Práctica 5
Descripción y Reconocimiento de regiones

- Ejemplo de cálculo del Convex Hull de una nube de puntos
- Practica de cálculo de Área minima

Robótica y Visión por Computador
Ing. Telecomunicaciones

Universidad Miguel Hernández

Objetivo de la práctica:

El objetivo de la práctica es profundizar en las técnicas de reconocimiento y descripción de regiones en visión artificial utilizando las librerías OpenCv. Como sabemos, los diferentes descriptores estudiados tienen como finalidad extraer las características de un objeto y tratar de describir matemáticamente alguna de sus características.

Dentro de los descriptores de contorno estudiados podemos encontrar técnicas de buscar la envolvente convexa (convex hull) de un objeto. En la presente práctica se muestra como ejemplo (extraído de los ejemplos de OpenCv) cómo se puede calcular el convex hull de una nube de puntos generada de forma aleatoria.

De forma análoga a la técnica mostrada, se pueden utilizar las librerías OpenCv para obtener una descripción matemática de una determinada región. En la práctica se propone calcular el rectángulo y círculo de mínima área que encierre a la anterior nube de puntos.

Por tanto, la presente práctica se estructura en dos partes:

- Un ejemplo de cálculo del convex hull (envolvente convexa) de una nube de puntos
- Programación por parte del alumno de un ejemplo de cálculo del área mínima de la nube de puntos anterior.

En la práctica no se utiliza ninguna imagen fuente, sino que se genera una imagen con nubes de puntos aleatorias.

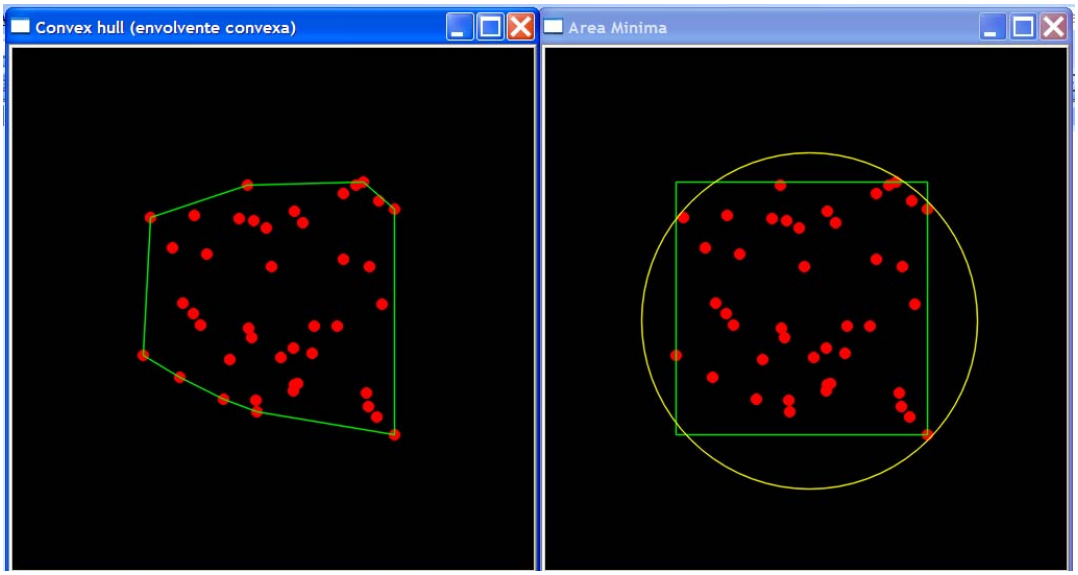
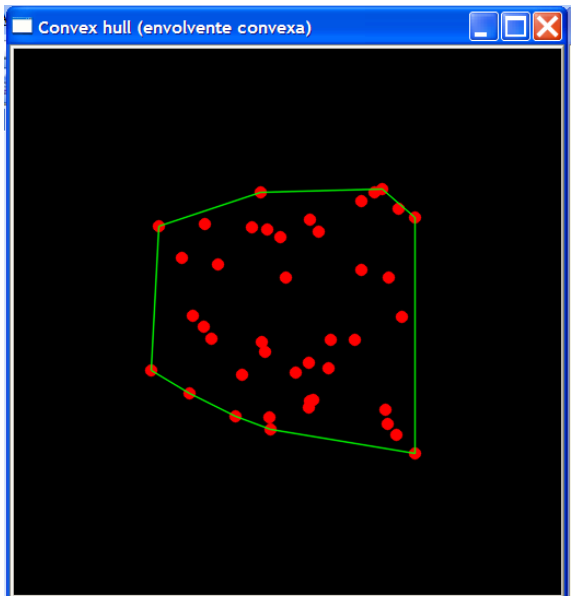


Imagen de la envolvente conexa y del cálculo de área mínima de una nube de puntos

Realización de la práctica:

1. En primer lugar se debe abrir el workspace creado en la práctica 0 o bien generar uno nuevo tal y como se explicó en dicha práctica.
2. Añadimos un nuevo proyecto al workspace, que llamaremos **practica5**
   a. Si estamos trabajando sobre el workspace anterior, los proyectos `cv.dsp, cvaux.dsp` y `highgui.dsp` ya están añadidos, y lo único que debemos hacer es establecer las dependencias y los settings del nuevo proyecto creado.
   b. Si hemos creado un proyecto nuevo, se deben añadir los proyectos `cv.dsp, cvaux.dsp` y `highgui.dsp` tal y como se explicó en la práctica 0 y luego establecer la configuración adecuada.
   c. Desde el explorador de Windows, guardamos el fichero `practica5.c` de la web y lo añadimos al proyecto actual.
3. Compilamos el ejemplo para obtener una figura similar a la siguiente.



4. Estudiamos con detalle el código. En especial se debe comprender …

   a. Como se ha generado la imagen,

```
IplImage* img = cvCreateImage( cvSize( 500, 500 ), 8, 3 );
cvNamedWindow( "Convex hull (envolvente convexa)", 1 );

  int i, count = rand()%100 + 1, hullcount;
  CvPoint pt0;
  CvPoint* points = (CvPoint*)malloc( count * sizeof(points[0]));
…
        for( i = 0; i < count; i++ )
        {
            pt0.x = rand() % (img->width/2) + img->width/4;
            pt0.y = rand() % (img->height/2) + img->height/4;
            points[i] = pt0;
        }
```

b. Cómo se utiliza la función `cvConvexHull2`. En especial cómo se utilizan las matrices `pointMat` y `hullMat` para describir los puntos generados y almacenar los datos de la envolvente convexa respectivamente.

```
CvMat pointMat = cvMat( 1, count, CV_32SC2, points );
CvMat hullMat = cvMat( 1, count, CV_32SC1, hull );

cvConvexHull2( &pointMat, &hullMat, CV_CLOCKWISE, 0 );
```

c. Cómo se genera la imagen a visualizar (`img`), tanto cómo se generan los puntos de la nube como cómo se generan las líneas de la envolvente, tomando como punto inicial `pt0`.

```
cvZero( img );
for( i = 0; i < count; i++ )
{
    pt0 = points[i];
    cvCircle( img, pt0, 5, CV_RGB( 255, 0, 0 ), CV_FILLED, CV_AA, 0 );
}
pt0 = points[hull[hullcount-1]];

for( i = 0; i < hullcount; i++ )
{
    CvPoint pt = points[hull[i]];
    cvLine( img, pt0, pt, CV_RGB( 0, 255, 0 ), 1, CV_AA, 0 );
    pt0 = pt;
}
```

5. Sobre el fichero anterior añadir los comandos necesarios para calcular el área mínima que encierra esa nube de puntos. Para ello utilizaremos las funciones `cvMinAreaRect2` y `cvMinEnclosingCircle` para utilizar un rectángulo y un círculo. Para ello se deben seguir los siguientes pasos:
   a. Crear una nueva ventana gráfica para visualizar la imagen
   b. Crear las variables necesarias para las funciones que vamos a utilizar.
   c. Utilizar las funciones `cvMinAreaRect2` y `cvMinEnclosingCircle.`
   d. Visualizar los puntos de la nube
   e. Para visualizar el rectángulo obtenido en `cvMinAreaRect2` se recomienda utilizar la función `cvBoxPoints` que encuentra los vértices de la caja. De esta forma se puede utilizar la función `cvLine` para utilizando un punto inicial, dibujar las líneas del rectángulo.

      NOTA: para poder utilizar los datos `double` con los que trabaja la función `cvBoxPoints` en la función `cvLine` se recomienda utilizar la función `cvRound` para cada componente de la estructuras `cvPoint`.

   f. Para visualizar el circulo obtenido en `cvMinEnclosingCircle` se utilizará la función `cvCircle`.

6. El resultado debe tener el aspecto mostrado en la figura inicial. Observar cómo con cada pulsación de *return* se modifica la nube de puntos aleatoria, de forma que para finalizar la ejecución del programa se debe pulsar la tecla "*Q*".