4 Librerías de Visión por computador

En esta apartado se van a describir las librerías (Toolboxes) existentes en Matlab para implementar a aplicaciones de Visión por Computador. En la actualidad se disponen de varios paquetes tanto comerciales como de código abierto.

Matlab permite adquirir las siguientes toolboxes:

- ***Image Adquisition Toolbox***: proporciona funciones para la captura de imágenes a partir de diferentes dispositivos.
- ***Image Processing Toolbox***: proporciona funciones básicas de manejos de imágenes, lectura y escritura desde diferentes formatos de ficheros, conversión, preprocesamiento, filtrado, transformaciones espaciales y visualización.
- ***Computer Vision Toolbox***: implementa funciones avanzadas de procesamiento de imágenes centradas en la segmentación y descripción de imágenes, extracción de características, detección de movimiento, seguimiento de objetos, visión 3D, y procesamiento de video.

Entre los paquetes de código abierto disponibles, el más completo y utilizado es:

- ***Machine Vision Toolbox*** (***MVTB***) de Peter Corke: contiene una recopilación de código tanto propio como de otros autores que abarca desde la captura de imágenes, su preprocesamiento, filtrado, segmentación, extracción de características, visualización, y visión 3D.

En este tutorial nos centraremos en la implementación de aplicaciones de Visión por computador utilizando la librería de código libre **MVTB** de Peter Corke lo que nos permitirá trabajar con una versión básica de Matlab. Veremos el procedimiento de instalación, configuración y el manejo de las funciones principales. Hasta el momento esta librería no es compatible con Octave.

Adicionalmente, la librería base de Matlab dispone de algunas funciones para el manejo de imágenes que se describen a continuación.

4.1 Funciones base de Matlab para manejo de imágenes

La librería estándar de Matlab incorpora algunas funciones para la gestión básica de imágenes sin necesidad de instalar Toolboxes adicionales (tabla 4.1).

Manejo Imágenes	
`imread`	Lee una imagen desde un fichero
`imshow`	Muestra una imagen
`imwrite`	Graba una imagen en un fichero

Tabla 4.1 Funciones base de Matlab para el manejo de imágenes

La función para cargar el contenido de una imagen es **`imread`**. La imagen es devuelta como una matriz. Existen varias modalidades en función de los parámetros y el valor devuelto:

```
IM = imread(FILE, FMT)
```

Abre el fichero de imagen indicado en FILE (puede ser una URL). El segundo parámetro es opcional indicando el formato del fichero (tabla 4.2). Si no se indica se utiliza la extensión del fichero para seleccionar el formato.

```
[IM, MAP] = imread(FILE, FMT)
```

Abre el fichero de imagen indicado en FILE. Almacenado la matriz de datos en IM escalada [0,1] y el mapa de color en MAP.

Opciones	Descripción
'BMP'	Windows Bitmap
'CUR'	Windows Cursor resources
'GIF'	Graphics Interchange Format
'ICO'	Windows Icon resources
'JPG','JPEG'	Joint Photographic Experts Group
'J2C','J2K' 'JP2','JPF'	Joint Photographic Experts Group 2000
'PBM'	Formato Portable Bitmap
'PCX'	Windows Paintbrush
'PGM'	Portable Graymap
'PNG'	Portable Network Graphics
'PPM'	Portable Pixmap
'RAS'	Sun Raster
'TIF','TIFF'	Tagged Image File Format
'XWD'	X Window Dump

Tabla 4.2. Opciones de formato de fichero función imread

Ejemplo: se carga una imagen en color y se muestra la información con la función **about**

```
>> im = imread('flowers4.png');
>> about im
im [uint8] : 426x640x3 (817.9 kB)
>>
```

La función para visualizar el contenido de una imagen es **imshow**:

```
imshow(IM, OPTIONS)
```

Muestra una figura con la visualización de la imagen IM. El parámetro OPTIONS es opcional y permite configurar el escaldo, mapa de color, etc. Para más detalles consultar la ayuda de Matlab.

Ejemplo:

```
>> imshow(im)
```

La función para grabar el contenido de una imagen en un fichero es **imwrite**:

```
imwrite(IM, FILENAME, FMT)
```

Graba la imagen IM en el fichero indicado en FILENAME (puede ser una URL). El parámetro FMT es opcional indicando el formato del fichero (tabla 4.2). Si no se indica se utiliza la extensión del fichero para seleccionar el formato.

Ejemplo:

```
>> imwrite(im, 'resultado.pgm')
```

4.2 Instalación de la librería 'Machine Vision Toolbox' MVTB

La instalación de la librería se realizará descargando varios paquetes de la web del profesor de la Universidad de Melbourne Peter Corke (figura 4.1):

http://petercorke.com/wordpress/toolboxes/machine-vision-toolbox

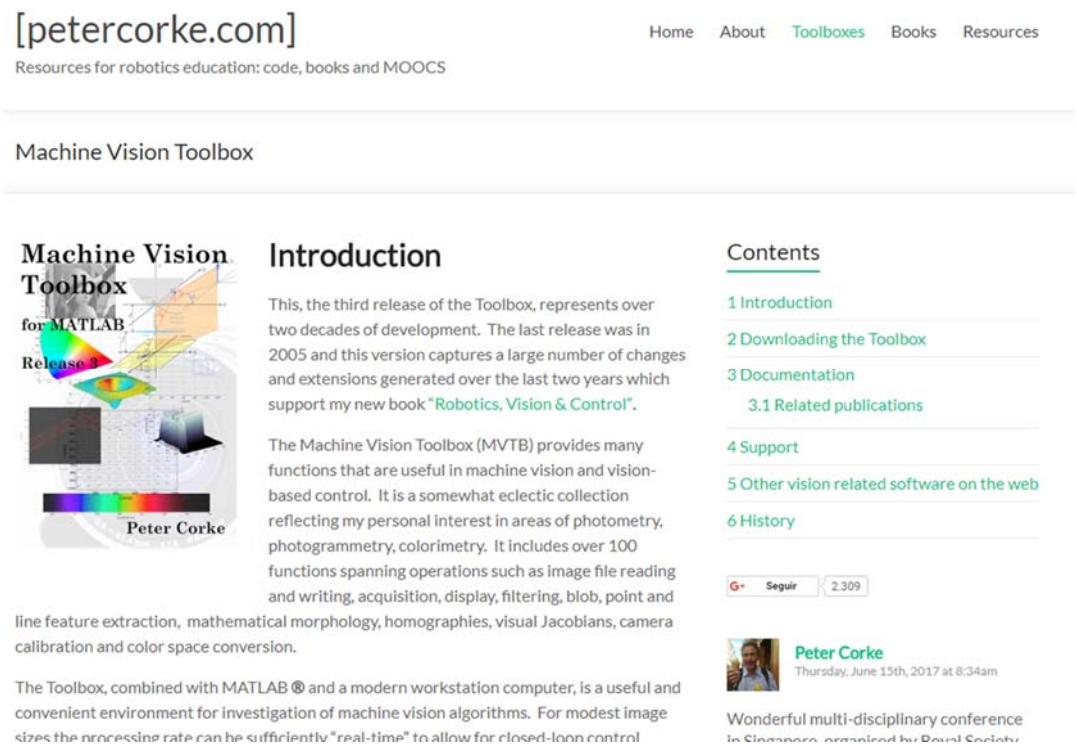


Figura 4.1 Portal web 'Machine Vision Toolbox' de Peter Corke

En la web se proporciona información sobre la librería, los enlaces de descarga, instrucciones de instalación, manuales y bibliografía complementaria. En esta web debemos acceder al enlace de descarga y tras solicitarnos los datos de nuestra organización, descargarnos los siguientes paquetes (figura 4.2):

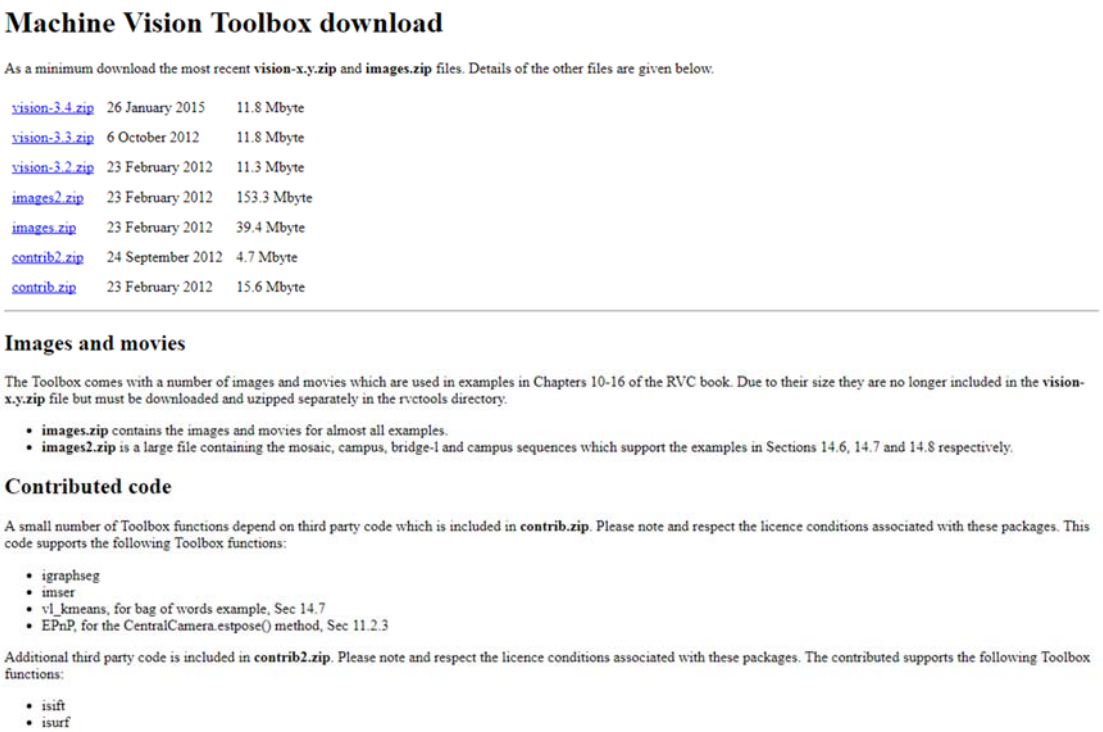
- Librería base: ***vision-3.4.zip***
- Paquete de imágenes de ejemplo: ***images.zip***
- Paquete de secuencias de imágenes: ***images2.zip***
- Código de otros autores: ***contrib.zip***
- Código adicional de otros autores (*isurf, isift*): ***contrib2.zip***

Nota: actualmente está disponible la nueva versión 4 de la librería. Se puede descargar desde la página anterior en dos formatos: un fichero comprimido único **vision-4.3.zip** o un instalador de Matlab (**mltbx**). La versión 3 está disponible en un enlace local (http://umh1782.edu.umh.es/material/software/).

Esta toolbox (versión 3) es compatible con MATLAB R2011a o posteriores. La versión 4 es compatible con Matlab R2016 o posteriores.

También podemos descargarnos el manual en formato pdf desde este enlace directo:

(http://www.petercorke.com/MVTB/vision.pdf)



Machine Vision Toolbox download

As a minimum download the most recent **vision-x.y.zip** and **images.zip** files. Details of the other files are given below.

vision-3.4.zip	26 January 2015	11.8 Mbyte
vision-3.3.zip	6 October 2012	11.8 Mbyte
vision-3.2.zip	23 February 2012	11.3 Mbyte
images2.zip	23 February 2012	153.3 Mbyte
images.zip	23 February 2012	39.4 Mbyte
contrib2.zip	24 September 2012	4.7 Mbyte
contrib.zip	23 February 2012	15.6 Mbyte

Images and movies

The Toolbox comes with a number of images and movies which are used in examples in Chapters 10-16 of the RVC book. Due to their size they are no longer included in the **vision-x.y.zip** file but must be downloaded and uzipped separately in the rvctools directory.

- **images.zip** contains the images and movies for almost all examples.
- **images2.zip** is a large file containing the mosaic, campus, bridge-l and campus sequences which support the examples in Sections 14.6, 14.7 and 14.8 respectively.

Contributed code

A small number of Toolbox functions depend on third party code which is included in **contrib.zip**. Please note and respect the licence conditions associated with these packages. This code supports the following Toolbox functions:

- igraphseg
- imser
- vl_kmeans, for bag of words example, Sec 14.7
- EPnP, for the CentralCamera.estpose() method, Sec 11.2.3

Additional third party code is included in **contrib2.zip**. Please note and respect the licence conditions associated with these packages. The contributed supports the following Toolbox functions:

- isift
- isurf

Figura 4.2 Página de descarga ‘Machine Vision Toolbox’ de Peter Corke

Para instalarla debemos elegir o crear la carpeta en la que vamos a instalarla y descomprimir el contenido de los cuatro paquetes indicados anteriormente (versión 3). El fichero comprimido contiene todo el árbol de subdirectorios por lo que es posible que al descomprimir los paquetes sucesivos nos pida confirmación para sobrescribir fusionándolo con el contenido existente.

A continuación lanzaremos Matlab y cambiaremos la carpeta de trabajo por la carpeta en la que hemos instalado la toolbox. Localizaremos y ejecutaremos el siguiente script para configurar la librería:

```
rvctools/startup_rvc.m
```

En este momento la librería está configurada y la ruta de búsqueda de Matlab puede localizar las funciones de la librería. También tendremos accesible la ayuda específica de esta librería dentro del visor de ayuda de Matlab, incluyendo el acceso a la documentación en pdf.

4.3 Funciones disponibles en la MVTB

A continuación se enumeran por categorías las funciones proporcionadas por la librería '*Machine Vision Toolbox*' (**MVTB**).

Esta librería suele utilizarse de forma conjunta con otras librería de procesamiento de imágenes como la "*Image Processing Toolbox*", por lo que aquellas funciones que se solapan tienen el prefijo **i** en el nombre en la **MVTB**.

Modelos de Cámara	
`Camera`	Superclase abstracta para definir modelos de cámaras
`CentralCamera`	Clase cámara perspectiva (Pin-hole)
`CatadioptricCamera`	Clase cámara catadióptrica
`FishEyeCamera`	Clase cámara ojo de pez
`SphericalCamera`	Clase cámara estérica
`camcald`	Calibración de una cámara a partir de puntos
`invcamcal`	Descomposición de un matriz de cámara

Tabla 4.3 Funciones modelos de cámara

Fuentes de Imagen	
`iread`	Lee una imagen de un fichero
`pnmfilt`	Lee un fichero de imagen a través de un programa externo PNN como ImageMagick y netpbm
`AxisWebCamera`	Adquiere una imagen desde una cámara web Axis
`Eartview`	Adquiere una imagen desde Google Earth
`ImageSource`	Superclase abstracta para dispositivos de captura
`Movie`	Adquiere imágenes desde un fichero de video
`YUV`	Clase para leer un fichero de video en formato YUV4MPEG
`mkcube`	Crea una imagen de un cubo
`mkgrid`	Crea una imagen con una rejilla de puntos
`testpattern`	Crea imágenes de test
Funciones adicionales de la librería general de Matlab	
`imread`	Lee una imagen de un fichero
`imwrite`	Escribe una imagen en un fichero

Tabla 4.4 Funciones fuentes de imagen

Funciones de manipulación de color y formato de imagen	
`colorspace`	Conversión entre espacios de color
`icolor`	Colorea un imagen en escala de grises
`igamm`	Corrección de color gammna
`imono`	Convierte una imagen en color en monocroma
`inormhist`	Normalización del histograma
`istretch`	Normalización de la imagen entre 0 y 1
`idouble`	Convierte una imagen a tipo double entre 0.0 y 1.0
`iint`	Convierte una imagen a tipo entero de 8 bits

Tabla 4.5 Funciones manipulación de color y formato de imagen

Operadores Espaciales	
Convolución Lineal	
icanny	Detección de bordes con el algoritmo de Canny
iconv	Correlación cruzada de dos imágenes (si una de las imágenes es una máscara de convolución implementa una convolución)
ismooth	Suavizado gaussiano
isobel	Detector de bordes de Sobel
radgrad	Gradiente radial
Máscaras	
kcircle	Máscara (kernel) circular
kdgauss	Máscara (kernel) derivada de la gaussiana
kdog	Máscara (kernel) diferencias de gaussianas
kgauss	Máscara (kernel) gaussiano
klaplace	Máscara (kernel) laplaciano
klog	Máscara (kernel) laplaciano gaussiano
ksobel	Máscara (kernel) Sobel
ktriangle	Máscara (kernel) triangular
No lineal	
dtransform	Transformación de distancia
irank	Filtro de rango (mín. máx, mediana)
ivar	Estadísticas del entorno de vecindad
iwindow	Operador espacial general
Morfología	
idilate	Dilatación morfológica
ierode	Erosión morfológica
iclose	Cerramiento o closing morfológico
iopen	Apertura u opening morfológica
imorph	Filtro morfológico general
hitormiss	Transformada Hit or Miss
ithin	Esqueletización morfológica
iendpoint	Busca los puntos finales de una imagen binaria esqueletizada
itriplepoint	Busca puntos triples (intersección de tres líneas)
morphdemo	Demostración de procesamiento morfológico
Similaridad	
imatch	Correspondencia de patrones
isimilarity	Localiza un patrón en la imagen
ncc	Correlación cruzada normalizada
sad	Suma de diferencias absolutas
ssd	Suma de diferencias al cuadrado
zncc	Correlación cruzada normalizada respecto a la media
zsad	Suma de diferencias absolutas respecto a la media
zssd	Suma de diferencias al cuadrado respecto a la media

Tabla 4.6 Funciones operaciones espaciales

Características de la Imagen	
Características de regiones	
RegionFeature	Clase base característica de una región
colorkmeans	Segmentación de región por color mediante clustering
ithres	Umbralización global (interactiva)
imoments	Momentos de primer y segundo orden de la imagen (RegionFeature)
ibbox	Busca el rectángulo envolvente
iblobs	Segmentación y características de objetos binarios conexos
igraphseg	Segmentación de la imagen basada en grafos
ilabel	Etiqueta una imagen con objetos binarios conexos
imser	Segmentación con el algoritmo “Maximal Stable Extremal Regions”
niblack	Umbralización adaptativa
otsu	Umbralización mediante el algoritmo de Otsu
Representación de bordes	
boundmatch	Correspondencia de perfiles de contorno
edgelist	Lista de pixels del borde de una región
Momentos	
humoments	Momentos de Hu
mpq	Momentos de la imagen de orden p,q
mpq_poly	Momentos de un polígono de orden p,q
upq	Momentos centrales de la imagen de orden p,q
upq_poly	Momentos centrales de un polígono de orden p,q
npq	Momentos centrales normalizados de la imagen de orden p,q
npq_poly	Momentos centrales normalizados de un polígono de orden p,q
Características de líneas	
Hough	Clase transformada de Hough
LineFeature	Clase características de línea
Características de puntos	
FeatureMatch	Clase correspondencia características
PointFeature	Clase base punto característico
ScalePointFeature	Clase punto característico con información e escala
SiftPointFeature	Clase punto característico SIFT
SurfPointFeature	Clase punto característico SURF
icorner	Detector de esquinas
iscalespace	Secuencia de imágenes a diferente escala
iscalemax	Detector puntos característico máximos en el espacio de escala
isift	Extractor características y descriptor SIFT (interfaz a la librería VLFeat)
isurf	Extractor características y descriptor SURF (interfaz a la librería OpenSurf)
Otras Características	
peak	Busca el pico en un vector
peak2	Busca el pico en una matriz
ihist	Histograma de la imagen
iprofile	Extrae pixels en una línea

Tabla 4.7 Funciones características de imágenes

Múltiples Vistas (3D)	
Geometría Proyectiva	
e2h	Coordenadas Euclídeas a Homogéneas
h2e	Coordenadas Homogéneas a Euclídeas
homeline	Línea homogénea a partir de dos puntos
Plucker	Clase coordenadas de Plucker
epidist	Distancia de un punto a la línea epipolar
epiline	Dibuja líneas epipolares
fmatrix	Calcula la matriz Fundamental
homography	Calcula una homografía
Estéreo	
istereo	Correspondencia estéreo
irectify	Rectifica un par de imágenes estéreo
stdisp	Muestra un par de imágenes estéreo
3D	
icp	Alineamiento de una nube de puntos
Ray3D	Clase Rayo 3D en el espacio
Secuencias de Imagen	
BagOfWords	Clase Bag of Words, captura características y compara imágenes
ianimate	Muestra una secuencia de imágenes
Tracker	Clase seguimiento de puntos en una secuencia de imágenes
Transformaciones	
homwarp	Transformación de imagen mediante una homografía
homtrans	Aplica una trasformación homogénea a puntos
idecimate	Reduce el tamaño/resolución de una imagen
ipad	Rellena el contorno de una imagen con valores NaN
ipyramid	Crea una pirámide a diferentes escalas con un suavizado gaussiano
ireplicate	Expande una imagen replicando pixels
iroi	Extrae una región de interés rectangular
irotate	Rota una imagen
isamsize	Recorte de imágenes para que tengan la misma dimensión
iscale	Imagen escalada espacialmente
itrim	Recorta una imagen

Tabla 4.8 Funciones múltiples vistas 3D

Utilidades	
Visualización de imágenes	
`idisp`	Herramienta visualización de imágenes
`idisplabel`	Visualiza una imagen con una máscara con los pixels etiquetados
`showpixels`	Muestra los pixels de una imagen de baja resolución
`imshow`	Función de visualización de imágenes de la librería general de Matlab
Manipulación de imágenes	
`iconcat`	Concatena imágenes
`ipaste`	Pega una imagen dentro de otra
`iisum`	Suma de la imagen integral
`intdimage`	Calcula la imagen integral
`col2im`	Convierte una imagen unidimensional en una bidimensional
`im2col`	Convierte una imagen bidimensional en una unidimensional
Dibujo	
`iline`	Dibuja una línea en una imagen
`plot_arrow`	Dibuja una flecha en una ventana
`plot_box`	Dibuja una caja en una ventana
`plot_circle`	Dibuja un círculo en una ventana
`plot_ellipse`	Dibuja una elipse en una ventana
`plot_homline`	Dibuja una línea homogénea en una ventana
`plot_point`	Dibuja un punto en una ventana
`plot_poly`	Dibuja un polígono en una ventana
`plot_sphere`	Dibuja una esfera en una ventana
General	
`about`	Información de una imagen
`bresenham`	Genera las coordenadas de los pixels de una línea
`closest`	Busca los puntos más cercanos de dos conjuntos de puntos de dimensión N
`colnorm`	Norma columna de una matriz
`distance`	Distancia Euclídea entre conjuntos de puntos
`filt1d`	Filtro de rango unidimensional (mín. máx, mediana)
`imeshgrid`	Similar a la función meshgrid pero con una imagen de entrada
`iscolor`	Comprueba si la imagen es en color
`isize`	Tamaño de la imagen
`kmeans`	Clustering K-means
`polydiff`	Derivada de un polinomio
`ransac`	Algoritmo Random Sample Consensus
`zcross`	Detector de pasos por cero
`xaxis`	Configura el escaldo en el eje X
`yaxis`	Configura el escaldo en el eje Y
`xyzlabel`	Pone etiquetas en los ejes X,Y,Z

Tabla 4.9 Funciones utilidades

4.4 Ejemplos de uso de la librería MVTB

Para ilustrar el manejo de la librería vamos a mostrar algunos ejemplos de procesamiento de imágenes utilizando las funciones de la librería incluyendo la carga de imágenes y la visualización de resultados.

En primer lugar, seleccionaremos como directorio de trabajo la carpeta con las imágenes de ejemplo de la librería **MVTB**:

```
rvctools/vision/images
```

4.4.1 Cargar y visualizar una imagen

La función para cargar el contenido de una imagen es **`iread`**. La imagen es devuelta como una matriz. Adicionalmente disponemos de una función de la librería general de Matlab para leer imágenes denominada **`imread`**. Existen varias modalidades en función de los parámetros indicados:

```
IM = iread()
```

Presenta una interfaz GUI para que el usuario seleccione la imagen

```
IM = iread([], OPTIONS)
```

Presenta una interfaz GUI para que el usuario seleccione la imagen y adicionalmente permite indicar opciones (tabla 4.7)

```
IM = iread(PATH, OPTIONS)
```

Es similar al caso anterior pero indicando la carpeta inicial, presenta una interfaz GUI para que el usuario seleccione la imagen y permite indicar opciones (tabla 4.10)

```
IM = iread(FILE, OPTIONS)
```

Es la versión más habitual, abre el fichero de imagen indicado en FILE. El segundo parámetro es opcional indicando opciones de formato (tabla 4.10)

Opciones	Descripción
`'uint8'`	Devuelve una imagen de 8 bits sin signo (0-255) en cada canal
`'single'`	Devuelve una imagen con valores en coma flotante precisión simple (0.0-1.0)
`'double'`	Devuelve una imagen con valores en coma flotante precisión doble (0.0-1.0)
`'grey'`	Convierte la imagen a escala de grises según ITU rec 601
`'grey_709'`	Convierte la imagen a escala de grises según ITU rec 709
`'gamma',G`	Aplica una corrección de color gamma, el segundo parámetro (G) puede ser numérico o 'sRGB'
`'reduce',R`	Reduce la resolución por R en las dos dimensiones
`'roi',R`	Aplica la región de interés R a la imagen. R=[umin umax; vmin vmax]

Tabla 4.10. Opciones función iread

Ejemplo: se carga una imagen en color y se muestra la información con la función **`about`**

```
>> im = iread('flowers4.png');
>> about im
```

```
im [uint8] : 426x640x3 (817.9 kB)
>>
```

Ejemplo: se carga una imagen en color convirtiéndola a escala de grises y se muestra la información con la función **about**:

```
>> im_g = iread('flowers4.png', 'gray');
>> about im_g
im [uint8] : 426x640 (272.6 kB)
>>
```

La función para visualizar el contenido de una imagen es **idisp**.:

```
idisp(IM, OPTIONS)
```

Ejemplo:

```
>> idisp(im)
```

Visualiza una imagen proporcionando varias herramientas para mostrar información de la misma (figura 4.3). Se trata de un gráfico Matlab por lo que se dispone de todas las opciones para guardar, editar, poner etiquetas, añadir textos, etc, ya comentadas en el capítulo 2. En la tabla 4.11 se describen las opciones más usuales del segundo parámetro de esa función, para un listado más completo se recomienda consultar la ayuda de la función (**help idisp**).

Podemos controlar las diferentes ventanas mediante el comando **figure** (**subplot** no funciona con la función **idisp),** y poner un título mediante el comando **title,** tal como se estudió en detalle en el capítulo 2. Si deseamos mostrar varia imágenes juntas una al lado de otra se deben pasar como una lista entre {} (ver figura 4.7)

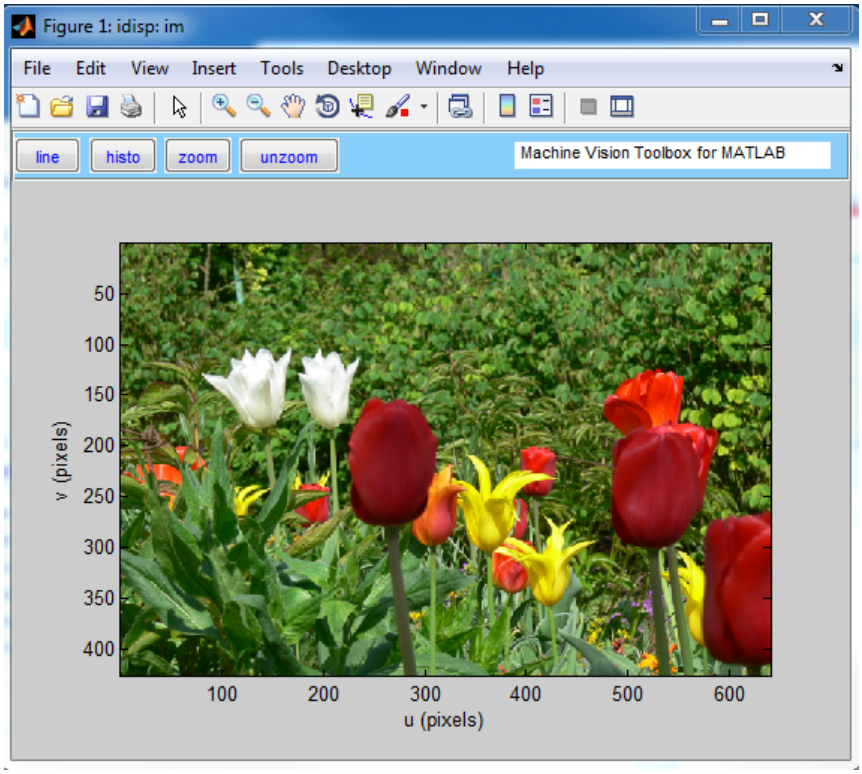


Figura 4.3 Ventana de visualización de imágenes con idisp

Cuando pulsamos en cualquier punto de la imagen se mostrará en recuadro superior derecho el valor del pixel en dicho punto (figura 4.4.b).

Opciones	Descripción
`'nogui'`	No muestra la interfaz gráfica GUI (botones, valor del pixel)
`'noaxes'`	No muestra los ejes
`'noframe'`	No muestra ni los ejes ni el marco exterior de la imagen
`'plain'`	No muestra ejes, marco exterior o GUI
`'flatten'`	Muestra separados los canales de una imagen en color
`'colomap',C`	Cambia el mapa de color de la imagen
`'new'`	Crea una nueva ventana para la imagen
`'title',T`	Muestra el titulo T en la barra superior de la ventana
`'bar'`	Añade una barra de color con los niveles de intensidad
`'clickfunc',F`	Llama a la función F(x,y) con la doble pulsación del botón del ratón
`'print',F`	Guarda la imagen en el fichero F en formato eps

Tabla 4.11. Opciones más usuales de la función idisp

Adicionalmente se disponen de cuatro botones que permiten acciones específicas sobre la imagen;

- **Zomm**/**unzoom**: permiten realizar un zoom de una región rectangular de la imagen elegida con el ratón. Pulsaremos en la esquina superior izquierda y arrastraremos el cursor hasta la esquina inferior derecha. (figura 4.4)
- **Histo**: muestra el histograma de la imagen (figura 4.5)
- **Line**: muestra un perfil de una línea marcada en la imagen (figura 4.6)

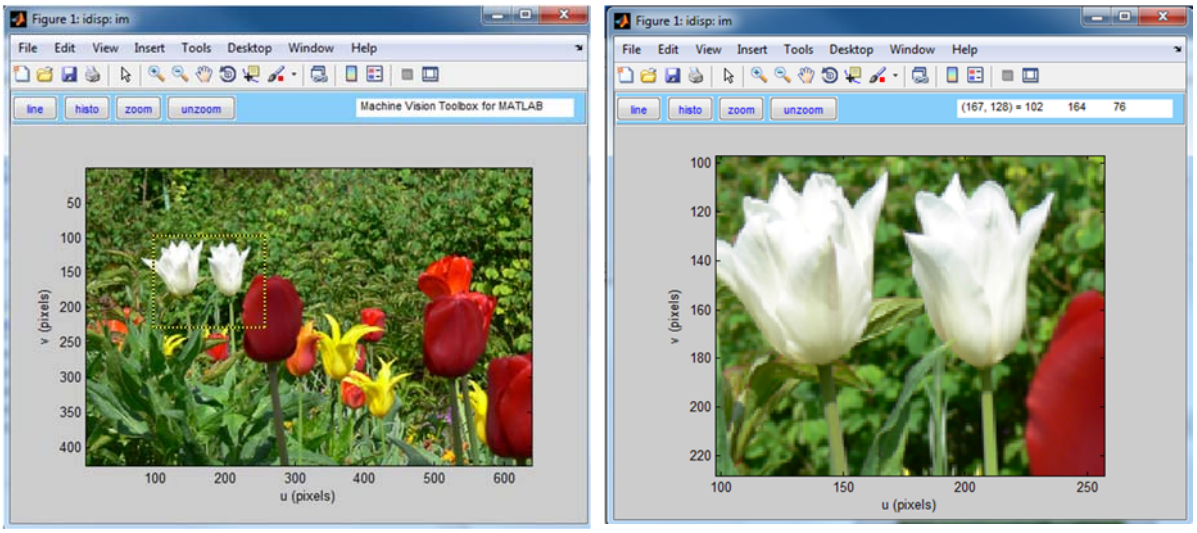


Figura 4.4 Botón de zoom. Visualización del valor de un pixel

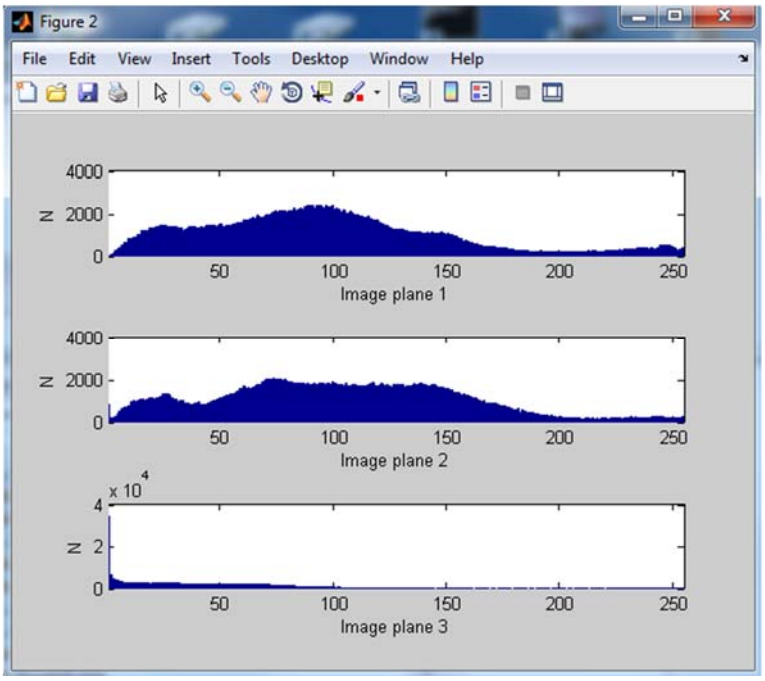


Figura 4.5 Histograma de la imagen

Ejemplo: visualización de una imagen en escala de grises, asignación de título y cálculo de un perfil (figura 4.6)

```
>> idisp(im_g)
>> title('Fichero: flowers4.png -> B/N')
```

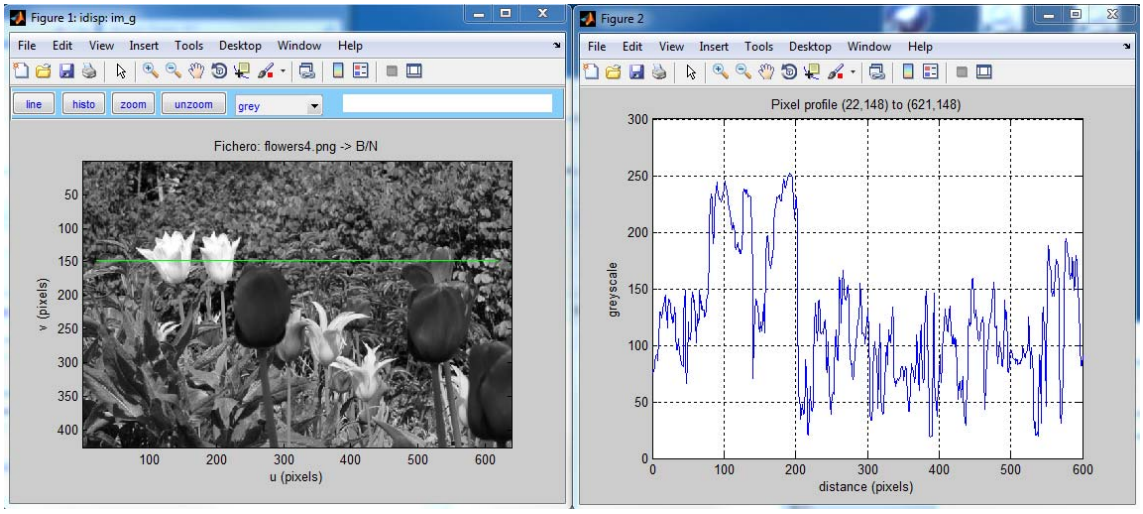


Figura 4.6 Perfil de una línea de la imagen

4.4.2 Procesamiento de imágenes

En este apartado veremos un ejemplo de uso de las funciones de procesamiento de imágenes y para realizar una segmentación por color. A continuación se muestra el código que presenta un ejemplo de conversión entre espacios de color, la separación de canales, la umbralización y el filtrado morfológico.

```
% directorio de trabajo    rvctools/vision/images

% lee la imagen
im = iread('flowers4.png');

% convierte la imagen de RGB a HSI
im_hsi = colorspace('RGB->HSI',im);

% extrae el canal de Hue
hue = im_hsi(:,:,1);

% visualiza la imagen y el canal de Hue
idisp({im, hue}, 'noframe', 'new','title', 'Imagen ::: Canal Hue');

% visualiza el histograma del canal de Hue
figure;
ihist(hue);
title('Hitograma Hue');

% umbraliza  el ángulo Hue en torno a 0º+-15º
seg = (hue<15 | hue>345)*255;

% visualiza la imagen y la segmentación
idisp({im, seg}, 'noframe', 'new','title', 'Imagen ::: Segmentación');

% filtro morfológico
kernel = [ 0 1 0; 1 1 1; 0 1 0];   % Elemento estructurante en cruz
seg = iopen(seg, kernel);   % elimina pixels sueltos
seg = iclose(seg, kernel);  % elimina huecos

% visualiza la imagen y la segmentación filtrada
idisp({im, seg}, 'noframe', 'new', 'title', 'Imagen ::: Segmentacion
Filtrada');
```

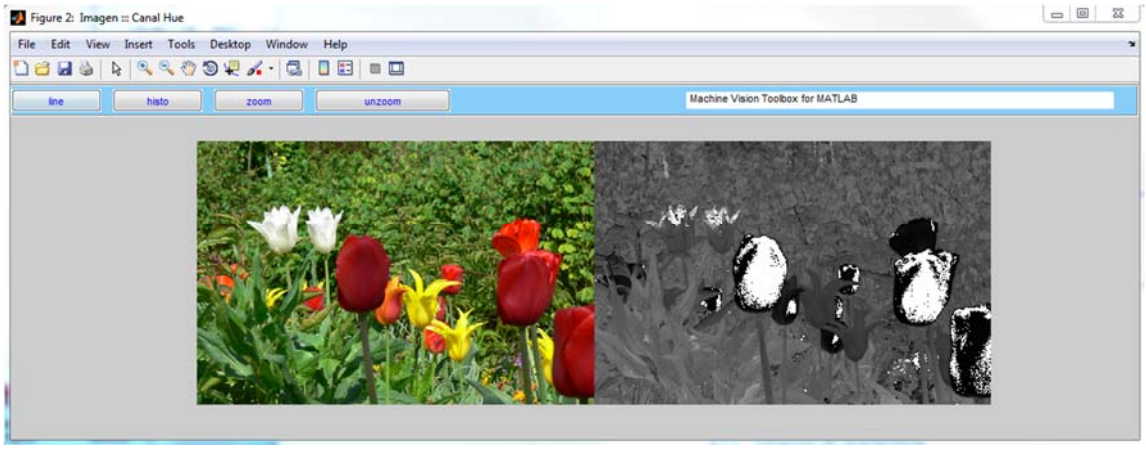


Figura 4.7 Cambio al espacio de color HSI (canal de Hue)

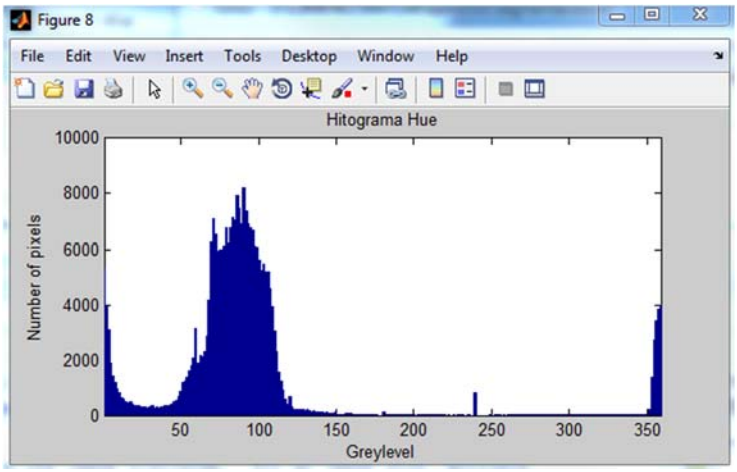


Figura 4.8 Histograma Canal Hue. Función ihisto

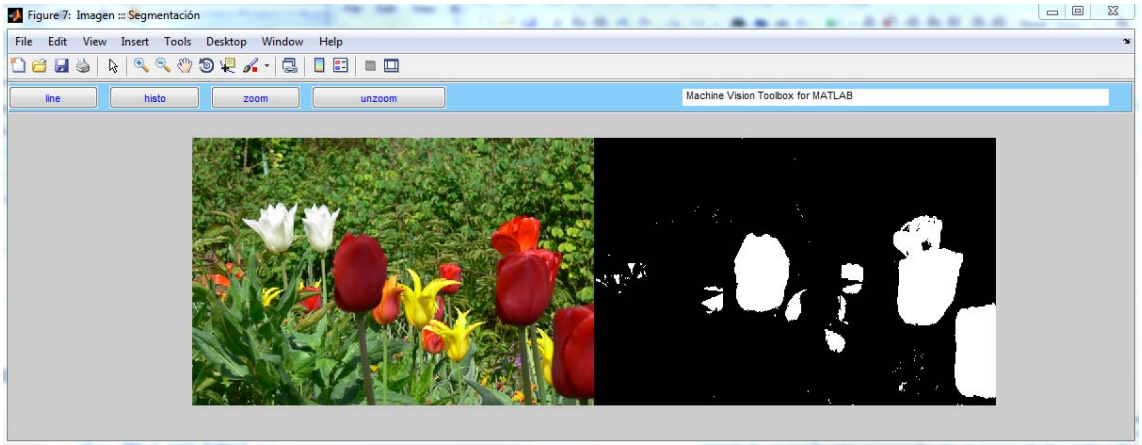


Figura 4.9 Segmentación del canal de Hue por umbralización

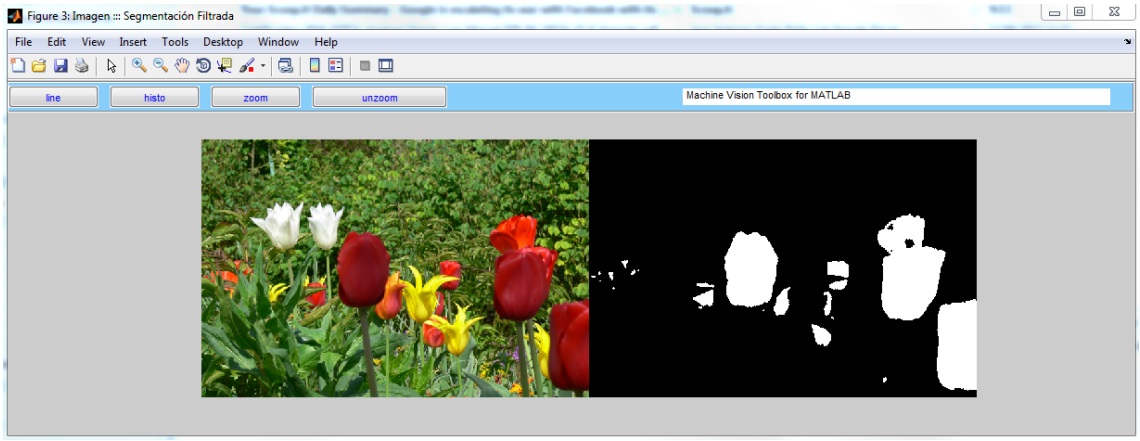


Figura 4.10 Segmentación del canal de Hue por umbralización con filtro morfológico

A continuación se analizan en detalle las funciones utilizadas:

a) Conversión del espacio de color:

```
B = colorspace(S,A)
```

Transforma una imagen en color entre dos representaciones (espacios) de color. **A** es la imagen de entrada, **B** es la imagen resultado. **S** es una cadena que indica el tipo de transformación a realizar con cualquiera de las siguientes sintaxis:

```
'EspacioEntrada->EspacioSalida'

'EspacioSalida<-EspacioEntrada'
```

Los diferentes tipos de espacios de color con sus códigos están indicados en la tabla 4.12.

En el ejemplo la transformación realizada se realizada entre el espacio RGB y el espacio HSI por lo que la cadena de conversión es:

```
'RGB->HSI'
```

Opciones	Descripción
'RGB'	sRGB IEC 6966-2-1
'YCbCr'	Luma + Chroma (versión digitalizada de Y'PbPr)
'JPEG-YCbCr'	Luma + Chroma espacio utilizado en JFIF JPEG
'YDbDr'	SECAM Y'DbDr Luma + Chroma
'YPbPr'	Luma (ITU-R BT.601) + Chroma
'YUV'	NTSC PAL Y'UV Luma + Chroma
'YIQ'	NTSC Y'IQ Luma + Chroma
'HSV' o 'HSB'	Hue, Saturation Value/Brightness
'HSL' o 'HLS'	Hue Saturation Luminance
'HSI'	Hue Saturation Intensity
'XYZ'	CIE 1931 XYZ
'Lab'	CIE 1976 L*a*b* (CIELAB)
'Luv'	CIE L*u*v* (CIELUV)
'LCH'	CIE L*C*H* (CIELCH)
'CAT2 LMS'	CIE CAT02 LMS

Tabla 4.12. Opciones de espacios de color de la función **colorspace**

b) Visualización de imágenes:

Se muestra el uso de la función **`idisp`** con opciones y mostrando dos imágenes juntas usando el operador lista {} (figura 4.7).

```
idisp({im, hue}, 'noframe', 'new','title', 'Imagen ::: Canal Hue');
```

c) Visualización del histograma: (figura 4.8)

El canal de Hue utiliza coordenadas angulares circulares por lo varían entre 0-359º. El cálculo y visualización del histograma se realiza con la función **`ihist`** que presenta varias opciones de uso:

```
ihist(IM, OPTIONS)
```

Visualiza en una ventana el histograma de la imagen IM. El campo OPTIONS es opcional y permite modificar la configuración del histograma según la tabla 4.13

```
H = ihist(IM, OPTIONS)
```

Calcula y devuelve el histograma de la imagen IM. El campo OPTIONS es opcional y permite modificar la configuración del histograma según la tabla 4.13

```
[H,X] = ihist(IM, OPTIONS)
```

Como en el caso anterior pero devuelve las coordenadas de cada valor de intensidad en X.

Opciones	Descripción
`'nbins'`	No muestra la interfaz gráfica GUI (botones, valor del pixel)
`'cdf'`	Calcula el histograma acumulado
`'normcdf'`	Calcula el histograma acumulado normalizado (0-1)
`'sorted'`	Probabilidad de ocurrencia ordenada por magnitud descendente. Los valores de cada elemento se indican en X

Tabla 4.13. Opciones de la función **ihisto**

d) Umbralización: (figura 4.9)

La umbralización se realiza utilizando las operaciones de cálculo matricial disponible en Matlab. Cuando realizamos una operación de comparación de una matriz con un escalar, nos devuelve otra matriz en la que están a 1 los elementos de la matriz que cumple la comparación y 0 en caso contrario. Igualmente ocurre con las operaciones lógicas OR (|) y AND (&). Una vez obtenido la matriz con los valores lógicos (0 o 1), se multiplica por 255 para obtener una imagen binaria: blanco (255) y negro (0).

```
seg = (hue<15 | hue>345)*255;
```

e) Filtrado morfológico: (figura 4.10)

```
OUT = iclose(IM, SE, OPTIONS)

OUT = iopen(IM, SE, OPTIONS)
```

Realizan un filtrado morfológico (closing/opening) a la imagen IM con el elemento estructurante SE. En el primer caso una dilatación seguida de una erosión, en el segundo una erosión seguida de una dilatación. La tabla 4.14 indica las opciones.

```
OUT = iclose(IM, SE, N, OPTIONS)

OUT = iopen(IM, SE, N, OPTIONS)
```

Igual que en el caso anterior pero las erosiones y dilataciones se aplican N veces.

Opciones	Descripción
'norder'	El valor para los pixels fuera de la imagen se obtiene replicando el del borde (valor por defecto)
'none'	Los pixels fuera de la imagen no se incluyen en la ventana
'trim'	No se calcula el filtro para los pixels en los que la ventana cae fuera de la imagen
'wrap'	Los bordes se consideran circulares (arriba-abajo, izquierda-derecha)

Tabla 4.14. Opciones filtros morfológicos

4.4.3 Extracción de características

En este apartado veremos un ejemplo completo se extracción de puntos característicos junto a sus descriptores en dos imágenes estableciendo la correspondencia entre los puntos de ambas imágenes. A continuación se muestra el código que presenta un ejemplo de detección de esquinas con el operador de Harris.

```
% directorio de trabajo    rvctools/vision/images
% lee las imágenes
im1 = iread('seq/im.001.png');
im2 = iread('seq/im.002.png');

% visualiza las imágenes
figure(1); hold off;
idisp({im1, im2}, 'noframe', 'title', 'Imagenes');

% convierte las imágenes a escala de grises
im1 = imono(im1);
im2 = imono(im2);

% extrae puntos característicos
sf1 = icorner(im1, 'nfeat', 100);
sf2 = icorner(im2, 'nfeat', 100);

% dibuja los puntos característicos en la primera imagen
for i=1:size(sf1,2)
    plot_box('centre', sf1(i).uv()', 'size', [10 10], 'r');
end

% dibuja los puntos característicos en la segunda imagen
desp = size(im1,2);  % desplazamiento horizontal de la segunda imagen
for i=1:size(sf2,2)
    plot_box('centre', sf2(i).uv()'+[desp 0], 'size', [10 10], 'b');
end

% correspondencia de puntos
[m,C] = sf1.match(sf2);
m.plot('g');
```

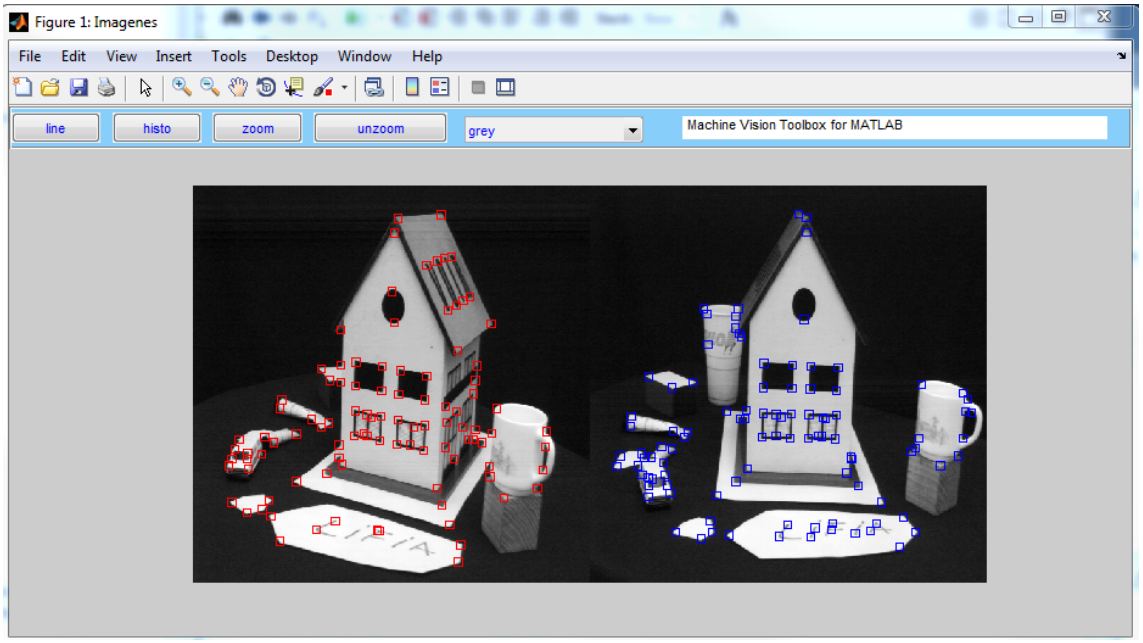


Figura 4.11 Detección de esquinas con la función **icorner**

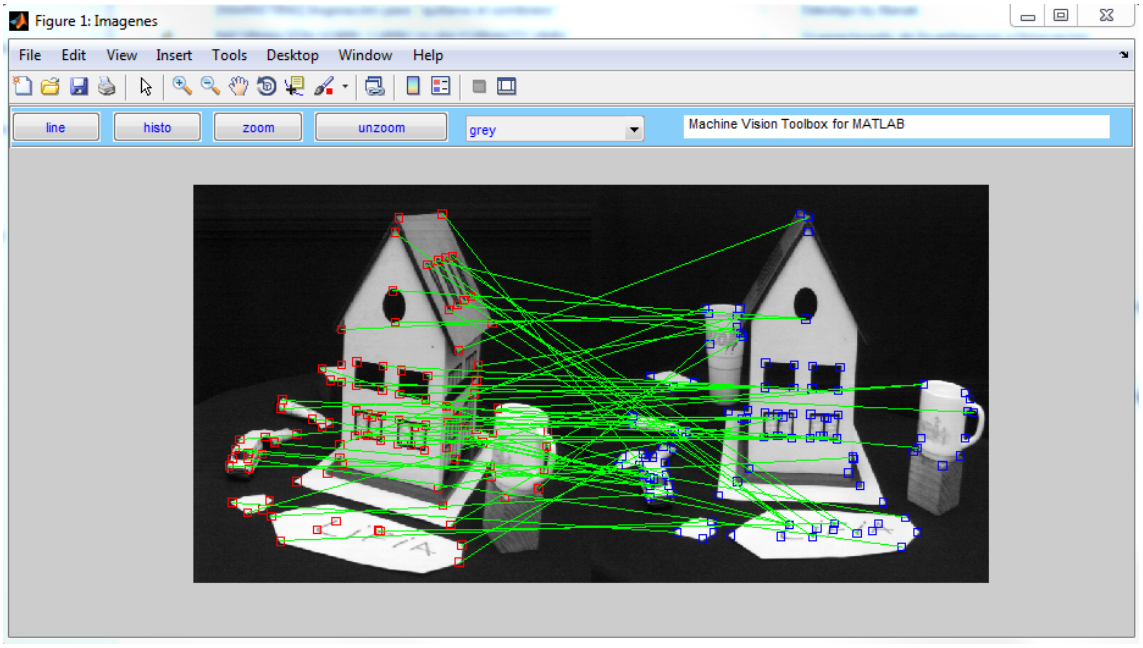


Figura 4.12 Correspondencia entre puntos **FeatureMatch**

A continuación se analizan en detalle las funciones utilizadas:

a) Clase base **PointFeature**

Los puntos característicos son almacenados como objetos de la clase **`PointFeature`**. Los métodos y propiedades de esta clase se detallan en la tabla 4.15.

Componente	Descripción
Métodos:	
`plot()`	Dibuja un recuadro en la posición del punto característico
`distance(f2)`	Distancia entre el descripto del punto actual y el punto f2
`ncc(f2)`	Similaridad entre el descripto del punto actual y el punto f2
`uv()`	Devuelve las coordenadas como un vector columna [u; v]
`display()`	Muestra una descripción del punto característico
`char()`	Convierte el valor a un cadena de caracteres
`match(F2,options)`	Correspondencia entre el punto actual y el vector de puntos punto F2
Propiedades:	
u	Coordenada horizontal
v	Coordenada vertical
`strength`	Intensidad de la característica
`decriptor`	Vector de descripción

Tabla 4.15. Componentes de la clase PointFeature

b) Detección de esquinas:

```
f = icorner(im, options)
```

Devuelve un vector de características de tipo **`PointFeature`**. Por defecto el detector utilizado es el de Harris pero mediante el campo de opciones pueden seleccionarse otros algoritmos (ver manual MVTB). También pude limitarse el nuúmero de características devueltas mediante la opción **`'nfeat',NF`** . El vector de descripción es muy sencillo (tres componentes) y almacena el tensor con las derivadas direccionales `[Ix* Iy* Ixy*]`.

c) Visualización de los puntos característicos: (figura 4.11)

Se utilizan las funciones de dibujo incluidas en la librería. En el caso de dibujo de rectángulos se proporcionan varias opciones de especificar las coordenadas. Las coordenadas del centro del punto son obtenidas usando el método **`uv()`** del objeto `PointFeature`.

```
plot_box(B, LS)
```
B=[XL XR; YL YR] LS: estilo de línea

```
plot_box(X1,Y1, X2,Y2, LS)
```

```
plot_box('centre', P, 'size', W, LS)
```
Centro P=[X,Y] W=[WIDTH HEIGHT]

```
plot_box('topleft', P, 'size', W, LS)
```
Esquina superior izquierda at P=[X,Y] W=[WIDTH HEIGHT].

d) Clase **`FeatureMatch`**:

Las correspondencias entre puntos son almacenadas como objetos de la clase **`FeatureMatch`**. Los métodos y propiedades de esta clase se detallan en la tabla 4.16.

Componente	Descripción
Métodos:	
`plot()`	Dibuja una recta uniendo las coordenadas de los puntos correspondientes En la ventana debe dibujarse previamente las imágenes como una lista {im1,im2}
`show()`	Muestra un resumen de las estadísticas de la correspondencia
`ransac(func,opt)`	Determina inliers y outliers según el modelo definido por func
`inlier()`	Devuelve las correspondencias correctas (inliers)
`outlier()`	Devuelve las correspondencias incorrectas (outliers)
`subset()`	Devuelve un subconjunto de las correspondencias
`display()`	Muestra una descripción de una correspondencia
`char()`	Convierte el valor a un cadena de caracteres
Propiedades:	
`p1`	Coordenadas del punto en la vista 1 (2x1)
`p2`	Coordenadas del punto en la vista 2 (2x1)
`p`	Coordenadas del punto en las vistas 1 y 2 (4x1)
`distance`	Distancia entre los vectores de descripción.

Tabla 4.16. Componentes de la clase FeatureMatch

e) Correspondencia de puntos:

La correspondencia entre vectores de puntos característicos se realiza mediante el método **`match()`** asociado a un vector de puntos característicos (**`PointFeature`**)

```
m = F.match(F2, options)
```

Devuelve un vector de correspondencias (**`FeatureMatch`**) entre puntos de los vectores F y F2. Almacena en m las coordenadas y la distancia.

```
[m,C] = F.match(F2, options)
```

Como la anterior pero devuelve una mariz C adicional que almacena en cada columna los índices de las correspondencias para cada punto de la primera imagen con la segunda.

Opciones:

`'thresh', T` Umbral de correspondencia (0.05 por defecto)

`'median'` Umbral en la mediana de la distancia

En el ejemplo el código utilizado es:

```
% correspondencia de puntos
[m,C] = sf1.match(sf2);
m.plot('g');
```

El método **`plot()`** asociado al vector de correspondencias visualiza sobre las dos imágenes una línea entre cada par de coordenadas correspondientes. Este método admite como parámetro un cadena de formato con la misma sintaxis usada en la función **`plot()`** de Matlab vista en el capítulo 2.

A continuación se muestra el primer componente del vector de correspondencias (**`m`**) y la matriz de correspondencias (**`C`**). Esta última matriz almacena en cada columna los índices de los puntos correspondientes en cada uno de los dos vectores de puntos.

```
>> m(1)
ans =
(243, 298) <-> (223, 325), dist=134.406036

>> C(:,1)
ans =
   41
   54
>>
```