Sistemas de Visión Artificial

**Práctica 3. Segmentación de imágenes**

Nombre: Juan Casado Ballesteros

Fecha: 24 de octubre de 2019

# Segmentación supervisada lineal, mediante umbral(es) global(es)

### ¿Cuantos umbrales son necesarios para segmentar todos los objetos de la imagen?

Se necesitan tres umbrales que nos generarán 4 zonas.

### Estime los niveles de umbral necesarios para segmentar todos los objetos de la imagen y represéntelos con distintos colores.

1: 0.15

2: 0.5

3: 0.85

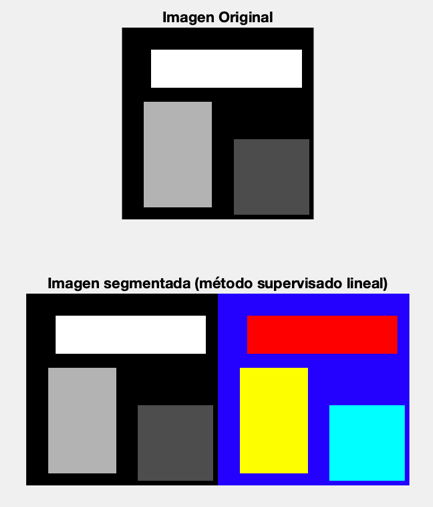
thresh = [0.15, 0.5, 0.85]; % vector de umbrales

BW\_seg = imquantize(BW,thresh); % Segmenta la imagen

RGB\_seg = label2rgb(BW\_seg); % Coloreado de la imagen

imshowpair(BW,RGB\_seg,'montage')

title('Imagen segmentada (método supervisado lineal)');



### Cambie el valor de la varianza del ruido y comente el resultado de la segmentación para distintos valores de la varianza con y sin filtrado.

Cuanto mayor es la varianza más juntas están las campanas de gauss creadas.

El filtro hace que la separación entre ellas aumente pues las hace más estrechas, reduce la varianza de este.

### A screenshot of a cell phone Description automatically generated

### Realice la segmentación supervisada lineal y comente el resultado.

% Segmentaci√≥n

thresh = [80,180];

I\_seg = imquantize(I,thresh);

RGB\_seg = label2rgb(I\_seg);

figure('Name', 'SEG');

imshowpair(I,RGB\_seg,'montage')

title('Imagen segmentada (método supervisado lineal)');

### A picture containing ground, outdoor, boat, sky Description automatically generated

La segmentación produce muy malos resultados pues los colores de los objetos que deseamos segmentar se confunden con los del fondo ya que comparten niveles de gris por lo que solo con el histograma no los podremos separar estos objetos.

# Segmentación supervisada no lineal, con modelos gaussianos

### Represente el histograma y marque en el mismo el valor de las medias obtenidas para el fondo y los objetos.

La calidad de los resultados obtenidos depende en gran medida de lo bien o mal que seleccionemos los puntos de la imagen a partir de los cuales se calculan los umbrales con los que se segmenta.

imhist(dI)

hold on

plot([mean\_f], zeros(1,2),'ro', 'LineWidth',2);

plot([mean\_b], zeros(1,2),'bo', 'LineWidth',2);

plot([mean\_f+std\_f,mean\_f-std\_f], zeros(1,2),'rx', 'LineWidth',1);

plot([mean\_b+std\_b,mean\_b-std\_b], zeros(1,2),'bx', 'LineWidth',1);

title('Colocación de los centroides de las clases tras entrenar');

hold off;

BUENOS VALORES

mean\_b = 0.3464;

mean\_f = 0.6941;

std\_b = 0.0830;

std\_f = 0.0793;

A close up of a logo

Description automatically generatedA screenshot of a cell phone

Description automatically generated

VALORES INVERSOS

Si al seleccionar lo píxeles del fondo hacemos clic en los granos y al seleccionar los granos en el fondo se nos invierte la selección.

mean\_b = 0.6879;

mean\_f = 0.3523;

std\_b = 0.0801;

std\_f = 0.0825;

A close up of a logo

Description automatically generatedA screenshot of a cell phone

Description automatically generated

MALOS VALORES

Si elegimos valores aleatorios la segmentación también lo será.

mean\_b = 0.5136;

mean\_f = 0.4993;

std\_b = 0.2755;

std\_f = 0.2895;

A close up of a logo

Description automatically generatedA screenshot of a cell phone

Description automatically generated

# Segmentación no supervisada lineal, kmeans

### ¿Qué significa que el bucle de entrenamiento se termine cuando (t < T\_max) y cuando (error < error\_max)?

Si salimos por (error < error\_max) es que nuestros centroides están suficientemente bien ajustados como para que la solución nos valga. El error mide cuanto se han desplazado los centroides respecto de su posición en la iteración anterior. Cuando el error sea muy pequeño es que apenas habrán cambiado de posición de modo que podremos finalizar pues ya no lo van a hacer más.

Si salimos por (t < T\_max) es que llevamos demasiadas iteraciones intentando ajustar los centroides sin lograrlo lo cual indica que las posiciones aleatorias iniciales de los centoides no nos han sido favorables para encontrar una solución o bien que no hay una con el error lo suficientemente bajo. La solución será errónea.

### Obtenga la imagen segmentada correspondiente a los centroides obtenidos en el apartado anterior.

La segmentación. Podría haberse realizado con la función bwlabel no obstante para este caso se ha realizado a mano.

CC = sort(CC);

previous = CC(1);

len = size(CC);

img\_labels = ones(size(I));

for x = 2:len(2)

current = CC(x);

mean = previous + (current - previous)/2;

img\_chunk = [];

img\_len = size(I);

for i = 1:img\_len(1)

for j = 1:img\_len(2)

if I(i,j) >= mean

img\_labels(i,j) = x;

end

end

end

previous = current;

end

imshow(label2rgb(img\_labels))

A screenshot of a social media post

Description automatically generatedA close up of a logo

Description automatically generated

### Cargue la imagen circlesBrightDark.png, cambie el número de clases k=3 y muestre cada clase segmentada con un color diferente.

k = 3;

# A screenshot of a social media post Description automatically generatedA close up of graphics Description automatically generated

# Segmentación basada en bordes, análisis local.

### Cambie el operador de bordes de sobel a canny y comente los resultados.

Con el operador canny podemos realizar una detección de bordes mucho más precisa y selectiva. Pues el un algoritmo que tiene en cuneta más parámetros para lograr determinar los bordes realmente representativos.

Resulta más sencillo ajustar el algoritmo de sobel pues solo tiene un umbral que el algoritmo de canny que tiene tres parámetros por los que puede ser ajustado.

No obstante, para este caso concreto que solo nos interesa el un contorno aproximado para luego rellenarlo podemos observar que, aunque los bordes de canny hayan sido mejores el resultado final es el mismo.

BWs = edge(I, 'sobel', 0.14);

BWc = edge(I, 'canny', [0.31 0.8], 0.5);

figure

subplot(1,2,1);imshow(BWs);title('Imagen de bordes sobel');

subplot(1,2,2);imshow(BWc);title('Imagen de bordes canny');

# A close up of a logo Description automatically generated

### Comente el código del operador morfológicos imdilate y comente los resultados.

Imdilate permite "copiar" los pixeles de una imagen en posiciones próximas a ellos. Una dilatación en forma de línea vertical dará la impresión de que la cámara se estaba moviendo de arriba a abajo al hacer la foto.

Con el kernel en forma de cruz que hemos utilizado lo que logramos es que cada píxel del borde se copie a su derecha y a su izquierda logrando que este acaba siendo más grueso de modo que se cierren los contornos.

A picture containing brass knucks, weapon

Description automatically generated

### Cree una máscara solapada con la imagen original

Labeloverlay Toma como entrada dos matrices. La primera representará una imagen y la segunda una máscara. Sobre la imagen dibuja de un color distinto cada valor presente en la máscara.

os = labeloverlay(I,Ls);

oc = labeloverlay(I,Lc);

figure

subplot(1,2,1);imshow(os);title('overlay sobel');

subplot(1,2,2);imshow(oc);title('overlay canny');

A close up of a logo

Description automatically generated

# Segmentación basada en bordes, análisis global (líneas).

### Calcule el número de picos y el número de líneas obtenidas. ¿Son iguales o diferentes? ¿Por qué?

En general si, pues a partir de cada punto de P se genera una línea con la rho y la theta por el punto indicadas.

No obstante, ya que en houghlines podemos indicar una longitud de línea mínima, MinLength podemos forzar a que no sean iguales poniendo un valor excesivamente alto de modo que las líneas más pequeñas se descarten. Adicionalmente, de forma similar, estableciendo un valor bajo de FillGap, podemos forzar la separación de una línea en dos o unir dos líneas en una si fuera alto.

### ¿Por qué solo se representan rectas horizontales? Realice cambios para representar rectas en otros ángulos y dibújelas.

Para cambiar el ángulo en el que queremos obtener las rectas debemos modificar la variable angles que afecta a las funciones hough y houghpeaks. Angles expresa el rango válido de los ángulos que queremos obtener, si el ángulo va desde [-90, 90) tomaremos todos los ángulos, si este esta toma solo ángulos próximos a 0 nos quedaremos solo con las líneas verticales, por el contrario, si nos quedamos con los extremos del rango obtendremos las líneas horizontales.

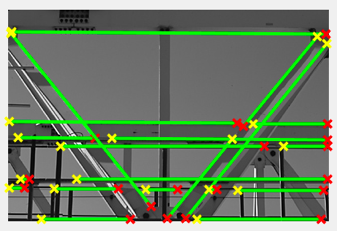
angles = [(-90:0.5:-80)]; % Líneas horizonatales.

angles = [(-5:0.5:5)]; % Líneas verticales.

angles = [(-90:0.5:89)]; % Todas las líneas.

A picture containing colorful, indoor

Description automatically generatedA picture containing sky, light, green, building

Description automatically generated

### Cambie el código para obtener solo la mejor recta (la que tenga una mayor valor en la transformada de Hough) para el siguiente rango de ángulos [40, 50].

Para hacer esto debemos modificar la variable angles para dejarla tomando valores de ángulos en el rango de [40, 50).

Tras hacer esto debemos cambiar la variable max\_peacks (y ponerla a 1) para controlar la cantidad de puntos que se eligen del espacio de hough para que nos devuelva solo el mejor.

Por último, ajustaremos el parámetro FillGap para que la línea se rellene hasta que solo nos sea devuelta una y no esa misma partida por la mitad.

angles = [(40:0.5:50)];

max\_peaks = 1;

lines = houghlines(BW, theta, rho, P, 'FillGap', 100, 'MinLength',5);

# A picture containing building Description automatically generated

# Segmentación basada en bordes, análisis global (círculos).

### Encuentre todos los círculos oscuros en la imagen y dibújelos. Cargue la imagen tape.png y estime el radio del rollo de cinta que se ve en la imagen.

Debemos buscar los umbrales superior e inferior de cada círculo.

RGB = imread('tape.png');

imshow(RGB);

Rmin = 60;Rmax = 85;

[centersExt, rExt] = imfindcircles(RGB,[Rmin Rmax],'ObjectPolarity','bright');

viscircles(centersExt, rExt,'Color','b');

Rmin = 30;Rmax = 60;

[centerMed, rMed] = imfindcircles(RGB,[Rmin Rmax],'ObjectPolarity','bright');

viscircles(centerMed, rMed,'Color','b');

Rmin = 20;Rmax = 45;

[centerInt, rInt] = imfindcircles(RGB,[Rmin Rmax],'ObjectPolarity','dark');

viscircles(centerInt, rInt,'Color','r');

display(strcat("Circulo exterior: ", num2str(rExt)))

display(strcat("Circulo medio: ", num2str(rMed)))

display(strcat("Circulo interior: ", num2str(rInt)))

Circulo exterior: 83.4879

Circulo medio: 47.5661

Circulo interior: 38.0574

### A picture containing indoor, sitting, floor Description automatically generated

Se han buscado tanto los círculos oscuros como los claros.

# Segmentación basada en regiones, watershed

### Cargue la imagen 'rice.png' umbralízela y realice una segmentación utilizando la transformación Watershed, tal y como se ha hecho en el ejemplo dado. ¿Existe sobre-segmentación, es decir, se han segmentados muchos más objetos de los que hay en la imagen?

I = imread('rice.png');

figure

imshow(I)

T = graythresh(I);

BW = ~imbinarize(I,T);

figure

imshow(BW)

D = bwdist(BW);

figure

imshow(D)

D = -D;

D(BW) = -Inf;

L = watershed(D);

rgb = label2rgb(L,'jet',[.5 .5 .5]);

figure

imshow(rgb);

title(sprintf('Transformada Watershed. Objetos segmentados: %d',max(max(L,[],1),[],2)));

Se produce sobre segmentación debido a que watershed solo funciona bien para segmentar formas circulares. Cuando las formas se alegan mucho de la circunferencia, en este caso por ser muy alargadas, el cálculo de la distancia a los bordes produce que exista sobre segmentación ya que las diferencias de proximidad de las paredes de los propios objetos producen que se generen barreras dentro de ella misma.

### A close up of a logo Description automatically generated

### Si se ha producido sobre-segmentación realice un rellenado de los objetos segmentados en el apartado anterior con valor 0 y vuelva a hacer el etiquetado a partir de la imagen rellenada. Finalmente compruebe si el número de objetos segmentados es menor.

fL=bwfill(L==0,'holes');

labeled = bwlabel(fL);

rgb = label2rgb(labeled,'jet',[.5 .5 .5]);

figure

imshow(rgb)

title(sprintf('Transformada Watershed. Objetos segmentados: %d',max(max(labeled,[],1),[],2)));

### A close up of a logo Description automatically generated

Ya que entorno a cada objeto se nos marca un borde de 0s podemos rellenar a partir de él de modo que generemos una imagen donde no se produzca sobre segmentación. Aquellos objetos próximos a los bordes de la imagen no serán rellenados pues no están íntegramente rodeados de 0s. No obstante, seguirán contando como segmentos distintos al resto.

Podemos ver ahora que la cantidad de objetos distintos si coincide con los que realmente hay.