Universidad Rey Juan Carlos

Práctica 1

Introducción al Procesamiento de Imágenes en el dominio espacial

Curso 2016 - 2017

En esta práctica se pretende familiarizar al alumno con las herramientas básicas de procesamiento de imagen en el dominio espacial utilizando MATLAB.
En la práctica se presentan los comandos de MATLAB que permiten leer imágenes y realizar conversiones (de tipos de imágenes), visualizar imágenes en pantalla y grabarlas en disco, remuestrear imágenes y reducir el número de niveles de intensidad. En la última parte de la práctica se realizarán transformaciones puntuales y globales.

Utilice la ayuda de MATLAB (`help`+ comando) para conocer el funcionamiento de los comandos utilizados en este guión. Tenga en cuenta que a lo largo de la práctica puede utilizar las instrucciones `clear all` y `close all` para evitar posibles interferencias con otras variables o ventanas.

Puesto que una imagen es una función bidimensional, ésta se puede representar como una matriz de valores donde cada elemento corresponde a un píxel. Si la imagen es digital, el conjunto de valores es discreto y hace referencia a los niveles de intensidad de la imagen.

MATLAB almacena las imágenes en formato matricial. La *toolbox* de procesamiento de imágenes soporta cuatro tipos de imágenes. Para una imagen de tamaño de MxN píxeles:

- Si la imagen es binaria (*binary image*), la matriz de tamaño MxN sólo contiene dos valores diferentes que codifican "ceros" (negro) y "unos" (blanco) lógicos. La matriz puede ser de tipo `uint8, double` o `logical.`
- Si la imagen es de intensidad (*intensity image*), ésta se representa por una matriz de tipo `uint8, uint16, o double`. Una imagen de tipo `uint8` tiene una profundidad de píxel de 8 bits y la matriz de tamaño MxN tiene valores enteros comprendidos en el intervalo [0, 255].
- Si la imagen es RGB (*true color*), la matriz tiene tamaño MxNx3 (array tridimensional) y sus valores pueden ser de tipo `uint8, uint16, o double`. Esta matriz equivale a tres matrices bidimensionales de tamaño MxN. El primer plano de la tercera dimensión representa las intensidades de color rojo, la segunda las de color verde y la tercera las de azul.
- Si la imagen es indexada (*indexed image*), la matriz MxN tiene P valores enteros que son índices a un mapa de color RGB (también denominado paleta de color o LUT). MATLAB representa una imagen indexada por una matriz de tipo `uint8, uint16, o double,` y por otra matriz de tamaño Px3 y de tipo `double,` la cual representa la paleta (LUT) de colores. Cada fila de la LUT especifica la componente de rojo, verde y azul de un único color.

Por defecto, MATLAB guarda la mayor parte de los datos como matrices de tipo MATLAB (64-bits). Sin embargo, esta representación puede no ser siempre la más adecuada para procesamiento de imagen, ya que puede tener altos requisitos de memoria. Para reducir los requisitos de memoria, MATLAB puede guardar los datos de una imagen en matrices de enteros sin signo de 8 bits (tipo `uint8`) o 16 bits (tipo `uint16`).

I. Lectura, visualización y almacenamiento de imágenes

El comando `imread` lee una imagen desde un archivo gráfico. Haga uso de la ayuda de MATLAB para conocer la sintaxis de este comando. Utilice el comando `imread` para leer las imágenes 'peppers.png', 'coins.png', 'cara.tif' (las dos primeras son imágenes internas de MATLAB, mientras que la tercera imagen se proporciona como material adjunto a la práctica).

Indique las instrucciones MATLAB utilizadas, tanto para cargar las imágenes anteriores como para responder a las siguientes preguntas:

- ¿qué dimensión tiene la variable asociada a cada una de las imágenes? Utilice el comando `size` para obtener la anchura y altura de la imagen (medidas en píxeles).
- ¿de qué tipo de imagen, de las cuatro indicadas anteriormente, se trata en cada caso?, ¿cómo lo ha deducido?
- obtenga analíticamente el tamaño de cada imagen en bits

Justifique razonadamente sus respuestas. Puede ser de utilidad el comando `whos`.

MATLAB permite visualizar una matriz de valores en forma de imagen utilizando el comando `imshow`. Habitualmente, el mínimo valor de la matriz corresponde al negro y el máximo al blanco. Si hace uso de la ayuda del comando `imshow` comprobará que, para una correcta visualización, es conveniente indicar el rango de niveles de la imagen a visualizar. De no indicarse, MATLAB considera que el rango dinámico es el máximo permitido. Para aprovechar el rango de niveles representable es conveniente utilizar el mínimo y máximo valor de intensidad de la imagen.
El comando `imtool` también permite visualizar imágenes, además de conocer las coordenadas de cada píxel (y su valor) cuando el cursor del ratón pasa sobre del píxel concreto. Estos valores aparecen en la esquina inferior izquierda de la ventana donde se representa la imagen.
Tenga en cuenta que la utilización del comando `imtool` genera una ventana cada vez que se utiliza. Por el contrario, el comando `imshow` reemplaza el contenido de la última figura que se haya representado, siendo por tanto recomendable generar una nueva figura antes de su utilización (comando `figure`).

Visualice cada una de las tres imágenes anteriores utilizando los dos comandos (`imshow` e `imtool`).

Algunas funciones de conversión de tipos de imágenes

Haga uso de la ayuda de MATLAB para explicar qué permite hacer cada uno de los siguientes comandos: `gray2ind, ind2gray, rgb2gray, im2bw, rgb2ind, ind2rgb`.

Utilice los comandos anteriores para realizar conversiones de tipo utilizando las imágenes anteriores, de modo que:

- Una imagen binaria pase a ser RGB. Construya manualmente el mapa de color, de modo que los píxeles blancos de la imagen binaria (denominados píxeles de primer plano) tengan color rojo, y los píxeles oscuros de la imagen binaria tengan color amarillo.
- Una imagen RGB pase a escala de grises.
- Una imagen RGB pase a una imagen indexada con 255 niveles.
- Una imagen RGB pase a indexada con 5 niveles.

- Una imagen de grises pase a imagen indexada con 5 niveles.
- Una imagen de grises pase a binaria

Visualice los resultados y anote las instrucciones utilizadas y una representación del resultado obtenido.

Las imágenes procesadas con MATLAB se pueden almacenar en disco utilizando el comando `imwrite`. Indique las instrucciones necesarias para almacenar dos imágenes: una de las imágenes RGB anteriores, y la correspondiente imagen indexada con 5 niveles. Compruebe si puede visualizar las imágenes con otros programas de visualización de imágenes (por ejemplo, con Paint).

II. Modificación de la resolución espacial y en intensidad

El comando `imresize` permite redefinir el tamaño de una imagen en MATLAB, modificando así la resolución espacial de la imagen. Haga uso de la ayuda de este comando para conocer su sintaxis. Como comprobará, el segundo argumento de entrada permite determinar si la imagen resultante corresponde a un diezmado (valor del segundo parámetro inferior a 1) o a una interpolación (valor del segundo parámetro superior a 1).
Utilizando la imagen 'Lena_512.tif' (de tamaño 512x512 píxeles y proporcionada como material adjunto a esta práctica), indique las instrucciones necesarias para:

- Redimensionar la imagen de Lena a 256x256 píxeles. Guarde la imagen generada en la variable Lena_256 y en el fichero "Lena_256.tif".
- Redimensionar la imagen de Lena a 128x128 píxeles. Guarde la imagen generada en la variable Lena_128 y en el fichero "Lena_128.tif".

Visualice las imágenes redimensionadas y comente los resultados obtenidos. ¿Qué imagen tiene menor resolución (espacial)?

Utilice la imagen de menor resolución espacial obtenida anteriormente para crear otra imagen del mismo tamaño que la imagen original. Explique qué se consigue con las siguientes instrucciones. Comente y justifique los resultados obtenidos.

```
Lena_512a = imresize(Lena_128,4,'nearest');
figure, imshow(Lena_512a)

Lena_512b = imresize(Lena_128,4,'bilinear');
figure, imshow(Lena_512b)
```

Como sabe, la calidad de una imagen también depende del número de niveles de intensidad utilizados (resolución en intensidad). Explique cómo se puede reducir el número de niveles utilizando alguno de los comandos presentados en esta práctica. Indique las instrucciones necesarias para:

- reducir a 16, 4 y 2 el número de niveles de intensidad de la imagen 'Lena_512.tif', almacenando el resultado en las variables Lena_512_16, Lena_512_4 y Lena_512_2, respectivamente.
- visualizar las variables anteriores, incluyendo en la figura un texto que explique la imagen representada. Puede ser de utilidad el comando `title`.

III. Histograma y mejora de contraste

El histograma de una imagen permite observar cuantitativamente la distribución de niveles de intensidad de una imagen. La función que ofrece MATLAB para este fin es `imhist`. Cuando esta función se utiliza sin argumentos de salida, MATLAB genera una figura con la visualización del histograma.

Haga uso de la ayuda de MATLAB para representar en una misma figura los histogramas de las imágenes 'Lena_512.tif', Lena_512_16, Lena_512_4 y Lena_512_2. Considere 256 intervalos (*bins*) en el histograma de la imagen de intensidad. Cuando represente el histograma de las imágenes indexadas, tenga en cuenta que el segundo argumento de entrada debe ser la LUT. Indique las instrucciones utilizadas y la figura resultante, identificando adecuadamente cada histograma con el comando `title`. Utilice la instrucción `subplot` para representar los histogramas en la misma figura. Utilice el editor de propiedades de la figura para ajustar adecuadamente los límites de representación en el eje de ordenadas si fuera necesario. Comente el resultado.

Una de las herramientas que permiten mejorar el contraste de una imagen es la ecualización. MATLAB realiza la ecualización de una imagen mediante el comando `histeq`. Haga uso de la ayuda que MATLAB ofrece para el comando `histeq` e indique las instrucciones necesarias para:

- leer la imagen 'pout.tif' (es una imagen interna de MATLAB) y visualizarla
- ecualizar la imagen y visualizarla
- representar los histogramas (imagen original y ecualizada) en una misma figura. Si fuera necesario, utilice el editor de propiedades de la figura para ajustar adecuadamente los límites de representación en el eje de ordenadas.

Comente los resultados obtenidos.

IV. Interpretación del color y transformaciones puntuales

Indique las instrucciones para:

- Cargar la imagen 'peppers.png' (es una imagen interna de MATLAB), extraer la componente roja y visualizarla. Justifique el resultado obtenido teniendo en cuenta los colores representados en la imagen *true color*. Comente las diferencias/similitudes de intensidad con la imagen de grises obtenida en el apartado I.
- Obtener el histograma de cada componente R, G, B. Analice y justifique las diferencias entre los histogramas.
- Obtener el negativo de la componente roja y, con ella, volver a componer una imagen RGB y visualizar el resultado. Justifique los cambios de color respecto a la imagen original teniendo en cuenta el modelo de mezcla de colores correspondiente (modelo aditivo).
- Representar el contenido de la componente roja en una imagen RGB, de modo que el color predominante de la imagen resultante sea el rojo. Visualice la imagen resultante.

Universidad Rey Juan Carlos

Práctica 2

Filtrado de imágenes en el dominio espacial

Curso 2016 – 2017

El objetivo de esta práctica es familiarizar al alumno con las herramientas básicas de filtrado espacial utilizando MATLAB.

Utilice la ayuda de MATLAB (`help`+ comando) para conocer el funcionamiento de los comandos utilizados en este guión. Tenga en cuenta que a lo largo de la práctica puede utilizar las instrucciones `clear all` y `close all` para evitar posibles interferencias con otras variables o ventanas.

En esta práctica trabajaremos con dos imágenes en escala de grises (*gray scale*):

- Una imagen sintética (artificial) que creará el propio alumno.
- La imagen 'coins.png'.

I. Imágenes contaminadas con ruido

Para observar la efectividad del filtrado espacial conviene trabajar con imágenes reales o con imágenes sintéticas contaminadas con alguna fuente de ruido. MATLAB permite degradar la calidad de una imagen con ruido de distinta naturaleza mediante la función `imnoise`. En esta práctica trabajaremos con tres fuentes de ruido: 'gaussian', 'salt & pepper' y 'speckle'.

Considere las fuentes de ruido 'gaussian' y 'speckle' (también denominados ruido gaussiano y ruido granular, respectivamente) y contamine con ellas una imagen sintética de 8 bits, intensidad uniforme y luminancia 128. Considere que la imagen es cuadrada y de tamaño 256x256 píxeles. Indique y justifique la secuencia de instrucciones utilizada, tanto para generar la imagen como para añadir ruido. Tenga en cuenta que la matriz que representa la imagen debe ser de tipo uint8. Considere que las fuentes de ruido tienen media nula y su potencia media es 0.02 en los dos casos.

Justifique los resultados obtenidos en las imágenes a partir de la comparación del histograma de todas las imágenes. En la **representación del histograma**, considere en todos los casos selección automática del rango en el eje vertical.

¿Qué sucede con el histograma si la fuente de ruido considerada es de tipo 'salt & pepper'? Utilice también el valor 0.02 como la probabilidad de que un píxel sea ruidoso cuando se considera el ruido impulsivo. Indique la secuencia de instrucciones utilizada. Represente la imagen y el histograma correspondiente, justificando las diferencias con las imágenes e histogramas anteriores.

II. Filtros espaciales suavizadores

Filtrado lineal

Si un operador local efectúa una transformación lineal, su comportamiento puede definirse completamente mediante la respuesta h[n,m] al impulso. Si h[n,m] es rectangular y simétrica respecto de su origen, su aplicación sobre la imagen (es decir, la operación de convolución) puede efectuarse aplicando la máscara w[n,m] = h[−n,−m] sobre cada píxel de la imagen original.

MATLAB ofrece la función `imfilter` para llevar a cabo la operación de filtrado lineal, para lo cual toma como parámetros la imagen original y la máscara de filtrado. La imagen resultante es del mismo tipo que la imagen original (double, uint8, uint16, etc.).

Teniendo en cuenta lo anterior, defina una máscara de suavizado de tamaño 5x5 que garantice que no se modifica el rango dinámico de la imagen. Indique y justifique la secuencia de instrucciones utilizada.

Filtre la imagen sintética contaminada con ruido de naturaleza gaussiana con la máscara generada en el párrafo anterior. Utilice para ello la función `imfilter` con dos argumentos de entrada (imagen y máscara): en este caso, `imfilter` genera una imagen del mismo tamaño que la imagen a filtrar y aplica la opción de tratamiento de bordes conocida como ‘zero padding’. Después, repita el filtrado considerando la opción de filtrado 'symmetric' (‘mirror padding’). Indique la secuencia de instrucciones utilizada en cada caso, representando las imágenes filtradas y sus histogramas. Comente y justifique las similitudes y diferencias (tanto en las imágenes como en los histogramas asociados) teniendo en cuenta las opciones de filtrado consideradas, en especial las asociadas a ‘Boundary options’.

Repita el procedimiento del párrafo anterior con la opción de filtrado 'symmetric' y las fuentes de ruido de tipo 'salt & pepper' y 'speckle'.

Justifique razonadamente qué efecto tendrá aumentar el tamaño de la máscara. Apoye su justificación en un ejemplo práctico considerando una máscara de tamaño 35x35 y la imagen sintética contaminada con ruido gaussiano. Obtenga el histograma de la imagen filtrada. Comente las diferencias con la imagen filtrada que usa una máscara de tamaño 5x5, así como con el histograma correspondiente.

Filtrado no lineal

Uno de los filtros no lineales más utilizados es el filtro de mediana, que asigna a cada píxel de la imagen procesada la mediana de los valores de los píxeles situados en un entorno local. MATLAB permite aplicar este tipo de filtrado con el comando `medfilt2`.

Utilice una máscara de tamaño 5x5 y aplique un filtro de mediana sobre todas las imágenes sintéticas contaminadas con ruido. Considere como opción de filtrado 'symmetric'. Indique la secuencia de instrucciones utilizada. Comente y justifique los resultados, considerando tanto las imágenes filtradas como sus correspondientes histogramas. Compare los resultados con los obtenidos al aplicar el filtro lineal. Extraiga conclusiones.

III. Filtros espaciales de realce de contornos

En esta última etapa utilizaremos la imagen de monedas 'coins.png'. Lea la imagen y almacénela en la variable I.

El comando `fspecial` permite crear máscaras de filtrado espacial de determinados tipos. Haga uso de la ayuda de este comando para crear una máscara H de tamaño 3x3 que implemente el filtro de Prewitt. Se pide:

- filtrar la imagen 'coins.png' con la máscara H;
- enfatizar los contornos de I ortogonales a los anteriores modificando la máscara de filtrado (aunque ésta debe seguir siendo de tipo Prewitt);
- utilizar las dos imágenes filtradas para construir una imagen que aproxime el módulo del gradiente. Almacene la imagen obtenida en la variable I_grad_Prewitt.

Incluya la secuencia de instrucciones utilizada para obtener la imagen I_grad_Prewitt, así como la propia imagen.

Umbralice (comando `im2bw`) la imagen I_grad_Prewitt para obtener una imagen binaria donde los píxeles de primer plano correspondan a los contornos de las monedas. ¿Considera adecuado el resultado? Justifique su respuesta.

Utilice ahora una máscara espacial diferente, también de tamaño 3x3, pero asociada a un filtro isotrópico de realce de contornos como el estudiado en las clases teóricas. Aplique el filtro a la imagen I y umbralice el resultado para obtener una imagen binaria donde los píxeles de primer plano correspondan a los contornos de las monedas. ¿Considera adecuado el resultado? Justifique su respuesta.

¿Qué filtro de realce de contornos, de los dos anteriores (Prewitt/isotrópico), considera más adecuado para obtener los contornos de las monedas? En lo que sigue utilizaremos ese filtro, pero aplicando una etapa de preprocesado a la imagen I para tener una versión suavizada de la misma. Recuerde cuáles son los dos tipos de filtro de suavizado utilizados en esta práctica y justifique razonadamente cuál de los dos considera que es más adecuado utilizar para obtener posteriormente los contornos de las monedas. Una vez elegido el filtro, utilícelo con una máscara de tamaño 11x11 para suavizar la imagen I, y umbralice la imagen resultante de modo que todos los píxeles asociados a las monedas sean píxeles de primer plano, y todos los píxeles de fondo correspondan a la superficie sobre la que se encuentran las monedas. Proporcione una imagen binaria de tipo uint8 donde el valor 255 corresponda al primer plano, guardando la imagen resultante en la variable I_BW. Indique la secuencia de instrucciones utilizada y la imagen binaria resultante.

Aplique el filtro de realce elegido previamente a la imagen I_BW y almacene el resultado en la variable I_BW_realce. ¿Qué obtiene?

IV. Composición de imágenes

En la última parte de esta práctica haremos una composición de las imágenes I e I_BW_realce para obtener una imagen en color similar a la que se muestra en la Figura 1. Proponga el procedimiento a seguir e impleméntelo en MATLAB. Anote la secuencia de instrucciones utilizada, teniendo en

cuenta que, en la medida de lo posible, es importante que las imágenes generadas sean de tipo uint8. Puede resultar conveniente hacer uso del comando `imadd`, que permite realizar la suma de dos imágenes del mismo tipo.

Figura 1. Imagen RGB con monedas enmarcadas por curvas de trazo continuo y color rojo.

Práctica 3

Filtrado de imágenes en el dominio espectral

Curso 2016 – 2017

Universidad Rey Juan Carlos

El objetivo de esta práctica es familiarizar al alumno con las herramientas básicas de procesamiento de imágenes en el dominio frecuencial utilizando MATLAB.

Utilice la ayuda de MATLAB (`help`+ comando) para conocer el funcionamiento de los comandos utilizados en este guión. Tenga en cuenta que a lo largo de la práctica puede utilizar las instrucciones `clear all` y `close all` para evitar posibles interferencias con otras variables o ventanas.

En esta práctica trabajaremos con las siguientes imágenes binarias:

- La imagen 'triangulo.bmp'.
- La imagen 'triangulodesp.bmp', que es una modificación de la anterior donde la posición espacial del triángulo dentro de la imagen es diferente.
- La imagen 'triangulozoom.bmp', versión escalada de 'triangulo.png'.
- La imagen 'triangulogirado.bmp', versión girada de 'triangulo.png'.

I. Representación de la imagen en el dominio transformado

MATLAB ofrece la función `fft2` para llevar a cabo la FFT (Transformada Rápida de Fourier, o *Fast Fourier Transform*) bidimensional. Lea la imagen 'triangulo.bmp', almacénela en la variable X y realice la FFT utilizando las siguientes instrucciones:

```
X_FFT = fftshift(fft2(double(X),256,256));
FFT_modulo=abs(X_FFT);
FFT_fase= angle(X_FFT);
```

Explique qué hace cada una de las instrucciones anteriores. Indique la secuencia de instrucciones para representar el módulo y la fase de la FFT de X (haga uso del comando `mesh` y del comando `imshow/imtool` considerando la opción de ajuste automático del rango dinámico utilizada en la Práctica 1 de la asignatura). Observe y comente las representaciones obtenidas.

Utilice la FFT para obtener el valor asociado a la componente continua. Indique su valor y detalle cómo lo ha obtenido. Obtenga también el valor medio de la imagen realizando las operaciones que considere oportunas en el dominio espacial, anotando la secuencia de comandos utilizada. ¿Coinciden los dos valores? ¿Por qué? Puede resultar de utilidad leer la ayuda de MATLAB sobre el comando `fftshift`.

Justifique qué transformación debería aplicar a la variable `FFT_modulo` para visualizar mejor su contenido. Aplique la transformación y justifique en qué direcciones se encuentra la mayor parte de la energía. Anote la secuencia de instrucciones utilizada y la representación obtenida.

II. Propiedades de la Transformada de Fourier

En este apartado comprobaremos tres de las propiedades del módulo de la FFT: traslación, escalado y rotación. Anote las instrucciones necesarias para

1) Leer la imagen 'triangulodesp.bmp' en la variable X y representar módulo y fase de su FFT
2) Leer la imagen 'triangulozoom.bmp' en la variable X y representar módulo y fase de su FFT
3) Leer la imagen 'triangulogirado.bmp' en la variable X y representar módulo y fase de su FFT

Incluya también las representaciones obtenidas. A partir de su visualización y de la comparación con las representaciones asociadas a la FFT de 'triangulo.bmp', justifique qué transformaciones afectan al módulo y cuáles a la fase. Compare estos resultados con los resultados teóricos (se recomienda consultar alguna de las referencias bibliográficas indicadas en la guía docente).

III. Filtrado Paso Bajo en el dominio frecuencial

En este apartado realizaremos el filtrado paso bajo de la imagen 'triangulo.bmp' utilizando dos tipos de filtros paso bajo (ideal y gaussiano). Teniendo en cuenta que D_0 es un número no negativo y $D(u,v)$ es la distancia del punto (u,v) al centro del filtro:

- la respuesta en frecuencia del filtro paso bajo ideal sigue la expresión

$$H(u,v)=\begin{cases}1 & \text{if } D(u,v)\le D_0\\ 0 & \text{if } D(u,v)>D_0\end{cases}$$

- la respuesta en frecuencia del filtro paso bajo gaussiano sigue la expresión

$$H(u,v)=e^{-D^2(u,v)/2D_0^2}$$

La función `lpfilter` proporcionada en esta práctica genera las anteriores respuestas en frecuencia. Haga uso de su ayuda para obtener:

- Las respuestas en frecuencia de un filtro paso bajo ideal del mismo tamaño que la imagen 'triangulo.bmp' y tres valores de D_0: 50, 30 y 10.

- Las respuestas en frecuencia de un filtro paso bajo gaussiano del mismo tamaño que la imagen 'triangulo.bmp' y tres valores de D_0: 50, 30 y 10.

Utilice el comando `mesh` para obtener una representación 3D de las respuestas en frecuencia obtenidas. Considerando el mismo valor de D_0, compare las **representaciones del módulo para cada tipo de filtro**, utilizando tanto el comando `mesh` como el comando `imshow`. Considere que el centro del espacio de representación coincide con la frecuencia espacial (0,0). Comente brevemente las similitudes y diferencias.

Realice el filtrado de la imagen 'triangulo.bmp' con los seis filtros anteriores utilizando las modificaciones que considere oportunas sobre el siguiente fragmento de código:

```
X = imread('triangulo.bmp');
```

```
H = lpfilter('ideal', 256, 256, 10);
F=fft2(double(X));

Filtrada_freq = H.*F;

Filtrada_espacio=real(ifft2(Filtrada_freq));
figure, imshow(Filtrada_espacio, [])
```

Explique qué hace cada una de las instrucciones anteriores.
Incluya en su memoria las imágenes filtradas con los seis filtros y justifique similitudes y diferencias.

Observará que al utilizar el filtro paso bajo ideal aparece un efecto de "ondulado" que no aparece al utilizar el filtro gaussiano. Haga uso de la teoría sobre transformaciones al dominio frecuencial para justificar razonadamente ese efecto.

IV. Filtrado Paso Alto en el dominio frecuencial

Justifique de manera teórica qué esperaría obtener al aplicar un filtro paso alto a la imagen 'triangulo.bmp'. Explique detalladamente cómo debería ser el módulo de la imagen resultante.

Haga uso de la función `lpfilter` del apartado anterior para diseñar en el dominio frecuencial un filtro paso alto ideal y otro filtro paso alto con transiciones suaves en la banda de paso. Considere D_0 = 100 en ambos casos. Realice el filtrado de la imagen 'triangulo.bmp' con los dos filtros anteriores y almacene en las variables FPA_ideal y FPA_gauss las imágenes filtradas en el dominio espacial. Anote la secuencia de comandos utilizada para realizar los dos filtrados.

Utilice el comando `imshow` para representar las variables FPA_ideal y FPA_gauss. ¿Por qué no obtiene lo esperado al inicio de esta sección? Para responder a esta pregunta puede resultar de utilidad representar de manera tridimensional las variables FPA_ideal y FPA_gauss utilizando el comando `mesh`. Proponga un procedimiento para resolver ese problema e impleméntelo para los dos filtrados, almacenando el resultado en las variables FPA_ideal2 y FPA_gauss2. Anote la secuencia de comandos utilizada.

Umbralice las dos imágenes anteriores para obtener una imagen binaria donde sólo aparezcan como primer plano los píxeles asociados a los cambios espaciales de intensidad. Para el caso de FPA_ideal2 utilice la siguiente secuencia de comandos:

```
II=uint8(255*mat2gray(FPA_ideal2));
Iumbr_FPAideal = im2bw(II,umbral);
```

donde debe elegir y justificar un valor adecuado para el parámetro `umbral`. Repita el procedimiento considerando FPA_gauss2 y muestre las imágenes resultantes. ¿Qué tipo de filtro considera más adecuado, el ideal o el gaussiano?

Práctica 4

Segmentación de Imagen (I)

Curso 2016 – 2017

Universidad Rey Juan Carlos

El objetivo de esta práctica es comenzar a familiarizar al alumno con las herramientas básicas de segmentación de imagen en entorno MATLAB. Para ello se trabajará con la imagen en escala de grises 'calculadora.tif', que acompaña al material de esta práctica.

Utilice la ayuda de MATLAB (`help`+ comando) para conocer el funcionamiento de los comandos utilizados en este guión. Tenga en cuenta que a lo largo de la práctica puede utilizar las instrucciones `clear all` y `close all` para evitar posibles interferencias con otras variables o ventanas.

I. Histograma y umbralización

Utilice la instrucción `imread` para leer en MATLAB la imagen 'calculadora.tif' proporcionada como material adjunto en la práctica. Examine visualmente la imagen y responda a las siguientes preguntas:

- ¿son todas las teclas del mismo tamaño?, ¿cuál es la de mayor tamaño?
- ¿tienen todas las teclas alguna letra/número en su interior?, ¿existe conectividad entre ellas?
- ¿aparecen letras/números en el exterior de las teclas?
- ¿qué letras/números presentan mayor intensidad luminosa, los del interior de las teclas o los del exterior?

Tras examinar la imagen, ¿qué propiedad cree que puede ser útil para segmentar las teclas de la calculadora?

Como sabe, el histograma de una imagen (función `imhist`) permite analizar la distribución de niveles de intensidad. Represente el histograma de la imagen 'calculadora.tif' e indique cuántas "crestas" se pueden identificar (tenga en cuenta que es posible que deba expandir el eje de ordenadas para no tener saturación en la representación del histograma). Identifique cada "cresta" del histograma con una etiqueta e indique qué zona de la imagen corresponde con cada etiqueta. Puede resultar útil hacer uso de la función `imtool` (en la esquina inferior izquierda de la representación proporciona el valor de intensidad del píxel sobre el que se sitúa el cursor del ratón).

Determine razonadamente un nivel umbral que permita separar adecuadamente el interior de las teclas del resto. Justifique el valor seleccionado y utilícelo para obtener una imagen binaria (en adelante variable `I_U`) con el comando `im2bw`, mostrando el resultado obtenido en una figura.

II. Segmentación y caracterización de regiones

MATLAB proporciona la función `bwlabel` para realizar la segmentación de una imagen binaria. Haga uso de la ayuda de esta función para indicar la vecindad considerada si se utiliza la siguiente instrucción

```
[Seg_I_U, Nobjetos] = bwlabel(I_U);
```

Utilice la instrucción anterior para segmentar la imagen `I_U`. Para visualizar la etiqueta de cada región haga uso de la función `imtool` como

```
imtool(Seg_I_U,[])
```

y pase el cursor del ratón sobre la región cuyo número de etiqueta desea conocer. El valor de la etiqueta aparece en la esquina inferior izquierda, justo tras las coordenadas del píxel sobre el que se sitúa el ratón.

La imagen en falso color que se genera a partir de la capa de segmentación (almacenada en la variable `Seg_I_U`) se puede visualizar con las instrucciones

```
RGB_Segment = label2rgb(Seg_I_U);
figure, imshow(RGB_Segment)
```

Como ya se indicó en las sesiones teóricas, recuerde que MATLAB representa la etiqueta 0 con el color blanco. Teniendo en cuenta las instrucciones anteriores y sabiendo que cada color identifica un objeto diferente, indique cuántos objetos se obtienen como resultado de la segmentación. ¿Se obtiene un objeto por cada tecla? Como resultado de la segmentación, ¿se han extraído objetos en la parte externa de las teclas?

Es posible analizar las propiedades de las regiones segmentadas para eliminar aquéllas que no interesan, por ejemplo las regiones externas a las teclas (de tamaño muy reducido). La función `regionprops` permite analizar determinadas propiedades de los objetos segmentados. Esta función toma como uno de los argumentos de entrada la capa de etiquetas obtenida de la segmentación (variable `Seg_I_U`), y como otro argumento la característica de los objetos/regiones que se desea conocer. Puesto que estamos interesados en el tamaño de los objetos consideraremos la propiedad `'Area'`, que proporciona el número de píxeles de un objeto.

Haga uso de la ayuda de MATLAB para conocer qué **otras** propiedades se pueden obtener de cada una de las regiones segmentadas.

Tras identificar la etiqueta de cada región, indique su tamaño (número de píxeles). Para obtener el tamaño del objeto con etiqueta *et* utilice las siguientes instrucciones, donde *et* hace referencia al número de etiqueta (valor entero no negativo).

```
Props = regionprops(Seg_I_U, 'Area')
Props(et).Area
```

Puesto que el número de objetos es muy numeroso, puede resultar conveniente obtener el tamaño de cada región a través de la siguiente secuencia de instrucciones:

```
V_Area = [];

for ind_obj=1:Nobjetos
    V_Area = [V_Area Props(ind_obj).Area];
end
```

Explique el objetivo del fragmento de código anterior y, tras ejecutarlo, utilice el comando `stem` para representar en una gráfica el tamaño de cada objeto (identificado a través de su número de etiqueta en el eje de abscisas). Analice la gráfica resultante y establezca un valor de tamaño umbral que permita eliminar las regiones externas a las teclas. Justifique la elección de este valor.

Construya de manera manual (no automática) un array uni-dimensional (vector) denominado `V_No_Interes`, cuyos elementos correspondan a los valores de las etiquetas que desea eliminar (filtrar).

Utilice la siguiente secuencia de comandos para filtrar, de la imagen binaria `I_U`, las regiones de no interés.

```
[n_filas, n_cols] = size(I_U);

for ind_nfila=1:n_filas
    for ind_ncol=1:n_cols
        if I_U(ind_nfila,ind_ncol)
            numero_et = Seg_I_U(ind_nfila,ind_ncol);
            if sum(ismember(V_No_Interes,numero_et)) > 0
                I_U(ind_nfila,ind_ncol) = 0;
            end
        end
    end
end
```

Explique el fragmento de código anterior (se recomienda realizar un diagrama de flujo). Visualice la imagen binaria resultante. En este punto debería tener al menos una región en el interior de cada tecla y no debería haber píxeles de primer plano en el exterior de las teclas. De no ser así, repita el proceso hasta conseguir el resultado deseado.

III. Procesado para identificación de la tecla 'Enter'

Puesto que en el interior de las teclas puede haber uno, dos o más caracteres, y la distancia entre ellos es relativamente pequeña (y siempre inferior a la distancia entre caracteres de otras teclas), se propone procesar la imagen binaria `I_U` (resultante de la etapa anterior) para agrupar en una misma región los caracteres asociados a la misma tecla, lo que permitiría caracterizar cada tecla con una región diferente.

Para conseguir el objetivo anterior se propone realizar un filtrado espacial de media (con una máscara de tamaño 5x5) sobre la imagen binaria `I_U`. Indique y justifique la secuencia de

instrucciones utilizadas para definir la máscara y realizar el filtrado. Tenga en cuenta el tipo de variable correspondiente a la imagen que se desea filtrar (probablemente 'logical') y de salida (de tipo 'uint8'). Almacene el resultado en la variable `I_U_Fmedia`.

Aplique un umbral a la imagen `I_U_Fmedia`, de modo que sólo tengan valor nulo los píxeles "suficientemente alejados" de las letras/números superpuestos a las teclas. Justifique el umbral seleccionado y visualice la imagen binaria resultante (que denominaremos `I_U_Fmedia_Th`). Utilice esta imagen para determinar automáticamente el número de teclas (indique en la memoria la secuencia de instrucciones utilizada).

Haciendo uso de los comandos presentados y utilizados en esta práctica, proponga un procedimiento para identificar automáticamente la localización de la tecla 'Enter' a partir de `I_U_Fmedia_Th`.

Implemente el procedimiento y proporcione como resultado una imagen `I_Enter` en escala de grises y del mismo tamaño que la imagen original. La imagen `I_Enter` debe tener un nivel de intensidad nulo en todos los píxeles salvo en los correspondientes a la región donde se encuentra la tecla 'Enter', en cuyo caso debe mantenerse el nivel de intensidad de la imagen original. Anote la secuencia de instrucciones utilizada. La imagen resultante debe ser similar a la imagen de la Figura 1.

Justifique razonadamente qué habría sucedido si el tamaño de la máscara utilizada en el filtrado espacial hubiera sido mucho mayor, por ejemplo 35x35.

Figura 1. Imagen con la tecla 'Enter'.

Práctica 5

Segmentación de Imagen (II)

Curso 2016 – 2017

Universidad Rey Juan Carlos

El objetivo de esta práctica es familiarizar al alumno con las técnicas de extracción de características a fin de abordar un problema de segmentación utilizando aprendizaje no supervisado, en concreto el algoritmo *k*-medias. Para ello se trabajará con la imagen en color 'cormoran.jpg', que acompaña al material de esta práctica.

Utilice la ayuda de MATLAB (`help`+ comando) para conocer el funcionamiento de los comandos utilizados en este guión. Tenga en cuenta que a lo largo de la práctica puede utilizar las instrucciones `clear all` y `close all` para evitar posibles interferencias con otras variables o ventanas. La instrucción "`imtool close all`" permite cerrar todas las ventanas generadas con la función `imtool`.

I. Análisis visual de la imagen

Lea en MATLAB la imagen 'cormoran.jpg' proporcionada como material adjunto en la práctica y almacénela en la variable `I`. Examine visualmente la imagen (representada en la Figura 1) y responda a las siguientes preguntas:

- ¿qué objetos considera claramente discriminativos en la imagen?
- a simple vista, ¿qué características considera discriminativas?

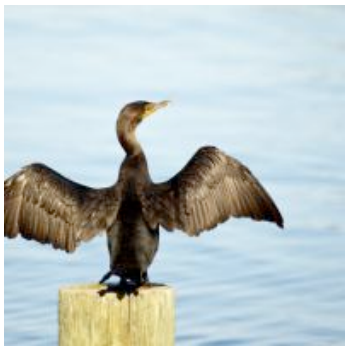

Figura 1. Imagen a segmentar en esta práctica.

Convierta la imagen a escala de grises (variable `I_gris`) y analice visualmente su histograma. ¿Cree que podría realizar la segmentación mediante umbralización múltiple? Justifique su respuesta.

II. Características RGB y algoritmo k-medias

Inicialmente abordaremos el problema de segmentación considerando como características los niveles de intensidad de las componentes R, G y B de la imagen original `I`. Para ello:

- Extraiga cada componente de color de la imagen `I`.
- Convierta cada componente en un vector columna utilizando el comando `reshape`. Así, por ejemplo, para almacenar como vector columna los niveles de intensidad de los píxeles de la componente R de `I`, utilizaremos la siguiente secuencia de instrucciones

```
I_R = I(:,:,1);
[nrows, ncols] = size(I_R);
I_R_res = reshape(I_R,nrows*ncols,1);
```

- Represente el *scatter plot* de los datos utilizando la función `plot3(`haga uso del comando `help` para conocer su sintaxis). ¿Observa grupos de puntos claramente diferenciados?

Sobre el espacio de tres dimensiones representado en la figura anterior aplicaremos el algoritmo de agrupamiento *k*-medias con *k*=3. Utilizaremos para ello la función `kmeans` del siguiente modo

```
ngrupos = 3;
rgb_res = double([I_R_res I_G_res I_B_res]);
[cluster_idx cluster_center] =
kmeans(rgb_res,ngrupos,'distance','sqEuclidean','Replicates',10);
```

Las **entradas** a la función `kmeans` son: (1) conjunto de ejemplos/observaciones (matriz `rgb_res`, donde cada ejemplo es una fila y cada columna representa una característica), (2) medida de similitud (cuadrado de la distancia Euclídea en nuestro caso, indicado a través del parámetro `'distance'`), y (3) número de inicializaciones (utilice el valor 10 en esta práctica, indicado con el parámetro `'Replicates'`). El algoritmo `kmeans` devuelve la posición de los centroides (variable `cluster_center`) y una etiqueta identificativa del *cluster* al que pertenece cada punto de entrada (variable `cluster_idx`). Puesto que el algoritmo se realiza 10 veces con distintas inicializaciones, la salida proporcionada corresponde a la realización para la que se obtiene la mínima suma de distancias intra-cluster.

Nota importante: la matriz de ejemplos/observaciones debe ser de tipo `double`; cada fila de la matriz es una observación (píxel de la imagen, en este caso) y cada columna es una característica.

Las **salidas** de la función son: (1) identificador del *cluster* al que pertenece cada punto (variable `cluster_idx`), y (2) centroide de cada *cluster* (variable `cluster_center`).

Para observar el resultado del algoritmo *k*-medias, puede representar sobre el *scatter plot* anterior los centroides resultantes del siguiente modo:

```
plot3(cluster_center(:,1), cluster_center(:,2), cluster_center(:,3),'sr');
```

El resultado de la segmentación se puede observar en el espacio imagen generando una imagen en falso color a partir del identificador de *cluster* asociado a cada observación (píxel). Para ello, debe transformar el vector con la identificación del cluster al que pertenece cada píxel en una matriz de las mismas dimensiones que la imagen original. Puede hacer uso del siguiente código.

```
pixel_labels_rgb = reshape(cluster_idx,nrows,ncols);
I_segm = label2rgb(pixel_labels_rgb);
figure, imshow(I_segm)
```

¿Considera que el resultado de la segmentación es el deseado?, ¿es posible afirmar que se produce sobresegmentación? Justifique razonadamente sus respuestas.

III. Características cromáticas ab

Como sabe, en el espacio RGB la información cromática está distribuida en las tres componentes (R, G y B). En esta sección realizaremos una transformación del espacio de representación para separar las componentes cromáticas y acromáticas de la imagen, de modo que aplicaremos el algoritmo *k*-medias únicamente sobre el espacio de componentes cromáticas. La transformación considerada es la transformación `Lab`, transformación que implementa la función `rgb2lab` proporcionada como material adjunto a la práctica.

Realice la transformación de la imagen original (espacio RGB) al espacio Lab y extraiga únicamente las componentes cromáticas (componentes ab). Represente el *scatter plot* correspondiente e intente identificar visualmente a qué color corresponden las nubes de puntos.

```
[lab_imL, l_L, a_L, b_L] = rgb2lab(I);
a_res = reshape(a_L,nrows*ncols,1);
b_res = reshape(b_L,nrows*ncols,1);

figure, plot(a_res, b_res,'.')
xlabel('a'), ylabel('b')
```

Aplique ahora el algoritmo *k*-medias sobre el nuevo espacio de características y represente la capa de etiquetas de la imagen segmentada. Comente las diferencias con la segmentación obtenida en el Apartado II.

Represente la posición de los centroides en el *scatter plot* correspondiente y responda a las siguientes preguntas:

- ¿cuál es la desviación típica (comando `std` de MATLAB) asociada a la componente `a` obtenida a partir de `I`?
- ¿cuál es la desviación típica asociada a la componente `b` obtenida a partir de `I`?
- ¿cree que alguna de las dos componentes tiene más influencia al determinar la posición de los centroides con el algoritmo *k*-medias?, ¿por qué?

Para que no haya una componente que domine en el cálculo de distancias del algoritmo *k*-medias únicamente porque su rango dinámico es mayor, se propone normalizar cada componente del espacio `ab` para que tenga media nula y desviación típica uno. La matriz de ejemplos con características normalizadas se asignará a la variable `ab_norm` (cada fila es un ejemplo y cada columna es una característica). Explique cómo se realiza la normalización si la secuencia de instrucciones es la siguiente:

```
ab_res = [a_res b_res];
ndim = size(ab_res,2);
ab_norm = ab_res;
for ind_dim=1:ndim
    datos = ab_res(:,ind_dim);
    datos_norm = (datos-mean(datos))/std(datos);
    ab_norm(:,ind_dim)=datos_norm;
end
```

Aplique el algoritmo *k*-medias sobre el espacio de características normalizado y visualice en el espacio imagen el resultado de la segmentación. ¿Observa alguna diferencia respecto a la segmentación obtenida sin normalizar las características?, ¿qué conclusión puede extraer sobre la normalización y la aplicación del algoritmo *k*-medias considerando distancia Euclídea?

IV. Características de textura

MATLAB proporciona una serie de filtros para extraer características relacionadas con la textura. La salida de cada filtro es una imagen del mismo tamaño que la imagen original donde cada píxel contiene la característica de textura extraída en un entorno local del mismo. Consulte la ayuda de las funciones `stdfilt`, `entropyfilt` y `rangefilt` de MATLAB e indique qué descriptor estadístico se obtiene con cada una de las tres funciones.

Considere un entorno de 7x7 píxeles y represente en el espacio imagen el resultado obtenido al aplicar cada uno de los descriptores de textura anteriores sobre la imagen de grises original (almacenada en la variable `I_gris`). Las instrucciones para representar el descriptor asociado al filtro `stdfilt` son las siguientes:

```
S = stdfilt(I_gris,ones(7,7));
imtool(S,[]), title('S')
```

Interprete cada una de las imágenes obtenidas conforme al descriptor utilizado. Elija dos de estos descriptores como características y realice la segmentación considerando el espacio de características correspondiente.

A la vista de los resultados, y si tuviera que elegir únicamente dos características, ¿qué características elegiría? ¿las obtenidas en el Apartado III o las extraídas en este apartado?

V. Utilización de características de distinta naturaleza

Para mejorar el resultado de la segmentación obtenida en los apartados anteriores, justifique razonadamente la elección de tres características si se desea utilizar descriptores de distinta naturaleza (color y textura).

Realice la segmentación de I considerando las tres características seleccionadas (recuerde utilizar características normalizadas). A la vista de los resultados, indique qué tipo de características (color, textura), aporta más información para realizar la segmentación de esta imagen.

Como observará, el resultado adolece de sobresegmentación, aunque ésta corresponde a regiones de tamaño muy reducido. Para resolver este problema, una alternativa es recurrir al análisis de cada región y realizar la fusión de regiones contiguas. No obstante, en esta práctica intentaremos resolver el problema de sobresegmentación preprocesando las componentes de color, que son las que más información parece que aportan para segmentar la imagen de esta práctica. Para ello, se propone aplicar un filtro de suavizado de tamaño 7x7 píxeles sobre cada componente cromática del espacio Lab.

Aplique un filtro de suavizado sobre cada componente de color y construya el nuevo espacio de características normalizado. Aplique el algoritmo de *k*-medias y comente el resultado.

Como comprobará, parte del pico del cormorán se asocia a la categoría “madera”, por tener un color similar. ¿Cree que la aplicación de una técnica de aprendizaje supervisado podría mejorar el resultado obtenido en esta práctica? Justifique razonadamente su respuesta.

Explique detalladamente (apoyándose en un diagrama de bloques) qué procedimiento seguiría si hubiera abordado la segmentación de esta imagen con un esquema de aprendizaje supervisado. Explique al menos una técnica de las que podría aplicar.

Práctica Tema 5

Análisis de Imagen (II)

Curso 2011-2012

Nombre: …….. **Apellidos:** ……...

La fecha límite de entrega de la presente memoria en el Campus Virtual es el día 30/03/12 a las 15h, y el mismo día/hora para la entrega en formato impreso. La entrega fuera del plazo indicado lleva asociada una penalización en la calificación.

En esta práctica se pretende familiarizar al alumno con las herramientas avanzadas de procesado morfológico de imagen en entorno MATLAB. Para ello se trabajará con la imagen 'I_celulas.bmp', imagen en escala de grises (uint8) procedente de un microscopio. En esta práctica se aplicarán herramientas avanzadas de segmentación y morfología matemática para realizar el conteo de células de forma automática, a la vez que se delimitan sus fronteras. Se trata de una aplicación bastante útil para monitorizar la velocidad de reproducción de células vivas, siendo una alternativa al proceso de conteo por inspección visual. Tenga en cuenta la dificultad de segmentar todas las células, especialmente si los núcleos aparecen superpuestos en la imagen.

Para conocer el funcionamiento de los comandos utilizados en este guión, utilice la ayuda de MATLAB. Tenga en cuenta que a lo largo de la práctica puede utilizar las instrucciones `clear all` y `close all` para evitar posibles interferencias con otras variables o ventanas.

Para realizar el diseño del sistema, se sugiere seguir los siguientes pasos:

1. ***Preprocesado***

   Realice un filtrado de la imagen aplicando un filtro alternado secuencial (ASF3) open-close con un EE plano que corresponda a un disco. Utilice discos de tamaño creciente (radio 1, radio 2 y radio 3). Incluya en la memoria la secuencia de instrucciones utilizada y la secuencia de imágenes obtenida tras aplicar cada una de las etapas del ASF3. Denotaremos a la variable que almacena la imagen resultante como 'I_ASF3'. Comente los resultados obtenidos.

   Comandos MATLAB a utilizar: `strel, imopen, imclose`.

2. *Segmentación por watershed*

Para realizar la segmentación haremos uso de la técnica *watershed* con marcadores, utilizando un marcador por célula y otro marcador externo para el fondo.

a. *Obtención de los marcadores de célula*

Puesto que el marcador de cada célula debe ser interno a la misma, se considerará como marcador la parte interna "oscura" de la célula. Para su extracción se propone seguir el siguiente proceso:

i. Obtener el negativo de 'I_ASF3' ⇒ 'I_neg'
ii. Erosionar 'I_neg' con un EE plano correspondiente a un disco de radio 9 ⇒ 'I_marker'
iii. Reconstrucción de I_neg utilizando como marcador 'I_marker' ⇒ 'I_rec'
iv. Obtención de una imagen binaria donde los píxeles de primer plano indiquen los máximos regionales de 'I_rec' ⇒ 'I_max_reg'

La relación de comandos MATLAB que debe utilizar es: `imcomplement`, `imreconstruct`, `imregionalmax`. Utilizando los nombres de las variables que se indican en cada paso, indique la secuencia de comandos utilizada en cada paso y represente cada una de las imágenes resultantes. Comente brevemente cada uno de los resultados.

De la comparación visual entre la imagen de células original ('I_celulas') y la imagen 'I_max_reg', se observa que no todos los máximos regionales obtenidos son útiles. A simple vista, ¿qué grupos de píxeles indicadores de máximos habría que eliminar? Haga uso del comando `imclearborder` para conseguirlo. Represente la imagen resultante y la instrucción utilizada. Denomine 'I_max_reg2' a la variable que almacena la imagen resultante.

Compare visualmente las imágenes 'I_celulas' e 'I_max_reg2'. Observará que en 'I_max_reg2' aparecen marcadas regiones que no corresponden a células. Para eliminar estas regiones tendremos en cuenta que el nivel medio de intensidad de 'I_celulas' en las zonas de interés marcadas por 'I_max_reg2' es superior en las zonas a eliminar que en el interior de las células. Consideraremos 150 como el valor de intensidad umbral para determinar si el mínimo regional corresponde o no al interior de una célula. Para eliminar las regiones de 'I_max_reg2' que no cumplan esta restricción utilizaremos la siguiente secuencia de instrucciones:

```
cc = bwconncomp(I_max_reg2);
stats = regionprops(cc,I_celulas, 'MeanIntensity');
idx = find([stats.MeanIntensity] <150);
I_max_reg3 = ismember(labelmatrix(cc), idx);
```

Explique razonadamente qué hace cada una de las instrucciones anteriores. Incluya en la memoria la nueva imagen 'I_max_reg3' y justifique las diferencias con 'I_max_reg2'. Tenga en cuenta que las regiones de primer plano de 'I_max_reg3' actuarán como marcadores interiores en la segmentación por *watershed*.

Proponga un procedimiento para determinar automáticamente el número de células a partir de la imagen 'I_max_reg3'.

b. Obtención del marcador externo

Como marcador externo utilizaremos las líneas de *watershed* obtenidas sobre la función distancia de 'I_max_reg3', previamente dilatada con un disco de radio 7 para intentar que los marcadores externos no correspondan a zonas de la célula. Utilice la siguiente secuencia de instrucciones para obtener una primera aproximación a este marcador:

```
I_dilate = imdilate(logical(I_max_reg3),strel('disk',7));
D = bwdist(I_ dilate);
DL = watershed(D);
bgm = (DL == 0);
```

Explique **conceptualmente** el procedimiento utilizado. Para ello, **haga uso** de la ayuda de MATLAB y represente en 3D cada una de las variables obtenidas en una figura (comando `mesh`).

Utilice las instrucciones

```
figure, imshow(imadd(255*uint8(bgm),I_celulas))
```

para representar en una nueva figura el marcador externo superpuesto a la imagen a segmentar.

c. **Combine** en la misma imagen los **marcadores internos y el marcador externo**. Utilice para ello la instrucción

```
I_minimos = bgm | I_max_reg3;
```

d. Puesto que aplicaremos la técnica de *watershed* sobre el **módulo del gradiente de la imagen original**, el siguiente paso es obtener la correspondiente variable asociada (variable que denominaremos 'I_celulas_grad'). Incluya en la memoria la secuencia de comandos utilizada para obtenerla, así como su representación en una imagen.

e. Como se indicó en las clases teóricas, puesto que la segmentación por *watershed* parte de los mínimos regionales de la imagen gradiente, el resultado puede conducir a sobresegmentación. Esto se puede evitar reduciendo el número de mínimos

regionales y forzando a que éstos sean los marcadores previamente extraídos. Es por ello que el siguiente paso es imponer como únicos mínimos regionales las regiones de primer plano de 'I_minimos', obteniendo como resultado la imagen 'I_celulas_grad_mrk'. Este proceso se realiza a través del comando `imimposemin`. Haga uso de la ayuda de este comando e indique en la memoria la instrucción utilizada.

f. Aplique la técnica de segmentación por *watershed* a la imagen 'I_celulas_grad_mrk' y extraiga las líneas de *watershed*, representándolas como regiones de primer plano en la variable 'L_frontera'. Incluya y justifique la secuencia de comandos utilizada.

g. Por último, superponga la imagen 'L_frontera' a la imagen de partida. Incluya en la memoria la imagen resultante y coméntela.

Práctica 6

Morfología Binaria

Curso 2016 – 2017

Universidad Rey Juan Carlos

El objetivo de esta práctica es familiarizar al alumno con las herramientas básicas de análisis de imagen, en concreto segmentación y morfología matemática para imágenes binarias.

Se trabajará con la imagen 'Board_Recorte.tif', imagen *true color* que acompaña al material de esta práctica. El objetivo es aplicar herramientas de segmentación (utilizadas en la práctica 4) y morfología matemática binaria para identificar de forma automática los chips del circuito impreso.

Para conocer el funcionamiento de los comandos utilizados en este guión, utilice la ayuda de MATLAB. Tenga en cuenta que a lo largo de la práctica puede utilizar las instrucciones `clear all` y `close all` para evitar posibles interferencias con otras variables o ventanas.

I. Transformación del espacio de representación

Utilice la instrucción `imread` para leer en MATLAB la imagen 'Board_Recorte.tif'. Visualice la imagen en color e identifique manualmente los 7 chips de mayor tamaño. Para seleccionar manualmente los chips tenga en cuenta que todos deben compartir las mismas (o similares) características.

Visualice cada componente (R, G, B) e indique los comandos MATLAB utilizados. ¿Considera que alguna de las componentes podría ser más relevante para segmentar la imagen y obtener únicamente los 7 chips indicados anteriormente? Justifique su respuesta.

Considere la transformación a otro espacio de representación, en concreto el espacio HSI. Utilice para ello la función `rgb2hsi` proporcionada como material de la práctica. Utilice la ayuda de MATLAB para obtener las componentes en el espacio transformado. Visualice cada componente e indique qué rango de variación tiene cada una de las nuevas componentes. ¿Cómo ha determinado el rango dinámico?.

A partir de la visualización de las componentes de la imagen en el espacio transformado, elija únicamente una componente (que denotaremos 'Componente') para continuar con el proceso de extracción de los 7 chips de mayor tamaño. Justifique razonadamente su elección.

II. Umbralización y filtrado

Observe el histograma de 'Componente'. Puesto que no parece muy clara la elección de un valor umbral a partir del histograma, utilizaremos la instrucción `graythresh` para obtener automáticamente un valor umbral utilizando el método de Otsu. Haga uso de la ayuda de Matlab para explicar qué criterio sigue el método de Otsu para determinar el umbral. ¿Qué valor umbral se obtiene? Utilice ese valor umbral para umbralizar la imagen 'Componente'. Escale la imagen

resultante para que la escala de variación sea [0,255] y convierta la variable resultante a una de tipo `uint8`. Anote la secuencia de comandos utilizada y visualice la imagen resultante.

Analice con detalle el resultado y justifique qué operador, relacionado con filtrado espacial y utilizado en prácticas anteriores, se puede aplicar para homogeneizar la región interna de los chips de modo que la imagen resultante siga siendo una imagen binaria tras la aplicación del filtro. Aplique el operador considerando una máscara cuadrada de tamaño 5x5. Indique la instrucción utilizada para realizar el filtrado y visualice la imagen filtrada, comentando las principales diferencias (relacionadas con las regiones de los objetos de interés) con la imagen sin filtrar.

III. Aplicación de operadores morfológicos

Como sabe, la aplicación de operadores morfológicos requiere la definición de un elemento estructurante (EE), que en MATLAB se genera mediante la instrucción `strel`. Utilice la ayuda de MATLAB para generar un EE cuadrado de lado 35 píxeles que almacenaremos en la variable 'EE_cuadrado'.

Utilice 'EE_cuadrado' para aplicar sobre el resultado final de la etapa II los siguientes operadores morfológicos:
- Erosión (instrucción `imerode`)
- Dilatación (instrucción `imdilate`)
- Apertura (instrucción `imopen`)
- Cierre (instrucción `imclose`)

Visualice la imagen obtenida tras aplicar cada uno de los operadores anteriores y justifique razonadamente el resultado teniendo en cuenta los contenidos presentados en las sesiones teóricas. Justifique qué imagen (de las cuatro anteriores) se debe considerar para delimitar lo mejor posible cada uno de los 7 chips seleccionados manualmente al inicio de la práctica. Denotaremos a la variable que almacena esta imagen como 'Im_Res_Morf'.

IV. Segmentación y caracterización de objetos

MATLAB proporciona la función `bwlabel` para realizar la segmentación de una imagen binaria. Haga uso de la ayuda de esta función para indicar la vecindad considerada si se utiliza la siguiente instrucción

```
IM_Seg = bwlabel(Im_Res_Morf);
```

Utilice la instrucción anterior para segmentar la imagen 'Im_Res_Morf'. La capa de segmentación (almacenada en la variable `IM_Seg`) se puede visualizar con las instrucciones

```
RGB_Segment = label2rgb(IM_Seg);
figure, imshow(RGB_Segment)
```

¿Cuántos objetos se obtienen como resultado de la segmentación? En este caso el número de objetos es muy reducido y se puede contabilizar teniendo en cuenta el número de colores (recuerde que cada color identifica un objeto diferente). Para obtener el número de objetos mediante una única instrucción se puede utilizar la siguiente expresión en línea de comandos:

```
Num_objetos = max(IM_Seg(:))
```

¿Qué es lo que sucede? ¿Coincide el resultado con el número de chips obtenidos manualmente? Justifique el resultado.

Realice alguna modificación sobre la imagen binaria a segmentar para que, cuando se realice la segmentación binaria, el número de objetos segmentados sea 7. Incluya en la memoria la secuencia de instrucciones utilizada, junto con la capa de segmentación resultante.

Para determinar el número de chips con forma cuadrada y forma rectangular se puede recurrir a examinar las propiedades de los objetos segmentados mediante la función `regionprops`. La función `regionprops` toma como uno de los argumentos de entrada la capa de etiquetas obtenida de la segmentación (variable `IM_Seg`), y como otro argumento la característica de los objetos que se desea conocer. En esta práctica consideraremos la propiedad de excentricidad (`'Eccentricity'`), relacionada con la forma de las regiones.

Haga uso de la ayuda de MATLAB para explicar qué representa esta característica (`'Eccentricity'`) desde el punto de vista geométrico, y qué valores son esperables (altos o bajos) para cada una de las regiones segmentadas, teniendo en cuenta la etiqueta de cada región. Para obtener visualmente la etiqueta de cada región haga uso de la función `imview` como

```
imview(IM_Seg,[])
```

y pase el cursor del ratón sobre la región cuyo número de etiqueta desea conocer. El valor de la etiqueta aparece en la esquina inferior izquierda, justo tras las coordenadas del píxel sobre el que se sitúa el ratón.

Tras identificar la etiqueta de cada región indique, para cada etiqueta, la excentricidad asociada con el objeto que ésta representa. Para obtener la excentricidad del objeto con etiqueta *et* (*et* es un valor entero positivo) utilice la instrucción

```
Props = regionprops(IM_Seg, 'Eccentricity')
Props(et).Eccentricity
```

En base a esta información, **indique qué procedimiento seguiría** para separar los chips rectangulares de los cuadrados. Justifique su respuesta.

Las técnicas de morfología matemática también se pueden utilizar para delimitar las fronteras de los objetos. Explique razonadamente cómo obtener los contornos (finos) de los chips utilizando únicamente un elemento estructurante, dos operadores morfológicos y un operador aritmético punto a punto. Indique también la secuencia de instrucciones y la imagen resultante del proceso. Tenga en cuenta que la imagen de partida es la imagen binaria cuya segmentación ha proporcionado 7 objetos.
Explique la influencia del tamaño y forma del EE.