UNIVERSIDAD POLITÉCNICA DE MADRID

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, AUTOMÁTICA Y FÍSICA APLICADA

Prácticas de Visión Artificial

Práctica 6

Prácticas de Segmentación de las imágenes

6 Técnicas de segmentación de las imágenes

En esta práctica se tratará de experimentar con las técnicas clásicas de segmentación de imágenes empleando *"Image Procesing Toolbox"* de Matlab. Segmentar es dividir la imagen en regiones con interés; crear una descripción visual de nivel medio. La práctica se inicializará exponiendo la técnica basadas en las transformadas de Hough para líneas rectas y círculos. Luego se emplearán las técnicas de umbralización combinadas con el procesamiento morfológico y la extracción de las características de los objetos segmentados. Y para acabar, se usarán los métodos de crecimiento de regiones y algoritmos basados en técnicas de dividir y fusionar (*split & merge*), empleando árboles cuaternarios (*quadtree*).

6.1 Transformadas de Hough

Las transformadas de Hough determinan la localización de curvas parametrizadas dentro de una imagen. En este caso se emplearán para la localización de líneas rectas y círculos. La entrada a este algoritmo es una imagen binarizada, donde se ha etiquetado los píxeles representantes de los bordes con el nivel '1' y al fondo se le coloca con nivel '0'. Para esta práctica se he elegido el detector de Canny. Descárguese el material de la práctica y sitúese en el directorio de **/Segmentacion/Houhg**, cargue la imagen de los cables de un teleférico y aplique el detector de Canny:

```
imgEnt=imread('cables_gris.bmp');
imgBorde=edge(imgEnt,'canny');
imshow([im2double(imgEnt),imgBorde]);
```

Varíe los parámetros del detector de Canny para mejorar los resultados de etiquetación de los bordes. Considérese que sólo se desea determinar las líneas de gran extensión. El siguiente paso será preparar el espacio paramétrico y aplicar posteriormente la transformación:

```
theta = 0:180;
[acum,rho] = radon(imgBorde,theta);
figure(1);
imagesc(theta,rho,acum), colorbar;
xlabel ('theta (grados)'), ylabel ('rho (pixeles desde el centro)')
title('Espacio de líneas');
```

Una vez realizada la transformación, la selección de las rectas dependerá del umbral colocado en las votaciones. La transformada de Hough hace del espacio paramétrico una rejilla de votaciones; por cada píxel seleccionado da unos votos a cada celda elegida según la variación del ángulo de la normal de la recta, θ. Sólo se

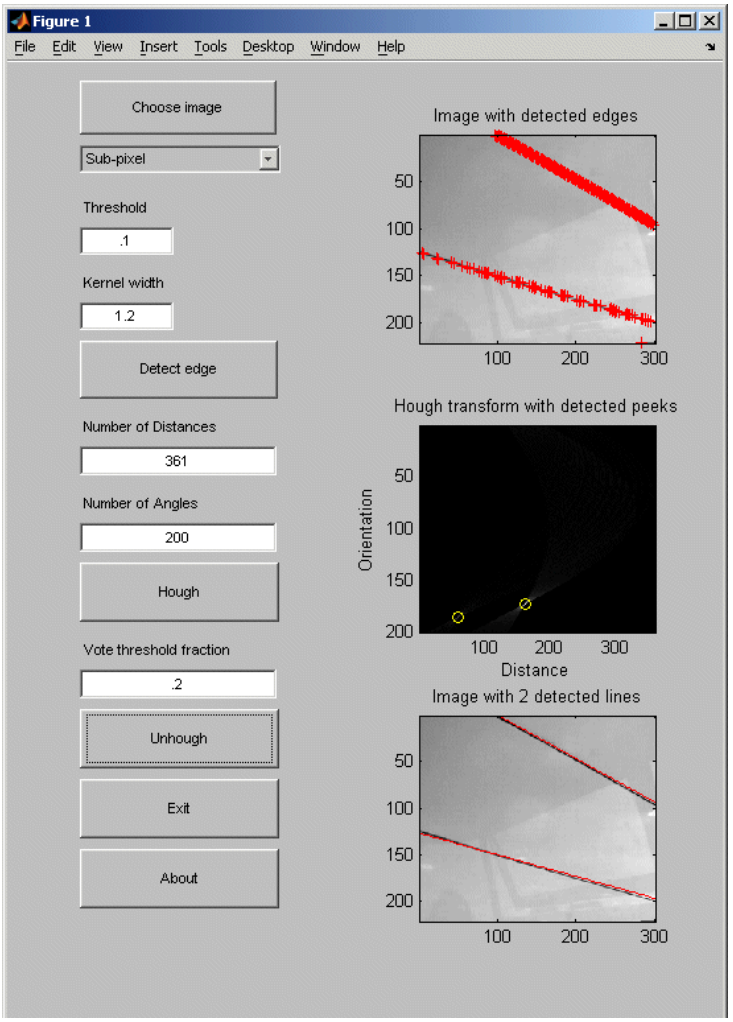
considerarán aquellas rectas que superen ese umbral que serán las que se representen:

```
% Votaciones
[x,y] = find(acum>100);
hold on; plot(theta(y),rho(x),'*r');hold off;
t = -theta(y')*pi/180;pause;
lineas = [cos(t)' sin(t)' -rho(x)];
cy = size(imgEnt,1)/2-1;
cx = size(imgEnt,2)/2-1;
lineas(:,3) = lineas(:,3) - lineas(:,1)*cx - lineas(:,2)*cy;
figure(2);
imshow(imgEnt);
```

Haga una función en Matlab donde pueda decirle el nombre del fichero de la imagen, los parámetros del operador de Canny y el umbral de las votaciones. Aplíquelo sobre la imagen 'circuit.tif'.

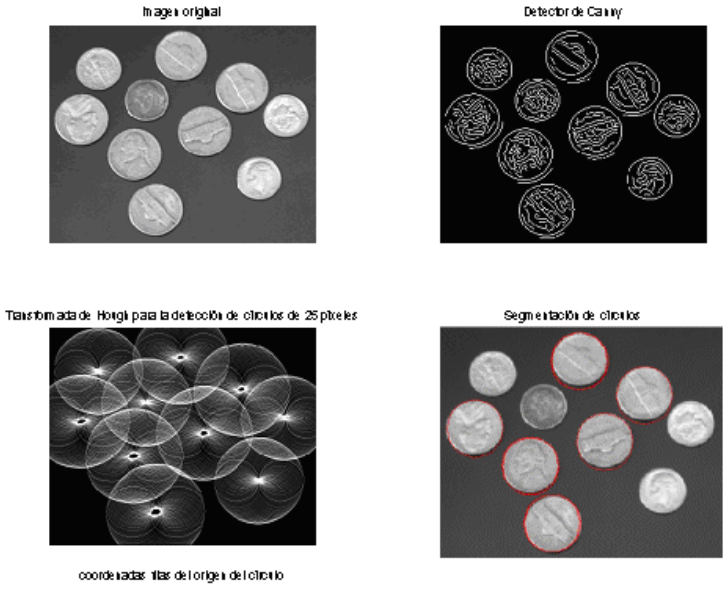
Para una mejor comprensión de las transformadas de Hough utilice la función demo *cvproj*[1] :

[1] Funciones extraídas de '*Simon Fraser University, Burnaby, BC V5A 1S6, Canada'*



La siguiente tarea es emplear la transformada de Hough para localizar círculos. Utilice la función de *houghcircle*[2] sobre la imagen *coins.png.* Mida cuantos píxeles es el radio de las monedas y determine el centro de cada una de ellas.

[2] Amin Sarafraz, University of Tehran, Iran.



Resolución Matlab

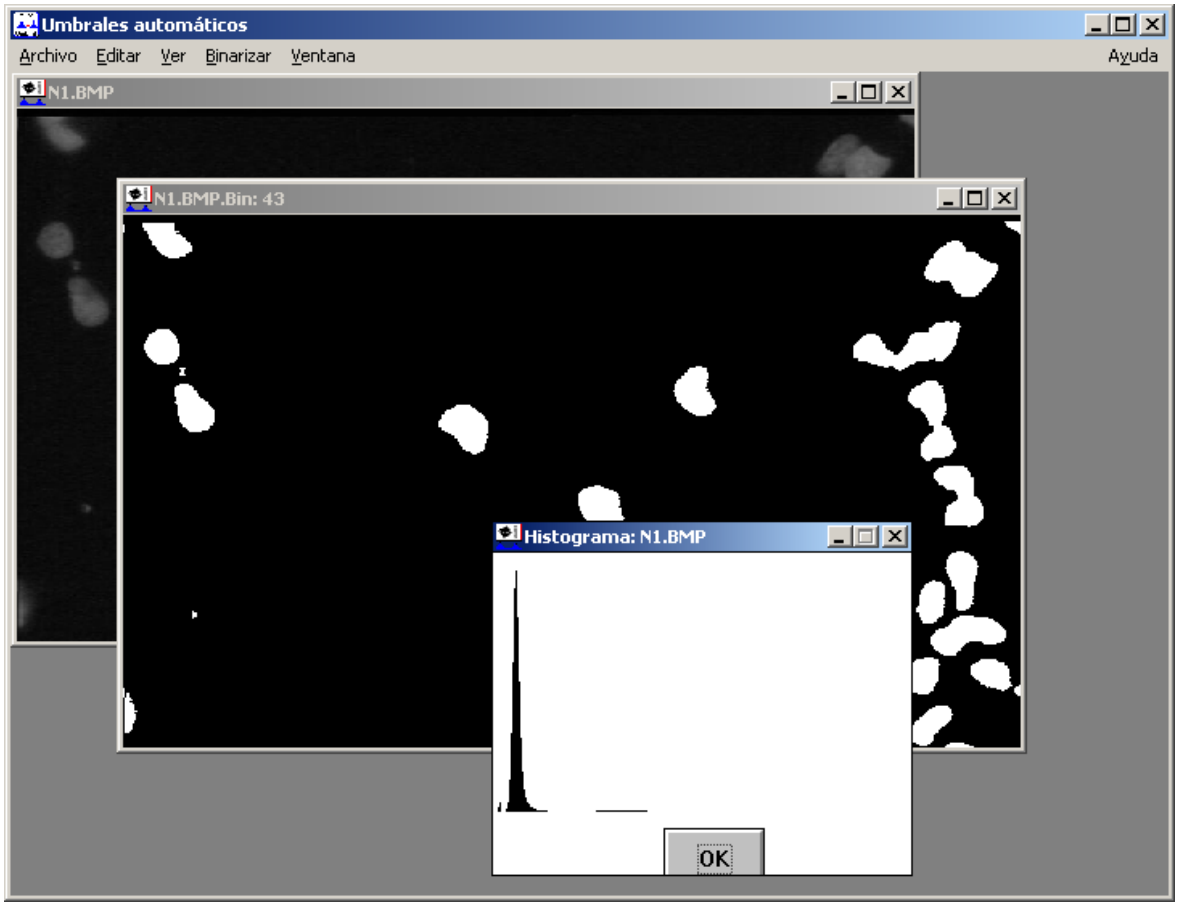
```
imgEnt = imread('coins.png');radioMoneda = 30;
[y0,x0,Accumulator]=houghcircle(edge(imgEnt,'canny'),radioMoneda,4);
UmbralVotaciones = 45; [x,y]=find(Accumulator>UmbralVotaciones);
figure(1);
imshow(imgEnt);hold on;
for i=1:size(x,1)
        dibujarCirculos(radioMoneda,y(i),x(i));
end;hold off
```

6.2 Umbralización

La umbralización es una técnica de segmentación ampliamente utilizada en la industria. Se trata de definir un umbral, de forma que separe los objetos de interés respecto del fondo. Para su aplicación se exige una clara diferencia entre los objetos y el fondo de la escena. La técnica más utilizada es la segmentación por análisis del histograma. Cuando éste presenta dos picos y en entre ambos hay un valle, el umbral quedará fijado por la posición del valle. Los píxeles de los objetos se les aginará '1' y al fondo '0', quedando binarizada la imagen. En esta práctica, se empleará el método de Otsu para la detección del umbral. La escena será una imagen de microscopía. Se trata de determinar cuántas células hay en la imagen. Cambie al directorio **/Segmentacion/Umbralizacion** y haga las siguientes instrucciones:

```
imgEnt = imread('n1.bmp');
figure(1);imhist(imgEnt);
umbral = graythresh(imgEnt);
imgBW = im2bw(imgEnt,umbral);
figure(2);imshow([im2double(imgEnt),imgBW]);
```

Ejecute la aplicación *'winumb.exe'* (**valido para WIN32**). Visualice la imagen de las células y haga una tabla entre los distinto métodos de obtención del umbral y el nivel dado. Posteriormente, elija cual es el mejor método.



Método	Umbral	Observaciones
Otsu		
Varianza continua		
...		

Estas técnicas también han sido implementadas en Matlab. Se ha escrito un script capaz de aplicar los métodos de segmentación basados en análisis del histograma. Admite un solo parámetro de entrada, el nombre de la imagen. Vea el resultado con:

```
determinar_umbral('n1.bmp');
```

Sin embargo, esta etapa de umbralización no completa la interpretación final de la imagen. El objetivo es contar cuántas células hay en ella. Se precisa de una etapa de post-procesamiento. En este caso se va a emplear técnicas de procesamiento morfológico. Obsérvese que las células tienen una forma de tipo ovoide y que ocupa un área de cientos

de píxeles contiguos. Para la eliminación del ruido generado por la umbralización se procederá a aplicar un filtrado morfológico tipo *'opening'*:

```
se = strel('disk',10);
imgObj = imopen(imgBW,se);
figure(1);imshow([imgBW,imgObj]);
```

Separado los objetos de la imagen se procederá a etiquetarlos. Esta operación tiene como finalidad asignar un número de 1 hasta L, siendo L el número de objetos en la imagen para su posterior extracción de sus características. Se rastrea la imagen y se le coloca una etiqueta buscando la conectividad del objeto:

```
imgEtiq = bwlabel(imgObj);
figure(2); imshow(label2rgb(imgEtiq));
```

Cuando la imagen está etiquetada se puede extraer las características del objeto. En este caso, sólo se va a capturar el área, el centroide y el eje mayor y menor de cada objeto en la imagen:

```
datos = regionprops(imgEtiq,'Area','Centroid',...
    'MajorAxisLength','MinorAxisLength');
matrizSal=[];
for i=1:size(datos,1)
    matrizSal = [matrizSal;datos(i).Area,datos(i).Centroid];
end
matrizSal
```

Utilice impixelinfo para identificar cuál es la célula más grande y que debe de coincidir con la información dada por `matrizSal`.

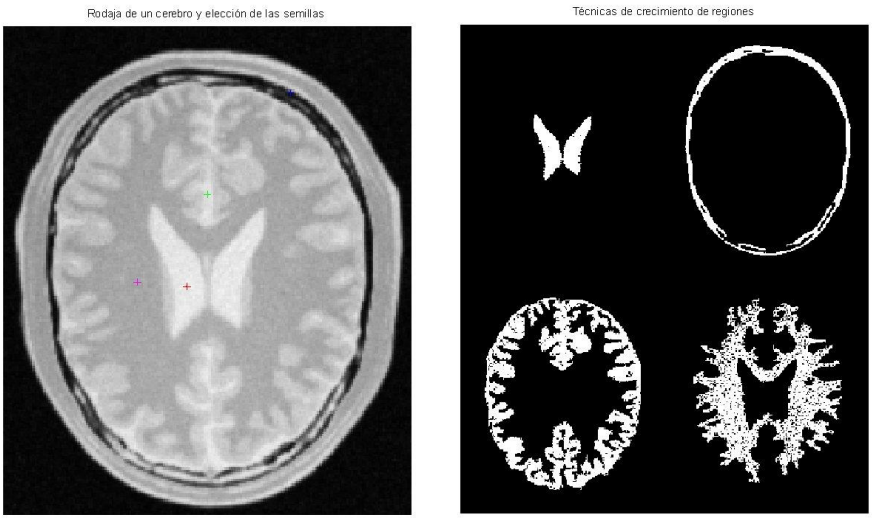
6.3 Segmentación orientada a las regiones

Las técnicas de segmentación orientadas a las regiones tienen su base en las reglas de similitud y en la conectividad de los píxeles. Las regiones se forman mediante píxeles que tengan conectividad y presenten alguna propiedad de similitud y discrepancia respecto al resto de los píxeles que no pertenecen a la región.

La primera técnica a experimentar se basa en el crecimiento de regiones. Se elige un píxel semilla de la región a obtener y se le aplica a sus vecinos la regla de similitud. Aquellos píxeles que cumplan se añadirán a la región creciente. Sobre estos nuevos píxeles añadidos se volverá aplicar la regla de similitud a sus vecinos. El algoritmo parará cuando los píxeles vecinos a la región creciente no cumplan el criterio de similitud. Para su estudio dirigirse al directorio **/Segmentación/Crecimiento**. Se cargará la imagen, se realizará un suavizado previo y se buscará la semilla adecuada. La regla de similitud empleada se basa en que la diferencia del nivel de gris del píxel a estudiar y el brillo de la región creciente sea menor a un determinado umbral. Véase la segmentación de los ventrículos del cerebro:

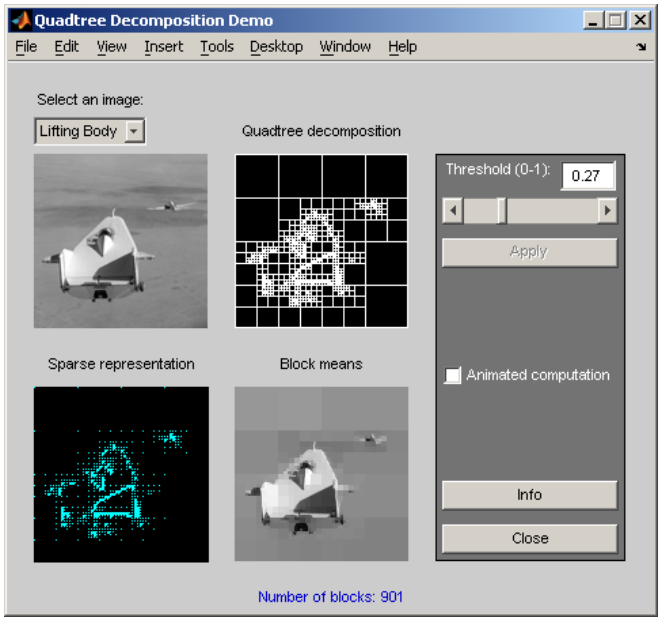
```
imgEnt = imread('brain.png');
im1=imfilter(imgEnt,fspecial('gaussian'));
%Ventrículo semila 116, 82
%Solo izquierdo
imgBW1 = regionGrowing(im1, 116, 82, 10);
%Los dos
imgBW2 = regionGrowing(im1, 116, 82, 20);
```

Localice las semillas adecuadas y el umbral necesario para determinar el hueso, el tejido blando y el tejido duro. Los resultados de la segmentación deben de ser los indicados en la figura:



6.3.1 Técnicas de división y fusión (split & merge) y árboles cuaternarios (quadtree)

En vez de emplear semillas para el crecimiento de regiones, se pasa a la descomposición de la imagen en regiones arbitrarias, de forma que si la región es muy discrepante con algún tipo de regla se dividirá, en caso contrario, buscará fusionarse con regiones adyacentes. Para la partición de la imagen en regiones arbitraria se suele emplear árboles cuaternarios. Se toma la imagen y se divide en cuatro rectángulos iguales. Se analiza cada región, si ésta es muy discrepante se vuelve a dividir en otras cuatro, generando un árbol cuaternario. Utilice la demo *qtdemo* para entender mejor los árboles cuaternarios. Esta demo es obsoleta en versiones recientes.



La segmentación de división y fusión de regiones analiza cada subregión generada por el árbol cuaternario, el cual ha dividido la imagen en regiones homogéneas a diversas resoluciones. Sitúese en el directorio **/Segmentación/Descomposicion**:

```
imgEnt = imread('liftingbody.png');
im1=imfilter(imgEnt,fspecial('gaussian'));
imgDescomp = qtdecomp(im1,.27);
imgDivision = imgEnt;
for dim = [128 64 32 16 8 4 2 1]
    [valores,fila,columna] = qtgetblk(im1, imgDescomp, dim);
    if (~isempty(valores))
        doublesum = sum(sum(valores,1,'double'),2);
    end
    imgDivision = qtsetblk(imgDivision, imgDescomp, ...
        dim, doublesum ./ dim^2);
end
figure(1);
imshow([imgEnt,uint8(imgDivision)]);
```

Una vez dividido la imagen se procederá a la fusión de regiones adyacentes empleando técnicas de crecimiento de regiones:

```
[m n] = size(imgEnt);
D = logical(zeros(m, n,'uint8'));
imgSegm = zeros(m, n);
x=1;y=1;
imgDivision=double(imgDivision);
i=1;
while( size(x,1) ~= 0)
    imCond = (imgDivision > (imgDivision(x(1),y(1))-20))...
        & (imgDivision < (imgDivision(x(1),y(1))+20));
    imCond = imCond & (~D);
    im1 = regionGrowing2(imgDivision, x(1), y(1), 20,imCond);
    imgSegm(im1)=i;
```

```
    i = i +1;
    D = D | im1;
    [x,y]= find(D == 0);
end
imgEtiq = label2rgb(imgSegm);
figure(2);imshow(imgEtiq);
```

Los resultados de la segmentación se visualizan sobre la imagen de entrada:

```
borde =edge(imgSegm,'canny');
imshow(ImagResMarcado(imgEnt,borde));
```

Imagen etiquetada

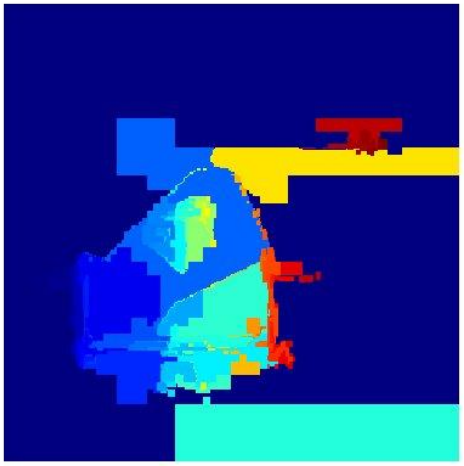


Imagen segmentada empleando técnicas de divisón y fusión