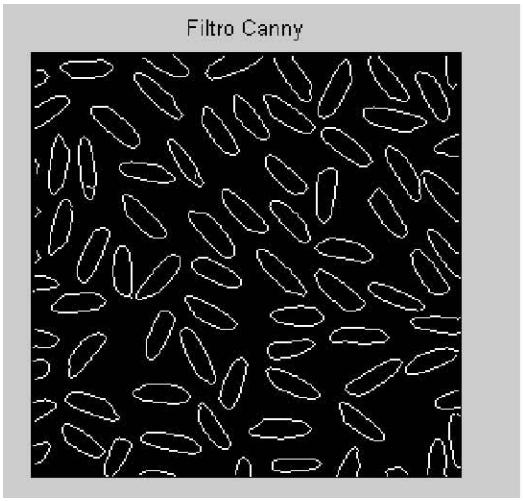
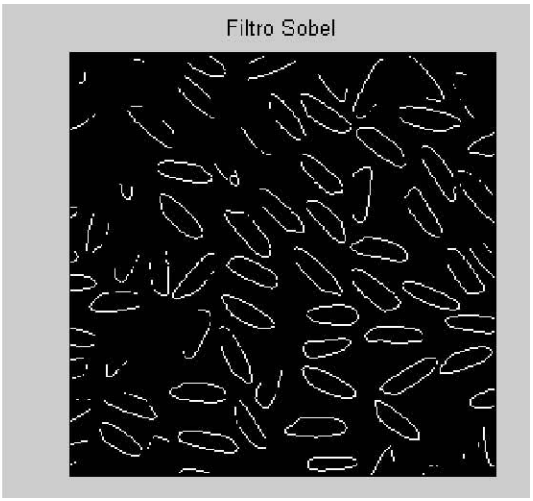
1. SEGMENTACIÓN BASADA EN BORDES

1.2. Localización de bordes

La función *edge* proporciona un número de estimadores derivativos, cada uno de los cuales implementa una de las definiciones indicadas anteriormente. Se puede elegir entre dos métodos de detección que obtienen buenos resultados: Canny y Sobel.

```
I=imread('rice.tif');        %Imagen de intensidad de tipo uint8
BW1 = edge(I, 'sobel');  %Imagen binaria (0 o 1) de bordes tras filtro de sobel
BW2 = edge(I,'canny');  %Imagen binaria de bordes tras filtro de canny
figure, imshow(BW1); title('Filtro Sobel');
figure, imshow (BW2); title ('Filtro Canny');
```





1.3. Análisis local

Se comienza detectando los bordes de la imagen mediante alguna de las técnicas estudiadas (típicamente pasar un filtro de Canny o de Sobel).

```
I=imread('rice.tif');
figure, imshow(I), title('Imagen original');

%BWs = edge(I, 'sobel');
%BWs = edge(I, 'canny');
BWs = edge(I, 'sobel', (graythresh(I) * .1));
%Detectamos bordes con la funcion edge, especificando la sensibilidad espectral del metodo
%sobel para mejorar la detección de bordes. Esta funcion ignora todos los bordes que no son
tan %fuertes como los especificados en la funcion graythresh
figure, imshow(BWs), title('Imagen de bordes');
```

A continuación, con alguna técnica local (analizando un entorno próximo alrededor de cada píxel) se intentan cerrar contornos (uniendo píxeles de los bordes que se hayan podido separar en los procesos previos, en el proceso de digitalización de la imagen, a

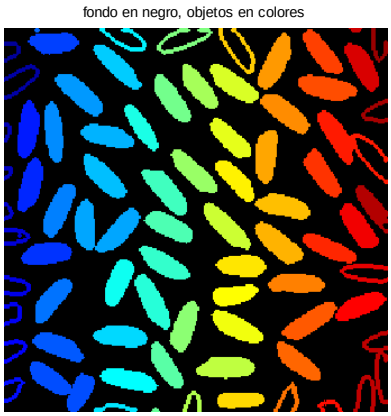
causa del ruido etc.). **Nota:** los operadores morfológicos como la *dilatación* y otros serán vistos en detalle en el próximo tema. Ahora sólo nos interesa experimentar su utilidad.

```
%Dilatamos para unir bordes y cerrar contornos
se=[0 1 0; 1 1 1; 0 1 0]; %Elemento estructurante usado para dilatar
BWsdil = imdilate(BWs, se);
figure, imshow(BWsdil), title('Imagen dilatada');
```

Después llenamos los contornos y segmentamos a partir de los mismos

```
%Segmentación por llenado de contornos (se rellenan los contornos porque no se desea, en este
%caso, segmentar otros objetos que haya dentro de un contorno ya cerrado)
BWrellenada=bwfill(BWsdil,'holes');
figure, imshow(BWrellenada), title('Imagen rellenada');

%Segmentación directa a partir de bordes.
L = bwlabel(BWrellenada, 4); %Imagen etiquetada.
numero_objetos = max(max(L)) %número de objetos igual a etiqueta mayor
map=[0 0 0; jet(numero_objetos)];
figure, imshow((L+1), map), title('fondo en negro, objetos en colores');
```



Observe (viendo las figuras a tamaño máximo) como la imagen de bordes (BWs) tenía algunos bordes pertenecientes al mismo objeto sin conectar (que hubieran dado lugar a objetos diferentes) y como en la imagen dilatada (BWsdil) esos bordes se unen. Fíjese también en que el proceso local de conexión de bordes mediante dilatación origina algún que otro efecto no deseado: hay algunos objetos que se unen. El análisis local no está carente de problemas pues, a veces, unas cosas se mejoran, pero otras empeoran.

En general, el paso inicial de detección de bordes, mediante filtros paso alto, es crucial. Si está bien hecho, el paso posterior de análisis de bordes para cerrar los contornos se facilita enormemente e incluso podría llegar a desaparecer. Observe, por ejemplo, en el programa anterior:

- qué ocurriría si en vez de la línea:

  ```
  BWs = edge(I, 'sobel', (graythresh(I) * .1) );
  ```

  utiliza la línea:

  ```
  BWs = edge(I, 'sobel');
  ```

- qué ocurriría si en vez de hacer un análisis local basado en una dilatación utiliza otro tipo de técnicas para cerrar contornos (no hacer nada, hacer un "closing"…)

1.4. Análisis global

Se pueden cerrar los contornos analizando la imagen global en busca de las rectas predominantes, círculos, elipses…

Ejecute el siguiente código (extraído de un ejemplo de Matlab) que utiliza la transformada de Hough para detectar segmentos en la imagen:

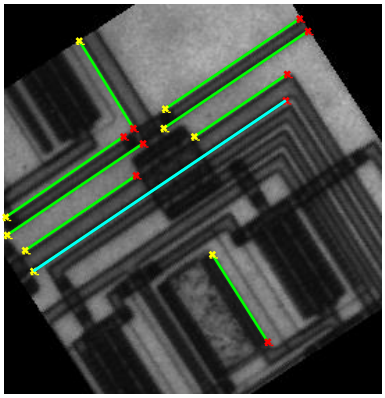
```
%Search for line segments in an image and highlight the longest segment.
close all, clear all
 I  = imread('circuit.tif');
rotI = imrotate(I,33,'crop');
BW = edge(rotI,'canny');
[H,T,R] = hough(BW);
imshow(H,[],'XData',T,'YData',R,'InitialMagnification','fit');
xlabel('\theta'), ylabel('\rho');
axis on, axis normal, hold on;
P  = houghpeaks(H,5,'threshold',ceil(0.3*max(H(:))));
%En vez de cinco picos pongo que encuentre más (pongo un número alto
%para que encuentre todos los picos mayor del umbral)
%P  = houghpeaks(H,1000,'threshold',ceil(0.3*max(H(:))));
x = T(P(:,2)); y = R(P(:,1));
plot(x,y,'s','color','white');

% Find lines and plot them
lines = houghlines(BW,T,R,P,'FillGap',5,'MinLength',7);
%lines = houghlines(BW,T,R,P,'FillGap',5,'MinLength',20);
figure, imshow(rotI), hold on
max_len = 0;
for k = 1:length(lines)
 xy = [lines(k).point1; lines(k).point2];
 plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');

 % plot beginnings and ends of lines
 plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
 plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');

 % determine the endpoints of the longest line segment
 len = norm(lines(k).point1 - lines(k).point2);
 if ( len > max_len)
   max_len = len;
   xy_long = xy;
 end
end

% highlight the longest line segment
plot(xy_long(:,1),xy_long(:,2),'LineWidth',2,'Color','cyan');
```



Si tiene dudas, mire la ayuda en las funciones *hough, houghpeaks y houghlines* de Matlab, que a "grosso modo" calculan la Transformada de Hough (obteniendo la tabla de incidencias de acumuladores de módulo *P* y ángulo θ), obtienen los picos o máximos en la misma (con determinadas restricciones), y detectan líneas (uniendo segmentos) con unas determinadas especificaciones, respectivamente.

Observe qué ocurriría si en vez de poner que se detecten cinco picos en la tabla de incidencias de acumuladores pone que se encuentre más picos (o un número alto, 1000 por ejemplo, para que se encuentren todos los picos mayores del umbral), o varía el umbral, ceil(0.3*max(H(:))), para detectar el pico:

```
P  = houghpeaks(H,1000,'threshold',ceil(0.3*max(H(:))));
```

Compruebe asimismo qué pasaría si varía el número de píxeles no unidos entre segmentos pertenecientes a la misma recta que se unen (formando un único segmento en lugar de dos), y si varía la longitud mínima del segmento para considerarse como tal; es decir varíe los valores 5 y 20 de 'FillGap', y 'MinLength', respectivamente, en:

```
lines = houghlines(BW,T,R,P,'FillGap',5,'MinLength',20);
```