Sesión 1. Imágenes en Matlab

1. ¿Es posible leer un determinado frame de un gif animado? A partir de la imagen "mrbean2.gif", se pide:

a) leer el frame 3 de la secuencia animada indicando los pasos seguidos.

```
%Nos movemos al directorio de las imágenes y ejecutamos…
>> image = imread('mrbean2.gif', 'Frame', 3)
```

b) determinar el tamaño de la imagen.

```
>> info = imfinfo('mrbean2.gif')
%Doble click en la variable 'info' del Workspace o ejecutar…
>> imageinfo(info)
%Tercer frame: 180x136

%Para obtener el tamaño de toda la imagen…
>> [X,Y] = size(image)

X =

   226


Y =

   300

>>
```

c) mostrar la imagen leída con imshow.

```
>> imshow(image)
```

d) indicar el tipo de imagen (escala de grises, color, binaria, indexada), así como el tipo de datos de cada píxel, justificándolo.

```
%Tenemos que guardar la imagen en un .jpg
>> imwrite(image,'mrbean2_frame3.jpg')
>> imfinfo('mrbean2_frame3.jpg')
%ColorType: 'grayscale' (escala de grises)
%BitDepth: 8
%Cada píxel es un entero de 8 bits en el rango de [0,255]
%Por lo que su tipo es: uint8
```

e) Guardar la imagen como bean2.png con 2 bits de profundidad y como bean4.png con 4 bits de profundidad. Añadirle un comentario a la cabecera indicando si la imagen es de 2 o de 4 bits de profundidad. ¿Ves diferencia entre las dos imágenes?

```
>> imwrite(image,'
ivc_practica1_imagenes/bean2.png','png','BitDepth',2,'Comment','2 bits
de profundidad')
>> imwrite(image,'
ivc_practica1_imagenes/bean4.png','png','BitDepth',4,'Comment','4 bits
de profundidad')
>> subplot(1,2,1)
imshow('bean2.png')
subplot(1,2,2)
imshow('bean4.png')
```

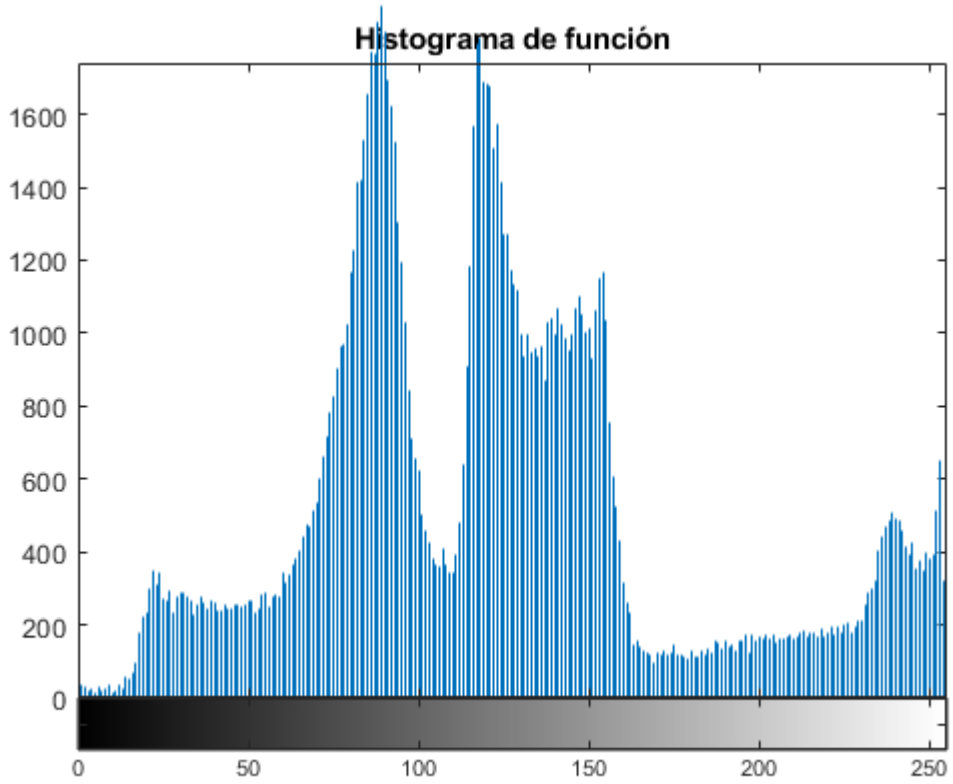
```
%La diferencia es que la de dos bits de profundidad tendrá un
%rango de [0,3] colores a representar, mientras que la de tres
%bits de profundidad está representada por un rango de [0,16]
%colores por lo que visualizará mejor
```

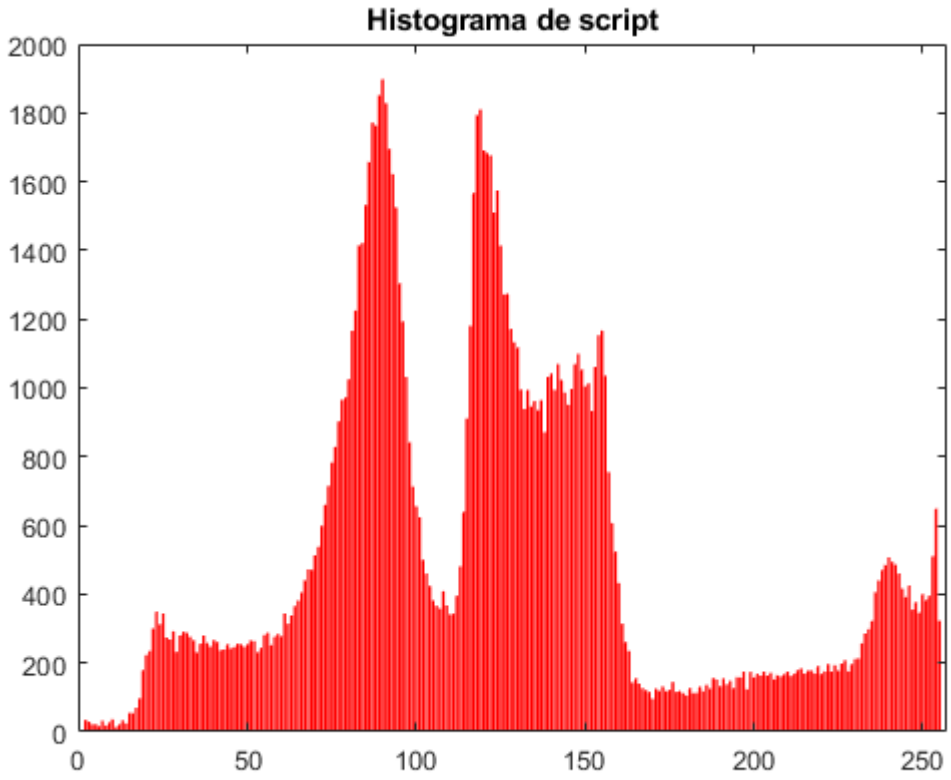
2. ¿Es posible guardar una imagen en disco con formato jpg con distintas calidades de compresión? Si es posible indicar cómo.

```
>> imwrite(image,'ivc_practica1_imagenes/mrbean2.jpg','Quality', 100)
```

3. Realizar un script que permita obtener el histograma de una imagen en escala de grises. Comparar el resultado con el histograma que se obtiene al utilizar la función imhist y comentar las diferencias.

```
>> ejercicio3
>>
```





4. **Una imagen almacenada como gif permite definir un color transparente. Este color transparente es uno de los colores indexados en el mapa de color. Realiza un script que lea la quinta imagen del gif animado "mrbean2.gif" y la transforme a escala de grises. A partir del histograma de esta imagen, elegir el valor de intensidad que más se repite y guardar la imagen en un archivo gif donde este valor de intensidad más repetido sea el color transparente.**

```
>> ejercicio4
>>
```

5. **Leer en Matlab la imagen "mano_ua.jpg". Convertir la imagen en escala de grises y guardarla como "mano_ua_gris.jpg" con el 75% de calidad. Binarizar la imagen con un umbral de 100 y guardar la imagen como "mano_ua_BW_100.jpg".**

```
>> mano = imread('ivc_practica1_imagenes/mano_ua.jpg');
>> imagen = rgb2gray(mano);
>>
imwrite(imagen,'ivc_practica1_imagenes/mano_ua_gris.jpg','Quality',75);
>> imagenGris = imread('ivc_practica1_imagenes/mano_ua_gris.jpg');
>> imagenBinaria = imbinarize(imagenGris,0.3);
>> imwrite(imagenBinaria,'ivc_practica1_imagenes/mano_ua_BW_100.jpg');
>>
```

6. **Obtener el negativo de la imagen "eps_gris.bmp" utilizando la función imadjust.**

```
>> imagen = imread('ivc_practica1_imagenes/eps_gris.bmp');
%Cambiamos los límites de contraste para la imagen de salida
>> imagenNegativa = imadjust(imagen,[0 1],[1 0]);
>> imshow(imagenNegativa);
```

7. Descargar dos imágenes distintas de Internet, convertirlas a escala de grises y utilizar histeq para ecualizar automáticamente el histograma. Realizar la misma operación con la función imadjust. Mostrar las imágenes originales y la salida tras ecualizar el histograma.

```
>> perro = imread('ivc_practica1_imagenes/perro.jpg');
>> gato = imread('ivc_practica1_imagenes/gato.jpg');
>> perroGris = rgb2gray(perro);
>> gatoGris = rgb2gray(gato);
%Mejora el contraste usando ecualización del histograma
>> ecualPerro = histeq(perroGris);
>> ecualGato = histeq(gatoGris);
%Ajuste de los valores de intensidad de la imagen o del mapa de color
>> intenPerro = imadjust(perroGris);
>> intenGato = imadjust(gatoGris);
%Mostramos imágenes
>> figure, imshow(perroGris);
>> figure, imshow(ecualPerro);
>> figure, imshow(intenPerro);
>> figure, imshow(gatoGris);
>> figure, imshow(ecualGato);
>> figure, imshow(intenGato);
```

Sesión 2. Transformaciones geométricas y en entorno de vecindad

8. Reducción de ruido. Aplicar mecanismos de reducción de ruido a las imágenes "barco_ruido1.gif" y "colina_ruido2.gif". Se pide:

a) Elegir el mejor filtro para mejorar el ruido de la imagen "barco_ruido1.gif". Para ello, probar con los filtros de media, gaussiano y mediana. Probar con la misma máscara de 3x3 para el filtro de media y mediana y 5x5 para el filtro gaussiano. Utilizar tres valores distintos de desviación estándar para el filtro gaussiano. Justificar la respuesta del filtro elegido.

```
>> barco=imread('ivc_practica1_imagenes/barco_ruido1.gif');

%Especificamos filtros, máscaras y desviaciones típicas
>> fmedia=fspecial('average',3);
>> fgaussiano1=fspecial('gaussian',5,0.2);
>> fgaussiano2=fspecial('gaussian',5,0.5);
>> fgaussiano3=fspecial('gaussian',5,0.9);

>> barcoMedia=imfilter(barco,fmedia);
>> barcoGaussiano1=imfilter(barco,fgaussiano1);
>> barcoGaussiano2=imfilter(barco,fgaussiano2);
>> barcoGaussiano3=imfilter(barco,fgaussiano3);
>> barcoMediana=medfilt2(barco);

>> figure, imshow(barcoMedia), figure, imshow(barcoGaussiano1);
>> figure, imshow(barcoGaussiano2), figure, imshow(barcoGaussiano3);
>> figure, imshow(barcoMediana);
```

b) Aplicar los filtros de media, gaussiano y de media a la imagen "colina_ruido2.gif". Utilizar máscaras de 3x3 y de 5x5 para los filtros de media y mediana para comprobar cómo afecta el tamaño de la máscara de convolución en la imagen resultante. Utilizar 0.25, 0.5 y 0.75 como desviación típica en el filtro gaussiano. ¿Cuál es el filtro que mejor funciona para eliminar el ruido presente en la imagen original?, ¿cómo influye el tamaño de las máscaras de convolución en el resultado? Compara y comenta los resultados.

```
>> colina=imread('ivc_practica1_imagenes/colina_ruido2.gif');
>> fmedia3=fspecial('average',3);
>> fmedia5=fspecial('average',5);
>> fgaussiano1=fspecial('gaussian',5,0.25);
>> fgaussiano2=fspecial('gaussian',5,0.5);
>> fgaussiano3=fspecial('gaussian',5,0.75);

>> colinaMedia3=imfilter(colina,fmedia3);
>> colinaMedia5=imfilter(colina,fmedia5);
>> colinaGaussiano1=imfilter(colina,fgaussiano1);
>> colinaGaussiano2=imfilter(colina,fgaussiano2);
>> colinaGaussiano3=imfilter(colina,fgaussiano3);
>> colinaMediana3=medfilt2(colina, [3 3]);
>> colinaMediana5=medfilt2(colina, [5 5]);

>> figure, imshow(colinaMedia3), figure, imshow(colinaMedia5);
>> figure, imshow(colinaGaussiano1), figure, imshow(colinaGaussiano2);
>> figure, imshow(colinaGaussiano3);
```

```
>> figure, imshow(colinaMediana3), figure, imshow(colinaMediana5);

%A medida que el filtro es más potente podemos eliminar más
%cantidad de ruido, sin embargo, las imágenes pierden nitidez.
%La clave es buscar un equilibrio entre ambos factores.
%El filtro que cumple al límite dicha estabilidad podría
%ser el filtro Gaussiano con desviación 0.5.

%Lo que sucede debe ser que el filtro de gauss se aplica
%mejor a las imágenes con una gran cantidad de ruido
%pequeño (tamaño píxel menor, mayor cantidad píxeles).
%Mientras que los filtros de mediana son mejores para
%imágenes con el ruido menos granulado y el gránulo más grande.
```

9. A partir de la imagen "despiste_ua.png", escribir un script que lea la imagen y la recorte de manera automática eliminando el marco blanco. Sobre la imagen recortada, se pide:

a) Rotarla 45 grados según las agujas del reloj.
b) Trasladar el resultado de la rotación 35 píxeles en X y 40 píxeles en el eje Y.
c) Realizar un sesgado en el eje X de 0.2 sobre el resultado de la traslación.
d) Escalar el resultado del sesgado un 50%.
e) Por último, sobre este resultado, realiza una rotación y traslación combinada (10 grados en el sentido contrario a las agujas del reloj y -40 píxeles en el eje Y.

```
>> ejercicio9
>>
```

2. **A partir de la imagen "mapa.png", se pide utilizar cualquier combinación de filtros morfológicos que consideres necesaria para detectar las islas del mapa. Se pide mostrar las islas detectadas en blanco sobre fondo negro.**

```
>> mapa=imread('ivc_practica1_imagenes/mapa.png');
>> mapaGris=rgb2gray(mapa);
>> mapaBinario=imbinarize(mapaGris);
%Usamos el elemento estructurante de disco con radio = 30
>> SE=strel('disk',30);
%Bottom-hat
>> BH=imbothat(mapaBinario,SE);
%Rellenamos los agujeros del mapa binario
>> mapaSinIslas=imfill(mapaBinario,'holes');
>> BH2=imbothat(mapaSinIslas,SE);
%La diferencia será el resultado
>> resultado=BH-BH2;
>> figure, imshow(resultado);
```

3. **Implementar un script que permita separar los puntos de las líneas en la imagen "puntosylineas.gif". Utilizar para ello los distintos filtros de morfología vistos en clase que más interesen para la aplicación.**

```
>> ejercicio3_s3
>>
```