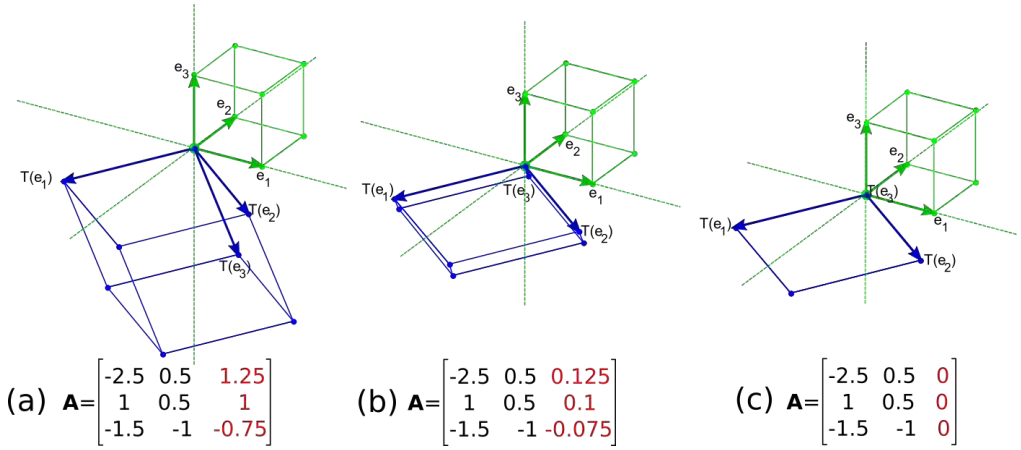


Figura 17.1. Interpretación geométrica de una transformación lineal.

Para describir completamente una transformación lineal de un espacio $\mathbb{R}^n$ en una espacio $\mathbb{R}^m$, es suficiente con conocer las imágenes de los vectores de una base de $\mathbb{R}^n$. Por la propiedad de linealidad, se tiene que:

$$\mathbf{y} = T(\mathbf{x}) = \mathbf{A}\mathbf{x} = T\left(\sum_i x_i^B \mathbf{v}_i\right) = \sum_i x_i^B T(\mathbf{v}_i) = [T(\mathbf{v}_1) \quad T(\mathbf{v}_2) \quad \cdots \quad T(\mathbf{v}_n)]\mathbf{x}$$

es decir, la matriz que define la transformación se obtiene colocando como columnas las transformadas de los vectores de la base de $\mathbb{R}^n$ respecto a la cual conocemos las coordenadas de los puntos que queremos transformar. El conjunto de vectores $\{T(\mathbf{v}_i)\}$ es un sistema generador del espacio imagen. El número de vectores $T(\mathbf{v}_i)$ linealmente independientes viene determinado por el ran($\mathbf{A}$), por lo que si eliminamos los vectores que sean linealmente dependientes, tendremos una base del espacio imagen.

Desde el punto de vista geométrico, cuando el espacio origen y el espacio imagen son de la misma dimensión una transformación lineal equivale a realizar un cambio de base. Cuando el espacio imagen es de menor dimensión que el espacio origen, tendremos una proyección.

En la figura 17.1 se muestra cómo actúa una transformación lineal $\mathbf{A}$ sobre el espacio de entrada. Los vectores de la base canónica $\{\mathbf{e}_i\}$, se transforman según la matriz $\mathbf{A}$ en los vectores $T(\mathbf{e}_i) = \mathbf{A}\mathbf{e}_i$. Cuando la matriz es de rango completo, ran($\mathbf{A}$)=3 como en los casos (a) y (b), un paralelepípedo se transforma en otro. El escalado en el nuevo espacio depende del valor del módulo de los vectores $T(\mathbf{e}_i)$. En el caso (b), el módulo del tercer vector se ha reducido por un factor de 10 con respecto al caso (a), lo que se traduce en una reducción proporcional de las longitudes en la dirección de $T(\mathbf{e}_3)$. En el caso (c) se aprecia el efecto producido cuando una columna se anula. En ese caso, el rango de $\mathbf{A}$

es 2 y la imagen ha quedado reducida a un plano. La situación sería la misma si la tercera columna fuese una combinación lineal de las otras dos.

Se dice que un vector **v**, no nulo, es un vector propio de la matriz cuadrada **A** si se verifica que $T(\mathbf{v}_i) = \mathbf{A}\mathbf{v}_i = \lambda\mathbf{v}_i$, donde λ es un escalar que se denomina valor propio asociado al vector. En consecuencia, los vectores propios no cambian de dirección sino que sólo sufren un escalado. El valor propio representa el factor de escala que se aplica al vector. El conjunto de vectores propios asociados a un mismo valor propio forman un subespacio vectorial y la dimensión de dicho subespacio es el orden de multiplicidad asociado al valor propio. Los valores propios de la matriz son las raíces de su polinomio característico, es decir, se obtienen resolviendo la ecuación $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$.

No todas las matrices admiten una descomposición en valores propios, pero puede demostrarse que las matrices simétricas tienen todos sus valores propios reales, que los autovalores correspondientes a valores propios distintos son ortogonales entre sí y que es posible encontrar una base ortonormal de $\mathbb{R}^n$ por n vectores propios de **A**.

Descomposición en valores singulares de una matriz

El resultado obtenido para las matrices simétricas es muy interesante ya que garantiza que podemos encontrar una base de $\mathbb{R}^n$ en la que es más sencillo explicar el efecto de la transformación lineal ya que éste reduce a escalar de forma independiente cada una de las componentes del vector expresado en coordenadas de dicha base ortonormal.

La descomposición en valores singulares nos permite obtener un resultado similar para una matriz $\mathbf{A} \in \mathbb{R}^{m\times n}$. En particular, la descomposición en valores singulares (SVD) de una matriz consiste en hacer una descomposición de la misma de la forma:

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T \tag{18.1}$$

donde $\mathbf{V} \in \mathbb{R}^{n\times n}$, $\mathbf{U} \in \mathbb{R}^{m\times m}$ son matrices ortonormales y $\mathbf{D} \in \mathbb{R}^{m\times n}$ es una matriz que sólo tiene valores no nulos a lo largo de su diagonal principal. Los valores de la diagonal principal de D se conocen como valores singulares de A, son no negativos y suelen ordenarse por valores decrecientes. Los valores singulares están relacionados con la raíz de los valores propios de la matriz simétrica obtenida como $\mathbf{A}^{\mathrm{T}}\mathbf{A}$.

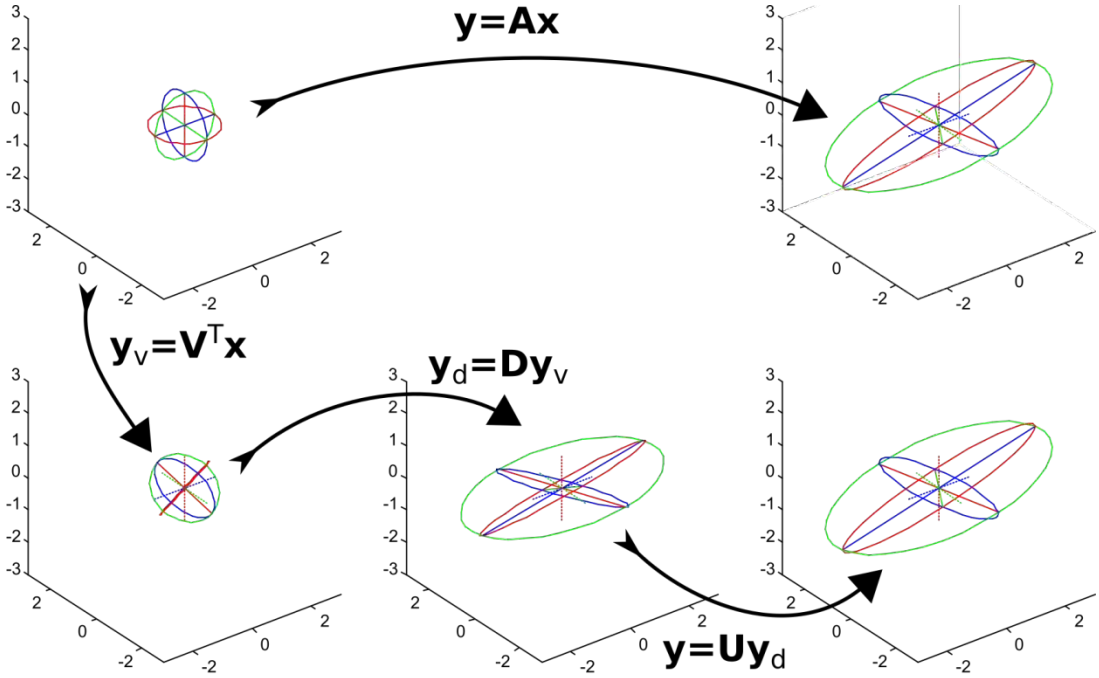


Figura 17.2. Interpretación geométrica de la descomposición en valores singulares de A: A = UDV^T.

Geométricamente, la matrices **V** y **U** permiten aplicar un cambio de base en los espacios de entrada y salida respectivamente de forma que, en esas bases la transformación sería lineal. Esta idea está representada en la figura 17.. El resultado de aplicar la transformación **A** es el mismo que se obtiene haciendo un cambio de base mediante la matriz $\mathbf{V}^T$, que deje los ejes orientados según las direcciones principales de la transformación. A continuación se aplica la trasformación **D**, que al tratarse de una matriz diagonal, modifica cada componente de forma independiente. La transformación **U** hace un cambio de base para dejar el resultado en la misma base del espacio $\mathbb{R}^m$ que la matriz **A**. Debe tenerse en cuenta que un cambio de base realizado con una matriz ortonormal consiste en aplicar una rotación de los ejes iniciales.

La matriz **A** puede expresarse como una suma ponderada de los productos exteriores de las columnas de **U** y **V**, de acuerdo con la fórmula:

$$\mathbf{A} = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T \tag{18.2}$$

donde k es el número de valores singulares no nulos de **A**.

Si **A** tiene k valores singulares no nulos y (n-k) valores singulares nulos, se verifica que:

1. Los vectores $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\}$, es decir, las k primeras columnas de **U** son una base del espacio imagen de **A**.
2. Los vectores $\{\mathbf{u}_{k+1}, \mathbf{u}_{k+2}, \ldots, \mathbf{u}_m\}$ son una base del complemento ortogonal de **A**. Como consecuencia, cualquier combinación lineal de estos vectores es perpendicular a cualquier elemento de la imagen de **A** y no puede ser imagen de ningún elemento del espacio de entrada.
3. Los vectores $\{\mathbf{v}_{k+1}, \mathbf{v}_{k+2}, \ldots, \mathbf{v}_n\}$ son una base del núcleo de la transformación lineal definida por **A**.

4. Los vectores $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$ son una base del complemento ortogonal del espacio nulo de **A** y por tanto está formado por aquellos vectores del espacio de entrada cuya transformación pertenece a la imagen de **A**.

Aplicaciones de la descomposición SVD

Desde el punto de vista del cálculo, la descomposición SVD es muy importante para el desarrollo de métodos numéricos de cálculo más estables y más eficientes. En el campo de la visión por computador la descomposición en valores singulares la encontraremos en diferentes tipos de problemas.

Compresión de imagen.

Está relacionado con el análisis de componentes principales (PCA) y el análisis estadístico de la covarianza de matrices. Se basa fundamentalmente en el uso la ecuación (18.2). Hay que tener en cuenta que las direcciones asociadas a los vectores propios más grandes son los que contienen la mayor parte de la información asociada a la matriz. Así, si tenemos una matriz de dimensiones $m \times n$, y la expresamos en función de los k primeros valores singulares, podremos almacenar una aproximación de la matriz usando las primeras k columnas de **U** y **V**, y un vector con los primeros k valores singulares. El ahorro es de (m-k)(m+n)-k elementos. La figura muestra un ejemplo. El tamaño original de la imagen es de 384 x 512 píxeles. Utilizando los primeros 51 valores singulares (un 10%) el tamaño necesario para almacenar la imagen es un 14.4% del tamaño original.

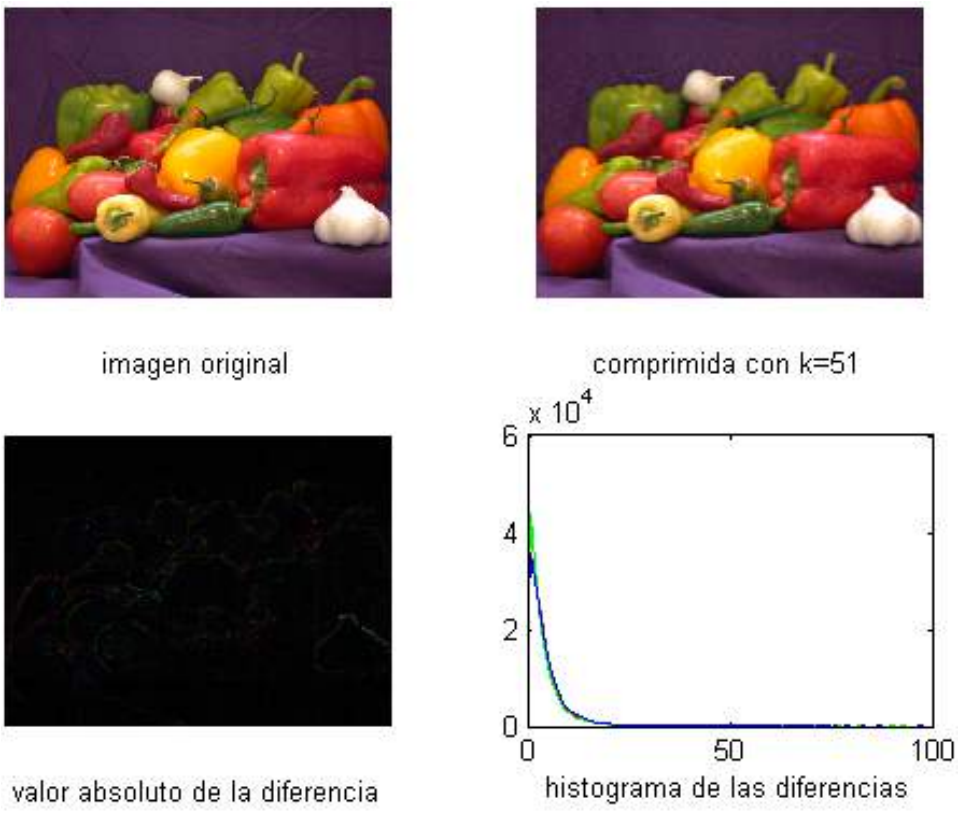


Figura17.3. Ejemplo de aplicación de la compresión de imágenes mediante descomposición SVD.

Solución de sistemas de ecuaciones

Otra aplicación importante en el campo de la visión por computador consiste en el uso de la descomposición SVD para la estimación lineal del modelo de la cámara, la matriz fundamental y la esencial o los epipolos de la imagen.

En muchos de estos casos se obtienen sistemas de ecuaciones con tres estructuras distintas:

1. Sistema de ecuaciones lineales sobredimensionado $\mathbf{Ax} = \mathbf{b}$. Aunque este caso suele resolverse mediante el uso de mínimos cuadrados lineales $\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\boldsymbol{A}^T\mathbf{b}$. La matriz $\mathbf{A}^T\mathbf{A}$ admite inversa siempre que el rango de **A** sea *n*, lo que está garantizado al ser el sistema sobredimensionado. El problema radica en que el cálculo de dicha matriz inversa está mal condicionado y puede llevar a soluciones totalmente inservibles debido a los problemas de representación asociados al uso de un ordenador. Sin embargo, podemos obtener la solución a partir de la descomposición SVD como $\mathbf{x} = \mathbf{V}\mathbf{D}^{+}\mathbf{U}^T\mathbf{b}$. Este resultado es muy útil porque la descomposición SVD puede calcularse de forma muy estable y el cálculo de la pseudoinversa de **D** sólo requiere sustituir la diagonal principal de **D** por la inversa del correspondiente valor singular y trasponer la matriz resultante.
2. Sistema de ecuaciones homogéneas de la forma $\mathbf{Ax} = \mathbf{0}$. El objetivo es calcular el conjunto de soluciones no triviales para la igualdad anterior. Basta comprobar que dicho conjunto es el subespacio nulo asociado a la matriz **A** y por tanto las soluciones serán combinaciones lineales de las (n-k) últimas columnas de **A**, donde k es el rango de **A**. Debido de nuevo a la precisión del cálculo, es posible que los valores singulares no sean exactamente cero. En ese caso se elige el suficiente número de los valores singulares más pequeños. Este problema es típico de problemas como la calibración de una cámara o la obtención de una matriz de homografía.
3. Sistema de ecuaciones homogéneas de la forma $\mathbf{x}^T\mathbf{A} = \mathbf{0}$. Este problema puede resolverse expresando **x** como una combinación lineal de las columnas de **U** asociadas a los (n-k) valores singulares más pequeños.

18.5. Descomposición KR

Un problema común en visión 3D es calcular la descomposición KR de una matriz, donde K es una matriz triangular superior y R es una matriz ortonormal. Este problema aparece por ejemplo en la factorización de la matriz de calibración de una cámara. Para resolver el problema se sigue el siguiente algoritmo:

Algoritmo17.1.

Objetivo: Obtener la descomposición **KR** de una matriz cuadrada **A** de dimensión 3×3.

Solución:

5. Calcular la matriz $\mathbf{Q}_x$ que aplica una rotación según el eje x, de modo que el elemento 3,2 de la matriz resultante es cero:

$$\mathbf{Q}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{bmatrix} \text{ con } c = -\frac{a_{33}}{\sqrt{a_{33}^2+a_{32}^2}} \text{ y } s = \frac{a_{32}}{\sqrt{a_{33}^2+a_{32}^2}}.$$

6. Hacer $\mathbf{A} \leftarrow \mathbf{A}\mathbf{Q}_x$ y $\mathbf{R} \leftarrow \mathbf{Q}_x$
7. Calcular la matriz $\mathbf{Q}_y$ que aplica una rotación según el eje y, de modo que el elemento 3,1 de la matriz resultante es cero:

$$\mathbf{Q}_y = \begin{bmatrix} c & 0 & s \\ 0 & 1 & 0 \\ -s & 0 & c \end{bmatrix} \text{ con } c = -\frac{a_{33}}{\sqrt{a_{33}^2+a_{31}^2}} \text{ y s}= -\frac{a_{31}}{\sqrt{a_{33}^2+a_{31}^2}}.$$

8. Hacer $\mathbf{A} \leftarrow \mathbf{A}\mathbf{Q}_y$ y $\mathbf{R} \leftarrow \mathbf{Q}_y\mathbf{R}$
9. Calcular la matriz $\mathbf{Q}_z$ que aplica una rotación según el eje z, de modo que el elemento 2,1 de la matriz resultante es cero:

$$\mathbf{Q}_z = \begin{bmatrix} c & -s & 0 \\ s & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ con } c = -\frac{a_{22}}{\sqrt{a_{22}^2+a_{21}^2}} \text{ y s}= \frac{a_{21}}{\sqrt{a_{22}^2+a_{21}^2}}.$$

10. Hacer $\mathbf{K} \leftarrow \mathbf{A}\mathbf{Q}_z$ y $\mathbf{R} \leftarrow \mathbf{Q}_z\mathbf{R}$
11. La solución son **K** y **R**

18.6. Bibliografía

Nielsen, H.B. (2010). *Linear Algebra for IT & Health* (Lecture Note). Informatics and Mathematical Modelling, Technical University of Denmark, DTU.

Strang, G. (2009). *Introduction to Linear Algebra* (Fourth Edition). Wellesley-Cambridge Press.

CAPÍTULO 19

GEOMETRÍA PROYECTIVA PARA VISIÓN 3D

Rafael C. GONZÁLEZ[1], José A. CANCELAS[1], Ignacio ÁLVAREZ[1], José M. ENGUITA[1]

[1] Profesores Titulares de Universidad: Departamento de Ingeniería Eléctrica, Electrónica, de Computadores y Sistemas, Universidad de Oviedo, Gijón, España

Nos movemos en un mundo que podemos describir correctamente mediante la geometría euclídea. Sin embargo, el proceso de proyección hace que la geometría de los objetos de la escena sufra grandes distorsiones. En general, no se conservan la mayor parte de las propiedades que se consideran invariantes en la geometría euclídea. La geometría proyectiva es una geometría más general, cuya principal ventaja es que permite manipular de forma sencilla y precisa todos los objetos de interés: puntos, rectas, planos o cónicas. En este capítulo se hace hincapié en la aplicación de esta herramienta al caso del plano proyectivo y al espacio proyectivo tridimensional. El primero es fundamental para razonar sobre los elementos geométricos de la escena y sus correspondientes en el plano imagen. El segundo es importante para establecer relaciones entre diferentes reconstrucciones de la escena.

19.1. Introducción

La geometría proyectiva es la herramienta matemática necesaria para poder modelar y manipular de forma sencilla los diferentes elementos geométricos que intervienen en la escena, resolviendo las limitaciones inherentes a la geometría euclídea. Con la geometría proyectiva se describe de forma simple todo el proceso de formación de la imagen desde un punto de vista puramente geométrico, dejando a un lado la fotogrametría.

Este capítulo está estructurado en tres secciones. La primera es la más importante, y en ella se introduce de forma intuitiva cómo interpretar los distintos elementos de la geometría. Las ecuaciones se obtienen a través de razonamientos geométricos con el fin de facilitar la comprensión de su significado. El siguiente apartado se dedica a describir los distintos tipos de transformaciones que existen. Por último, se introduce el espacio proyectivo de tres dimensiones, mediante analogías con los resultados obtenidos para el plano proyectivo.

19.2. El plano proyectivo

Antes de introducir una representación analítica de la geometría proyectiva, es conveniente pensar en cómo se genera una imagen. Los puntos de la imagen se obtienen como la intersección del rayo que une el centro de proyección de la cámara con la posición del punto en el espacio euclídeo. Conocida la posición del plano imagen, es indiferente conocer la dirección del rayo que proyecta un punto o la posición de su proyección en el plano imagen, ya que uno define al otro de forma unívoca. Consideremos un sistema de coordenadas cuyo origen es el centro de proyección y con el eje Z perpendicular al plano de la imagen. En este sistema, la recta que proyecta al punto $\mathbf{X}=[X,Y,Z]^T$ está formada por los puntos de la forma $\left\{k[X,Y,Z]^T / k \in \text{°} , k \neq 0\right\}$. Si el plano de imagen tiene como ecuación $Z=f$, resulta que $k=\frac{f}{Z}$, por lo que el punto de corete será $\left[x,y,f\right]^T=\left[\frac{fX}{Z},\frac{fY}{Z},f\right]^T$. Si se varía la posición del plano imagen, considerando distintos valores para f, la imagen obtenida es la misma salvo por un factor de escala. En consecuencia, en lo sucesivo consideraremos que f es igual a la unidad. Para definir el rayo de proyección es suficiente con conocer su vector director, ya que el rayo debe pasar por el centro de proyección situado en el origen del sistema de coordenadas.

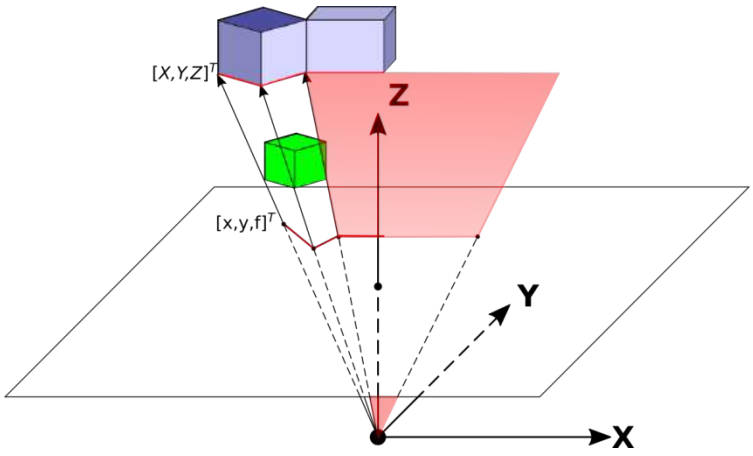


Figura 19.1. El plano proyectivo, $\mathcal{P}^2$, puede interpretarse como el conjunto de rayos que proyectan la escena en el plano imagen. Un punto se representa por el rayo que une el centro de proyección y su proyección en la imagen. Una recta se representa por el plano que contiene la proyección de la recta y el centro de proyección.

En la figura 19.se puede ver cómo podemos representar una recta en el plano proyectivo. Una recta del espacio y el centro de proyección determinan un plano cuya intersección con el plano imagen define la imagen de dicha recta. Resulta evidente que cualquier recta contenida en dicho plano produce la misma imagen. Para definir el plano es suficiente con conocer su vector normal ya que debe pasar obligatoriamente por el origen.

El planteamiento anterior permite introducir el concepto de coordenadas homogéneas. Sea un punto $\mathbf{x} = [x, y]^T$ un punto del plano imagen. Teniendo en cuenta la discusión del apartado anterior, se puede representar el punto del plano proyectivo $\mathcal{P}^2$, por un vector en la dirección de la recta de proyección. En particular, se puede obtener dicho vector como el vector que une el centro de proyección con la posición 3D del punto del plano imagen, con lo que resulta $\tilde{\mathbf{x}} = [x, y, 1]^T$. Teniendo en cuenta toda la discusión anterior, cualquier múltiplo no nulo del vector $\tilde{\mathbf{x}}$ representa el mismo punto. Esta forma de representar un punto del plano proyectivo $\mathcal{P}^2$ mediante un vector de $\mathbb{R}^3$, se conoce como representación mediante coordenadas homogéneas.

Para deshacer el cambio y determinar las coordenadas euclídeas del punto en el plano imagen basta con eliminar el factor de escala, haciendo que la tercera coordenada valga uno y tomando las dos primeras componentes:

$$\tilde{\mathbf{x}} = \left[\tilde{x}, \tilde{y}, \tilde{w}\right]^T = \tilde{w}\left[\frac{\tilde{x}}{\tilde{w}}, \frac{\tilde{y}}{\tilde{w}}, 1\right]^T \Rightarrow \mathbf{x} = \left[\frac{\tilde{x}}{\tilde{w}}, \frac{\tilde{y}}{\tilde{w}}\right]^T \tag{19.1}$$

Siempre es posible deshacer este cambio ya que por definición, para cualquier punto del plano la tercera componente de su representación en coordenadas homogéneas nunca puede ser cero. Es importante señalar que en coordenadas homogéneas, la igualdad debe ser interpretada siempre como una proporcionalidad, es decir, dos puntos del plano proyectivo $\tilde{\mathbf{x}}$ e $\tilde{\mathbf{y}}$ son iguales si sólo se diferencian en un factor de escala:

$$\tilde{\mathbf{x}} = \lambda\tilde{\mathbf{y}} \text{ , con } \lambda \neq 0 \in \mathbb{R} \tag{19.2}$$

Los vectores de $\mathbb{R}^3$ también se pueden usar para representar las rectas del plano proyectivo. Sea $\mathbf{l}$ una recta del plano imagen y sean $\mathbf{p} = [x_1, y_1]^T$ y $\mathbf{q} = [x_2, y_2]^T$ dos puntos de dicha recta. Las posiciones en el espacio 3D de dichos puntos son $\tilde{\mathbf{p}} = [x_1, y_1, 1]^T$ y $\tilde{\mathbf{q}} = [x_2, y_2, 1]^T$. Estos dos puntos definen, junto al centro de proyección, un plano Π cuya intersección con el plano imagen es la recta $\mathbf{l}$. Siguiendo la discusión de la sección anterior, se puede representar $\mathbf{l}$ de forma unívoca mediante el vector normal al plano Π. Para obtener dicho vector normal es suficiente con calcular el producto vectorial

$$\tilde{\mathbf{l}} = \tilde{\mathbf{p}} \times \tilde{\mathbf{q}} = \left[y_1 - y_2,\ x_2 - x_1, x_1 y_2 - x_2 y_1\right]^T \tag{19.3}$$

Al igual que antes, dos rectas serán iguales si sólo se diferencian en un factor de escala. En el caso de la recta, las dos primeras coordenadas no pueden ser simultáneamente cero ya que en ese caso los dos puntos serían el mismo y no definirían una recta.

Es fácil expresar la condición de pertenencia de un punto a una recta. Si un punto $\tilde{\mathbf{x}}$ pertenece a la misma recta que $\tilde{\mathbf{p}}$ y $\tilde{\mathbf{q}}$, entonces pertenece al plano definido por éstos y el centro de proyección, por lo que ha de ser perpendicular al vector normal a dicho plano:

$$\tilde{\mathbf{l}}^T\tilde{\mathbf{x}} = \tilde{\mathbf{x}}^T\tilde{\mathbf{l}} = 0 \tag{19.4}$$

Si se desarrolla la expresión anterior, se puede comprobar que coincide con la ecuación de una recta en el plano euclídeo. Sea $\tilde{\mathbf{l}} = [a, b, c]^T$ una recta del plano proyectivo y sea $\tilde{\mathbf{x}} = [x, y, 1]^T$ un punto de la recta, entonces la ecuación (19.4) puede escribirse como:

$$ax + by + c = 0 \tag{19.5}$$

De forma similar se puede deducir la expresión que permite calcular el punto de intersección, $\tilde{\mathbf{p}}$, de dos rectas del plano proyectivo expresadas en coordenadas homogéneas, $\tilde{\mathbf{l}}_1$ y $\tilde{\mathbf{l}}_2$. Dado que el punto de intersección pertenece a la recta $\tilde{\mathbf{l}}_i$, el rayo proyectivo debe pertenecer al plano que proyecta a la recta. En consecuencia, el vector director del rayo es perpendicular al vector normal al plano. Dado que $\tilde{\mathbf{p}}$ debe ser perpendicular a $\tilde{\mathbf{l}}_1$ y $\tilde{\mathbf{l}}_2$ se tiene que:

$$\tilde{\mathbf{p}} = \tilde{\mathbf{l}}_1 \times \tilde{\mathbf{l}}_2 \tag{19.6}$$

Observando la estructura de las ecuaciones, se pueden observar varias características interesantes del uso de coordenadas homogéneas:

1. Los puntos y las rectas del plano imagen se representan mediante vectores de $\mathbb{R}^3$.
2. Tanto en la condición de pertenencia como en la condición de intersección, los puntos y las rectas pueden cambiar su papel manteniéndose la estructura de la ecuación

Debido a estas propiedades, se dice que el punto y la recta del plano proyectivo son elementos duales.

Hasta ahora se ha considerado qué sucede con los vectores de $\mathbb{R}^3$ cuya tercera coordenada es cero ya que al hacer el cambio de coordenadas euclídeas del punto a las coordenadas homogéneas este caso no puede aparecer. Tampoco se ha analizado qué pasa con la ecuación (19.6) si las dos rectas son paralelas.

Dos rectas paralelas serán de la forma $\tilde{\mathbf{l}}_1 = [a, b, c_1]^T$ y $\tilde{\mathbf{l}}_2 = [a, b, c_2]^T$, ya que ambas deben tener el mismo vector director (*b,-a*). En el caso de la geometría euclídea, dos rectas paralelas no se cortan o bien se dice que se cortan en el infinito. En el caso de la geometría proyectiva tendremos que:

$$\tilde{\mathbf{p}} = \tilde{\mathbf{l}}_1 \times \tilde{\mathbf{l}}_2 = \left(c_2 - c_1\right)\left[b, -a, 0\right]^T = \lambda\left[x, y, 0\right]^T \tag{19.7}$$

Este es un resultado muy importante ya que supone que en el plano proyectivo todas las rectas se cortan. En particular, las rectas paralelas en el espacio euclídeo se cortan en un punto cuya tercera coordenada es cero, mientras que las otras dos coordenadas dependen exclusivamente de la dirección

de la recta. Por otra parte, los puntos del infinito pertenecen a una recta cuyas coordenadas homogéneas son $[0,0,1]^T$. Esta recta se denomina recta en el infinito.

19.3. Transformaciones en el plano proyectivo

Se denomina colineación, o transformación proyectiva, a una transformación del plano proyectivo $\mathcal{P}^2$ sobre sí mismo que transforma una línea en otra línea. Una colineación mantiene la colinealidad de cualquier conjunto de puntos. Se puede escribir como una transformación lineal de los puntos del plano proyectivo expresados en coordenadas homogéneas, en la que la matriz que realiza la transformación admite inversa. Cada punto del plano proyectivo se transforma según:

$$\tilde{\mathbf{x}}' = \mathbf{T}\tilde{\mathbf{x}} \tag{19.8}$$

En el caso del plano proyectivo, **T** es una matriz 3x3 y dado que el factor de escala no es importante, la matriz **T** sólo tiene 8 grados de libertad. Como cada punto contiene dos valores independientes, **T** queda definida si se conocen 4 parejas de puntos siempre que no haya más de dos pertenecientes a una misma línea.

Para determinar cómo se transforman las líneas, es suficiente con tener en cuenta que se debe preservar la colinealidad, por lo que si el punto $\tilde{\mathbf{x}}$ pertenece a la recta $\tilde{\mathbf{l}}$, sus respectivas transformaciones también deben mantener dicha relación:

$$\tilde{\mathbf{x}}^T\tilde{\mathbf{l}} = 0 = \left(\mathbf{T}^{-1}\tilde{\mathbf{x}}'\right)^T\tilde{\mathbf{l}} = \tilde{\mathbf{x}}'^T\left(\mathbf{T}^{-T}\tilde{\mathbf{l}}\right) \tag{19.9}$$

por lo que:

$$\tilde{\mathbf{l}}' = \mathbf{T}^{-T}\tilde{\mathbf{l}} \tag{19.10}$$

Una transformación proyectiva general no conserva ni las distancias, ni las relaciones entre distancias, ni el paralelismo. Sólo conserva la colinealidad y una relación denominada razón doble (en inglés cross ratio) entre cuatro puntos:

$$C_r(\mathbf{x}_1,\mathbf{x}_2;\mathbf{x}_3,\mathbf{x}_4) = \frac{d_{13}/d_{14}}{d_{23}/d_{24}} = \frac{d_{13}d_{24}}{d_{14}d_{23}} \tag{19.11}$$

donde d_{ij} representa la distancia entre los puntos *i*,*j* expresados en coordenadas euclídeas.

Las colineaciones pueden clasificarse en cuatro grupos según su estructura. La figura 19.2 muestra el efecto que sobre una misma figura tienen los distintos de transformaciones proyectivas:

1. **Transformaciones proyectivas**:

$$\mathbf{T} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix} \quad , \quad \exists \mathbf{T}^{-1} \tag{19.12}$$

Este tipo de transformación es la más general y sólo conserva las condiciones de pertenencia, colinealidad y la razón doble. La matriz **T** está definida salvo por un factor de escala, por lo que tiene 8 grados de libertad.

2. **Transformaciones afines**:

$$\mathbf{T} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ 0 & 0 & t_{33} \end{bmatrix} \quad , \quad \exists \mathbf{T}^{-1}, \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix}^{-1} \tag{19.13}$$

Este tipo de transformación mantiene invariante la recta en el infinito. Como consecuencia, además de conservar todas las propiedades de las matrices proyectivas, conserva el paralelismo entre líneas. La matriz **T** está definida salvo por un factor de escala, por lo que tiene 6 grados de libertad.

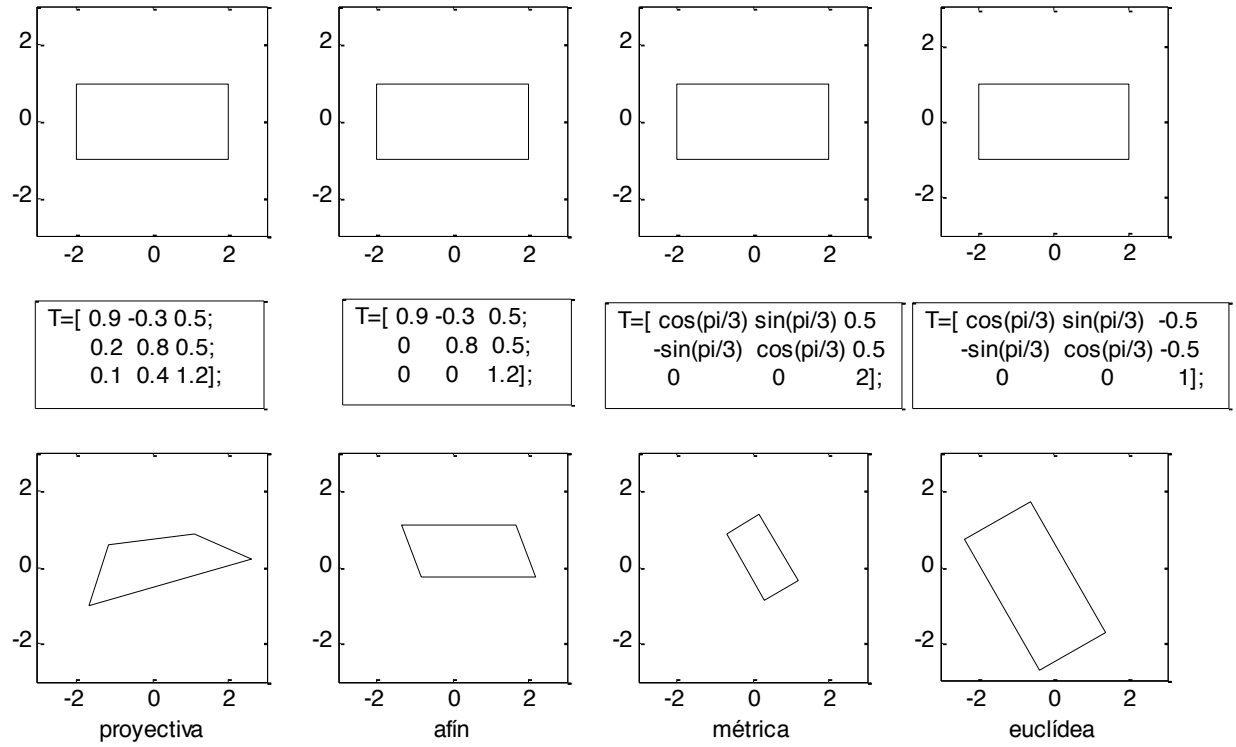


Figura 19.2. Ejemplos de los diferentes tipos de transformaciones proyectivas.

3. **Transformaciones métricas o de similaridad**:

$$\mathbf{T}=\begin{bmatrix}\cos(\alpha) & \sin(\alpha) & t_{13}\\ -\sin(\alpha) & \cos(\alpha) & t_{23}\\ 0 & 0 & t_{33}\end{bmatrix}\quad,\quad \exists\mathbf{T}^{-1} \tag{19.14}$$

Este tipo de transformación mantiene invariante la posición de los puntos dentro de la recta del infinito, por lo que además conserva los ángulos y las relaciones entre las distancias. La matriz **T** está definida salvo por un factor de escala, por lo que tiene 4 grados de libertad.

4. **Transformaciones euclídeas**:

$$\mathbf{T}=\begin{bmatrix}\cos(\alpha) & \sin(\alpha) & t_{13}\\ -\sin(\alpha) & \cos(\alpha) & t_{23}\\ 0 & 0 & 1\end{bmatrix}\quad,\quad \exists\mathbf{T}^{-1} \tag{19.15}$$

Este tipo de transformación es la más restrictiva, ya que además mantiene la escala por lo que también mantiene invariantes las distancias. La matriz **T** tiene 3 grados de libertad.

19.4. El espacio proyectivo $\mathcal{P}^3$

El espacio proyectivo $\mathcal{P}^3$ es análogo al espacio proyectivo $\mathcal{P}^2$. Hay una dualidad entre los puntos y los planos. En este caso, ambos tipos de entidades geométricas se representan mediante vectores de $\mathbb{R}^4$. La condición de pertenencia de un punto a un plano se determina con un producto vectorial como en la ecuación (19.4). De la misma forma, 3 puntos definen un plano:

$$\tilde{\mathbf{n}}=\begin{bmatrix}-\begin{vmatrix}y_1 & y_2 & y_3\\ z_1 & z_2 & z_3\\ w_1 & w_2 & w_3\end{vmatrix}, & \begin{vmatrix}x_1 & x_2 & x_3\\ z_1 & z_2 & z_3\\ w_1 & w_2 & w_3\end{vmatrix}, & -\begin{vmatrix}x_1 & x_2 & x_3\\ y_1 & y_2 & y_3\\ w_1 & w_2 & w_3\end{vmatrix}, & \begin{vmatrix}x_1 & x_2 & x_3\\ y_1 & y_2 & y_3\\ z_1 & z_2 & z_3\end{vmatrix}\end{bmatrix}^T$$

y 3 planos definen un punto:

$$\tilde{\mathbf{p}}=\begin{bmatrix}-\begin{vmatrix}y_1 & y_2 & y_3\\ z_1 & z_2 & z_3\\ w_1 & w_2 & w_3\end{vmatrix}, & \begin{vmatrix}x_1 & x_2 & x_3\\ z_1 & z_2 & z_3\\ w_1 & w_2 & w_3\end{vmatrix}, & -\begin{vmatrix}x_1 & x_2 & x_3\\ y_1 & y_2 & y_3\\ w_1 & w_2 & w_3\end{vmatrix}, & \begin{vmatrix}x_1 & x_2 & x_3\\ y_1 & y_2 & y_3\\ z_1 & z_2 & z_3\end{vmatrix}\end{bmatrix}^T$$

Los planos paralelos se cortan en un punto del infinito, que tiene la forma $[x,y,z,0]^T$. Todos los puntos del infinito forman el plano del infinito que tiene la forma $[0,0,0,1]^T$.

Al igual que el caso del plano proyectivo, las transformaciones proyectivas se clasifican en:

1. **Transformaciones proyectivas**:

$$\mathbf{T}=\begin{bmatrix}t_{11} & t_{12} & t_{13} & t_{14}\\ t_{21} & t_{22} & t_{23} & t_{24}\\ t_{31} & t_{32} & t_{33} & t_{34}\\ t_{41} & t_{42} & t_{43} & t_{44}\end{bmatrix}\quad,\quad \exists\mathbf{T}^{-1} \tag{19.16}$$

Este tipo de transformación es la más general y sólo conserva las condiciones de pertenencia, colinealidad y la razón doble. La matriz **T** está definida salvo por un factor de escala, por lo que tiene 15 grados de libertad.

2. **Transformaciones afines**:

$$\mathbf{T}=\begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ 0 & 0 & 0 & t_{44} \end{bmatrix} \quad , \quad \exists \mathbf{T}^{-1}, \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix}^{-1} \tag{19.17}$$

Este tipo de transformación mantiene invariante el plano en el infinito. Como consecuencia, además de conservar todas las propiedades de las matrices proyectivas, conserva el paralelismo entre líneas. La matriz **T** está definida salvo por un factor de escala, por lo que tiene 12 grados de libertad.

3. **Transformaciones métricas o de similaridad**:

$$\mathbf{T}=\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_{14} \\ r_{21} & r_{22} & r_{23} & t_{24} \\ r_{31} & r_{32} & r_{33} & t_{34} \\ 0 & 0 & 0 & t_{44} \end{bmatrix} \quad , \exists \mathbf{T}^{-1}, \mathbf{R}=\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} / \mathbf{R}\mathbf{R}^{\mathbf{T}}=\mathbf{I} \tag{19.18}$$

La matriz **R** es una matriz de rotación. Este tipo de transformación mantiene invariante la posición de los puntos dentro del plano del infinito, por lo que además conserva los ángulos y las relaciones entre las distancias. La matriz **T** tiene 7 grados de libertad.

4. **Transformaciones euclídeas**:

$$\mathbf{T}=\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_{14} \\ r_{21} & r_{22} & r_{23} & t_{24} \\ r_{31} & r_{32} & r_{33} & t_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad , \exists \mathbf{T}^{-1}, \mathbf{R}=\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} / \mathbf{R}\mathbf{R}^{\mathbf{T}}=\mathbf{I} \tag{19.19}$$

Este tipo de transformación es la más restrictiva, ya que además mantiene la escala por lo que también mantiene invariantes las distancias. La matriz **T** tiene 6 grados de libertad.

19.5. Bibliografia

Birchfield, S. (1998a). An introduction to projective geometry. Unpublished. Available: http://vision. stanford. edu/birch/projective.

Mohr, R.; Triggs, B. (1996). Projective geometry for image analysis. En XVIIIth International Symposium on Photogrammetry & Remote Sensing (ISPRS'96). Recuperado a partir de http://hal.univ-grenoble-alpes.fr/inria-00548361/

Mundy, J. L.; Zisserman, A. (1992). Appendix—projective geometry for machine vision. En Geometric invariance in computer vision (pp. 463–519). MIT Press

.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Índice de Términos

Grupo de Visión del
Comité Español de Automática (CEA)