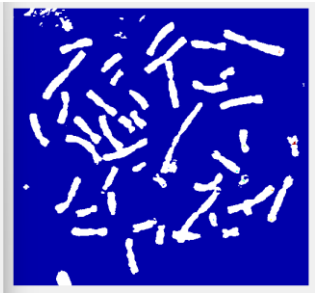
Práctica 9

Segmentación

Describe un algoritmo que determine el número de objetos de una imagen, como se muestra a continuación, prueba con otra imagen:

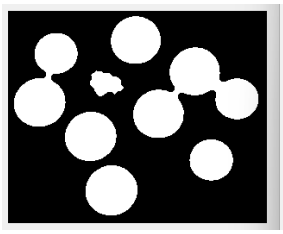


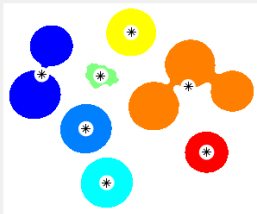
```
% Lee la imagen 'rice.tif' y la almacena en la variable I
I = imread('CHROMOS2.TIF');

% Calcula el umbral de binarización utilizando el método de Otsu
nivel = graythresh(I);

% Convierte la imagen en escala de grises a una imagen binaria utilizando el umbral calculado
B = im2bw(I, nivel);

% Crea una nueva figura y muestra la imagen binaria B
figure, imshow(B);

% Muestra información sobre las variables en el espacio de trabajo
whos;

% Etiqueta los componentes conectados en la imagen binaria B
% 'etiquetas' es una matriz con las etiquetas de los componentes conectados
% 'nobjetos' es el número de objetos conectados encontrados
[etiquetas, nobjetos] = bwlabel(B, 4);

% Muestra el número de objetos conectados detectados en la imagen
nobjetos;

% Convierte la matriz de etiquetas a una imagen RGB donde cada etiqueta
% tiene un color diferente, para visualización
E = label2rgb(etiquetas);

% Crea una nueva figura y muestra la imagen RGB con los objetos etiquetados
figure, imshow(E);
```

El algoritmo por la consola de comandos devuelve un total de 12 objetos detectados.

Realiza los siguientes apartados:

a) **Determina las componentes conectadas de ESTA IMAGEN y el centro de cada una de las regiones correspondientes.**





```
% Lee la imagen 'coins.png' y la almacena en la variable I
I = imread('coins.png');

% Aplica un filtro de mediana para reducir el ruido
I = medfilt2(I);

% Calcula el umbral de binarización utilizando el método de Otsu
nivel = graythresh(I);

% Convierte la imagen en escala de grises a una imagen binaria utilizando el umbral calculado
B = im2bw(I, nivel);

% Realiza operaciones morfológicas para mejorar la segmentación
B = imopen(B, strel('disk', 2)); % Eliminación de ruido pequeño
B = imclose(B, strel('disk', 5)); % Unión de partes cercanas

% Crea una nueva figura y muestra la imagen binaria B
figure, imshow(B);

% Etiqueta los componentes conectados en la imagen binaria B
[etiquetas, nobjetos] = bwlabel(B, 8);

% Convierte la matriz de etiquetas a una imagen RGB donde cada etiqueta
% tiene un color diferente, para visualización
E = label2rgb(etiquetas);

% Crea una nueva figura y muestra la imagen RGB con los objetos etiquetados
figure, imshow(E);

% Determinación del centro de cada objeto conectado
for k = 1:nobjetos
    % Encuentra las filas y columnas de los píxeles que pertenecen al objeto k
    [fila, col] = find(etiquetas == k);

    % Calcula la coordenada y del centro (centroide) del objeto k
    cy = mean(fila);

    % Calcula la coordenada x del centro (centroide) del objeto k
    cx = mean(col);

    % Mantiene la visualización actual y superpone el marcador del centroide
    hold on;

    % Dibuja un marcador circular blanco en el centroide
    plot(cx, cy, 'Marker', 'o', ...
         'MarkerEdgeColor', 'w', ...
         'MarkerFaceColor', 'w', ...
         'MarkerSize', 10);

    % Dibuja un marcador en forma de asterisco negro en el centroide
    hold on;
    plot(cx, cy, 'Marker', '*', ...
         'MarkerEdgeColor', 'k');
end
```

b) **¿Por qué se detectan varios centros en una de las monedas?**
El umbral calculado por graythresh puede no ser el óptimo para todas las monedas, resultando en una segmentación incorrecta.

c) **¿Qué ocurre si cambias bwlabel(B,8) por bwlabel(B,4)? ¿se detectan las mismas componentes conectadas y objetos? ¿Por qué?**
Con conectividad 4, podrías detectar más componentes conectadas si los objetos en la imagen tienen conexiones delgadas o están conectados diagonalmente. Estos se verían como componentes separadas.

d) **Prueba a detectar el centro de las componentes conectadas de otra imagen y comenta el resultado.**
No hay muchas diferencias más que solo existe un único centro justo en el centro de la imagen.

```
% Lee la imagen 'coins.png' y la almacena en la variable I
I = imread('CHROMOS2.TIF');

% Aplica un filtro de mediana para reducir el ruido
I = medfilt2(I);

% Calcula el umbral de binarización utilizando el método de Otsu
nivel = graythresh(I);

% Convierte la imagen en escala de grises a una imagen binaria utilizando el umbral calculado
B = im2bw(I, nivel);

% Realiza operaciones morfológicas para mejorar la segmentación
B = imopen(B, strel('disk', 2)); % Eliminación de ruido pequeño
B = imclose(B, strel('disk', 5)); % Unión de partes cercanas

% Crea una nueva figura y muestra la imagen binaria B
figure, imshow(B);

% Etiqueta los componentes conectados en la imagen binaria B
[etiquetas, nobjetos] = bwlabel(B, 8);

% Convierte la matriz de etiquetas a una imagen RGB donde cada etiqueta
% tiene un color diferente, para visualización
E = label2rgb(etiquetas);

% Crea una nueva figura y muestra la imagen RGB con los objetos etiquetados
figure, imshow(E);

% Determinación del centro de cada objeto conectado
for k = 1:nobjetos
    % Encuentra las filas y columnas de los píxeles que pertenecen al objeto k
    [fila, col] = find(etiquetas == k);

    % Calcula la coordenada y del centro (centroide) del objeto k
    cy = mean(fila);

    % Calcula la coordenada x del centro (centroide) del objeto k
    cx = mean(col);

    % Mantiene la visualización actual y superpone el marcador del centroide
    hold on;

    % Dibuja un marcador circular blanco en el centroide
    plot(cx, cy, 'Marker', 'o', ...
         'MarkerEdgeColor', 'w', ...
         'MarkerFaceColor', 'w', ...
         'MarkerSize', 10);

    % Dibuja un marcador en forma de asterisco negro en el centroide
    hold on;
    plot(cx, cy, 'Marker', '*', ...
         'MarkerEdgeColor', 'k');
end
```

Estás diseñando un sistema automático para que el vehículo no se salga del carril en los tramos rectos. Tu primera tarea es identificar las líneas de la carretera, describe un algoritmo que detecte las 3 líneas más largas de la IMAGEN PROPUESTA.

a) Prueba a aumentar el número de líneas a detectar y muestra el resultado.

No se pueden detectar más líneas puesto que la imagen no ha sido proporcionada.

b) Prueba a detectar líneas en otra imagen y comenta el resultado.

Se utiliza la función houghpeaks para encontrar los picos en la matriz de acumulación de la transformada de Hough. Luego, estos picos se procesan para obtener las líneas correspondientes mediante la función houghlines. Se filtran las líneas detectadas conservando solo las más largas (si modificamos la variable de *numLineas* detectara más o menos líneas). Estas líneas se dibujan en la imagen original junto con sus puntos finales y el segmento más largo se destaca en rojo.

```
I = imread('5.1.10.tiff');
B = im2bw(I, 0.70);
figure, imshow(B)
[H,theta,r] = hough(B);
numLineas=30; %Número de líneas a buscar, busca las más largas
P = houghpeaks(H,numLineas,'threshold',ceil(0.3*max(H(:)))); %Buscamos
los picos
x = theta(P(:,2)); %méramente ilustrativo
y = r(P(:,1)); %méramente ilustrativo
figure, plot(x,y,'s','color','black'); %méramente ilustrativo
xlabel('Theta');
ylabel('R');
lines = houghlines(B,theta,r,P,'FillGap',5,'MinLength',20); %Procesa
los picos (líneas). Si las líneas están más cerca de 'FillGap' las une,
si miden menos de 'MinLength' las descarta
figure, imshow(I), 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'); % Pinta
las rectas
 plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow'); % Pinta
el principio de las líneas
 plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red'); % Pinta el
final de las líneas
 % Cálculo del segmento más largo (es meramente ilustrativo)
 len = norm(lines(k).point1 - lines(k).point2);
 if ( len > max_len)
 max_len = len;
 xy_long = xy;
 end
end
% Destaca el segmento más largo
plot(xy_long(:,1),xy_long(:,2),'LineWidth',2,'Color','red');
```