Funciones de Matlab para manejar datos de imágenes:

Las imágenes se almacenan en Matlab como matrices de:

- 3 dimensiones (alto x ancho x 3) para imágenes de color real. La última dimensión corresponde a los tres planos de color (R, G y B) de la imagen.
- 2 dimensiones (alto x ancho) con un solo valor (en vez de 3) por píxel. En ese caso debe asignarse una paleta de color (o niveles de gris) a la figura (usando colormap) para poder visualizarla correctamente.

imformats -> lista formatos gráficos soportados por imread/imwrite

imread -> lee un fichero gráfico y almacena la imagen en una matriz de MATLAB de tamaño Alto x Ancho x 3 (color real) o Alto x Ancho (grises).

imwrite -> escribe una matriz (imagen) en MATLAB como un fichero gráfico para poder ser vista con cualquier otra aplicación.

image(im) -> visualiza una imagen im en una figura (ventana) Matlab.

colormap(paleta) -> establece paleta de color a usar en una ventana. Necesaria en caso de usar imágenes en color indexado o en niveles de gris.

gray(256) -> crea paleta de grises de 256 niveles.

```
im=imread('faunia.jpg');   % Lee JPG
figure(1);                 % Crea figura (ventana)
image(im);                 % Visualiza

im=im(:,end:-1:1,:); % Flip izquierda/derecha
imwrite(im,'foto_flip.jpg','Quality',50);  % Guardamos nueva imagen (poca calidad)

figure(2); image(im)

im=imread('faunia_bw.jpg'); % Lee foto BW
figure(3); image(im); pause

paleta=gray(256);   % Crea paleta color 256 grises
colormap(paleta);   % Asigna paleta a figura
```

MATLAB dispone de una toolbox (librería) con muchos programillas que hacen más fácil la manipulación y visualización de imágenes. Desgraciadamente la UPM ha dejado de licenciar la Image Toolbox.

Cuando necesitemos algo de esta toolbox os suministraré funciones "sustitutivas" que he escrito yo. Las reconoceréis porque empiezan siempre por fc (de Fotografía Computacional). Por ejemplo: fc_truesize.

Dada fc_truesize sabéis que existe una función MATLAB de la image ToolBox que se llama truesize y que hace más o menos lo mismo (intentar presentar la imagen en pantalla de forma que 1 pixel de la imagen corresponda a un píxel de la pantalla).

Ejercicio 1: Usando imread leer el fichero 'faunia_bw.jpg'. Visionarlo con image(). ¿Qué imagen véis? Asignar una paleta de grises a la figura usando:

cmap=gray(256); colormap(cmap).

Una paleta es una tabla de 256x3. Cada fila son tres valores [0-1] que codifican un color. Cuando un cierto píxel de la imagen vale 27 el sistema pinta en el correspondiente píxel de la pantalla el color representado por la fila 27 de la paleta.
¿Qué peculiaridad tiene la paleta de grises cmap creada antes?

Crear de colores aleatorios usando rand(256,3) y asignarla a la figura.
MATLAB tiene predefinidos algunas paletas. Probar con copper(256) o hsv(256)

Ejercicio 2: Usando imread leer un fichero gráfico "faunia.jpg". Veréis que es una imagen de color real guardada en una matriz 3D (alto x ancho x 3).

Usar image para presentar la imagen por pantalla. Ahora no se necesita una paleta, ya que la imagen tiene toda la información necesaria (3 colores por píxel).
Usar fc_truesize para que la imagen se muestre a su tamaño real.

Usar la notación de Matlab de matrices para extraer y mostrar por pantalla:

a) La mitad izquierda de la imagen
b) Una subimagen de alto 200 y ancho 400 empezando en el píxel x=100, y=200.

Usar fc_truesize en ambos casos. Adjuntar imágenes

Crear una imagen im2 con el plano de color azul (el tercero) puesto a 0.
Mostrar la imagen por pantalla.

Reordenar los elementos de una imagen: >> image(im(:,end:-1:1,:))
¿Qué imagen obtenéis? ¿Por qué?

Modificar la imagen original revolviendo sus planos de color. En vez de estar ordenados como RGB, reordenarlos como GBR. Visualizar la imagen resultante.

Ejercicio 3: A partir de la imagen de 'faunia.jpg' (tamaño alto x ancho), obtener una imagen de tamaño alto/2 por ancho/2 submuestreando en un factor 2 en cada dimensión. Usar fc_truesize al visualizarla para apreciar el cambio de tamaño.

Esta forma de reducir una imagen (descartando un píxel de cada N) no es muy conveniente, ya que puede dar lugar a efectos raros. Usar fc_imresize(im,0.5) que también reduce una imagen a la mitad (factor 0.5) y comparar con el resultado anterior. La función fc_imresize(im,P) admite diezmados no enteros (cualquier factor P) . Si usamos P>1 estaremos ampliando la imagen.

Adjuntar imágenes obtenidas. Comentar diferencias.

Ejercicio 4: La función rgb2gray de la toolbox de imágenes de MATLAB calcula la versión blanco/negro de una imagen color como:

```
0.2989*R + 0.5870*G + 0.1140*B
```

Obtener la versión BW de imagen de faunia.jpg licando la fórmula anterior. Recordar que R=im(:,:,1), G=im(:,:,2), etc. Para poder aplicar la fórmula indicada debéis convertir la imagen a double, usando la función double(). Tras aplicar la fórmula volver a una imagen de bytes usando uint8().

Mostrar ambas imágenes (color y grises) en una misma figura usando subplot.

Adjuntar código e imágenes resultantes

Podríamos haber usado [0.33 0.33 0.33] como coeficientes al promediar los planos de color. ¿Por qué se usan pesos distintos?

Ejercicio 5: Una fotografía correctamente expuesta (que no es lo mismo que una buena fotografía) debería idealmente tener un valor medio en las cercanías de 128 (en un rango de 0 a 255) lo que corresponde a lo que se denomina un 18% de gris.

Matemáticamente es sencillo comprobar si esto se cumple calculando la media de los valores de los píxeles. Cargar la imagen en blanco y negro anterior, pasarla a double y obtener su media mean(im(:)). Veréis un valor bastante centrado.

Una información más completa puede obtenerse de forma gráfica con el histograma de la imagen. Un histograma no es más que un recuento de los píxeles que tienen un cierto valor en la imagen. Usar la función fc_imhist(im) para visualizar el histograma de la imagen anterior. Observar que como está bastante centrado.

Adjuntar histograma. ¿Qué nos indica el pico de la derecha?

Obviamente un buen fotógrafo debe saber apartarse de ese ideal medio, pudiendo obtener fotos en las que predominan tonos oscuros (low key) o claros (high key).

Cargar las imágenes 'low.jpg' y 'high.jpg' y mostrar sus histogramas, junto con la media de sus valores.

Otro aspecto a considerar es el contraste o rango entre los negros (valores bajos) y los claros (valores altos) de la imagen. La imagen anterior tenía un contraste alto ya que tenía sus valores muy repartidos en todo el rango 0-255. El contraste está asociado a la desviación standard σ de los valores de la imagen que se puede obtener haciendo std(im(:)). En el histograma una imagen de alto contraste se caracteriza por una distribución "ancha" de sus valores cubriendo todo el rango.

Cargar las imágenes 'a.jpg' y 'b.jpg'. Mostrar sus histogramas y calcular sus desviaciones standard σ. ¿Cuál de ellas es una imagen de bajo/alto contraste?

Ejercicio 6: Cargar la imagen 'high.jpg' en im, una imagen donde predominan los valores altos (claros).

Se trata de aplicar la transformación $y = x^3$ (definida sobre [0,1]) a la imagen:

1. Convertir im a double usando double()
2. Dividirla por 255 para pasar a valores de píxel en el intervalo [0,1].
3. Elevar al cubo toda la imagen (im.^3). Los valores resultantes siguen estando en el rango [0,1].
4. Volver a escalar los valores de 0 a 255 (multiplicando por 255).
5. Volver a bytes usando la función uint8()

Visualizar la imagen final y su histograma. ¿Qué efecto ha tenido la transformación?

En una 2ª ventana (crearla con el comando figure) mostrar la imagen original (sin transformar).

Crear la paleta de paleta de grises habitual. Modificarla creando una 2ª versión elevada al cubo:

```
>> cmap=gray(256);
>> cmap3=cmap.^3;
```

Asociar una u otra paleta a la 2ª ventana. ¿Qué imagen veis en uno u otro caso?

¿Qué transformación estaríais viendo si aplicaseis cmap3 a la 1ª ventana?

Ejercicio 7: Ejecutar el programa ilusión. Os mostrará en pantalla una versión "rara" de una imagen. Para obtenerla hemos llevado a cabo el siguiente proceso:

- Conversión al espacio de color YIQ.
- Eliminar su información de luminancia. Para ello hacemos Y = 0.5 (constante).
- Invertir la información de sus planos de color (I = -I, Q = -Q).
- Convertir la imagen modificada de vuelta a RGB para visualizarla.

Se trata de mirar fijamente a la imagen sin mover la vista por ella (he añadido una cruz en el centro para que sea más fácil). Pasados 15-20 segundos o cuando veáis que los colores parecen perder intensidad, pulsar Enter (conviene tener el dedo sobre la tecla para no dejar de mirar la imagen de la pantalla).

Describir lo que veis. ¿Se os ocurre alguna explicación? ¿Por qué no funciona si movéis la vista por la imagen?

Informaros sobre el tema haciendo google de "color illusion castle"

Podéis probar con otras imágenes usando ilusion('xxx.jpg');

Proyecto: para entregar por parejas en 2 semanas

Adjuntar código / imágenes / respuestas / comentarios

A. Comparación de la información de luminancia y color en una imagen.

Cargar la imagen faunia.jpg.

Usando la función fc_rgb2ntsc pasar la imagen al espacio de color YIQ. Obtendréis una nueva imagen también con tres planos de color Y(1), I(2) y Q(3).

Hacer 2 copias de la imagen en el espacio YIQ y modificarlas de la siguiente forma:

En la primera dejar sin tocar los planos 2 y 3 (I,Q asociados al color) y modificar el plano 1 (luminancia) reduciéndolo en un factor P y ampliándolo en un factor P.
El resultado tendrá el mismo tamaño inicial pero habrá perdido información en el proceso.

En detalle:

Extraer plano 1 = Y
Aplicar fc_imresize con factor P=0.3
Aplicar fc_imresize con factor 1/P
Volver a meter el plano modificado en el plano 1 de la imagen

En la **segunda copia** dejar sin tocar el plano 1 (Y, luminancia) y modificar los planos I,Q (2 y 3) de la forma indicada antes (reducirlos y luego aumentarlos por el mismo factor P=0.3).

Una vez completado el proceso en ambas imágenes aplicar fc_yiq2rgb para volver al espacio RGB y visualizarlas.

Adjuntar las imágenes obtenidas.
Comentar los resultados. ¿En qué caso hemos modificado más la imagen?
¿En qué caso se nota la imagen más degradada?

Observar el tamaño de las imágenes comprimidas faunia.jpg y faunia_bw.jpg
Ambas están comprimidas con la misma calidad perceptual. ¿Cuál ocupa más?
¿Cuánto mas?

Relacionar ambos hechos.

B. Transferencia de color entre imágenes.

Hemos visto que la media/varianza de la distribución de los valores de los píxeles en una imagen monocroma está asociada al brillo medio/contraste de la imagen.

En diferentes espacios de color, la media de los valores de uno de los planos puede estar asociada a aspectos como el tono medio de color de la imagen (la diferencia entre una imagen cálida, de tonos amarillos o naranjas o fría, con una luz más azulada o blanca).


Se trata de intercambiar el tono de color entre ambas imágenes para conseguir:


El proceso es muy sencillo:

- Cargar las imágenes iniciales: beach1 y beach2
- Usando fc_rgb2lab convertirlas a un nuevo espacio de color Lab donde de nuevo L lleva la información de luminancia y los planos a,b la de color.
- Con un bucle, barrer los tres planos en el espacio de color Lab. En cada plano hallar la media (mean) y la desviación Standard (std) de ambas imágenes.
- Modificar ambos planos para que tengan la media y la varianza de la OTRA imagen.

Una vez terminado el "intercambio" volver al espacio inicial RGB usando fc_lab2rgb para visualizar las imágenes resultantes.

Cambiar un plano (imagen) x que tiene una media m1 y desviación standard $\sigma 1$ para que tenga media m2 y desviación standard $\sigma 2$ es muy sencillo. Basta hacer:

```
x = (x-m1)/σ1      % Pasamos a media cero y desviación Standard = 1
x = (x*σ2)+m2      % Pasamos a desviación Standard σ2 y media m2
```

Adjuntar código e imágenes resultantes.

Repetir el proceso pero ahora intercambiando sólo los planos de color ab (2 y 3), sin tocar el 1er plano (luminancia, L). Adjuntar las nuevas imágenes y compararlas con las anteriores ¿Cuáles parecen más naturales? ¿Por qué?