

# Prácticas de Visión Artificial

---

## Práctica 1: Introducción al MATLAB y al procesado de imágenes

### Procesado de imágenes

El imageprocessingtoolbox de MATLAB permite abrir imágenes, manipularlas, convertir el espacio de colores, realizar distintos tipos de filtrado, etc. En esta primera práctica vamos a realizar una serie de ejercicios para familiarizarnos con la manipulación de imágenes en el entorno MATLAB. Familiarizarse con el toolbox imageprocessingtoolbox y en particular, con las funciones: imread, imwrite, size, figure, imshow, imfinfo, etc.

### Recordar:

Las imágenes son nada más que matrices cuyos valores pueden ser 'double' (por defecto), 'uint8', 'uint16', etc. Si la imagen es blanca y negra, se representa con una matriz bidimensional:

```
im_grey_level=ones(<#filas>,<#columnas>,<#tipo>), % el tipo puede ser  
'uint8', 'double', etc.
```

Si es en color, es de tres dimensiones (filas, columnas, canales). Por ejemplo:

```
im_rgb=zeros(20,20,3);
```

Por lo tanto, hablamos de 4 diferentes tipos:

- **Grayscale** (1 canal, y valores de los píxeles uint8, uint16, double,... que definen los valores de intensidad del nivel de gris).
- **Truecolor o RGB Image** (formadas por 3 matrices, llamadas comúnmente canales, que contienen los valores de intensidad R, G y B, y los píxeles también pueden ser de tipo: double uint8, ...).
- **Binary** (valores 0s y 1s, interpretado como negro y blanco, respectivamente).
- **Indexed** (valores lógicos, de tipo uint8, uint16, double,... cuyos píxeles son índices en un mapa de colores (colormap). En cada índice del colormap se especifica una intensidad del color (R, G y B)).

Los tipos más populares son: de tipo 'uint8' o 'double'. Ejemplos:

```
im1=zeros(255,255,'uint8');  
im1=im1+300;
```

```
figure, imshow(im1);
```

¿Cuál es el máximo valor de la matriz? ¿Cómo funciona el max? ¿Qué hace max(max(im1))? ¿Por qué?

```

im2=zeros(30,30); % ¿De qué tipo es la matriz (los valores de sus elementos)?
im2=im2+300;
figure, imshow(im2);
¿Cuál es el resultado de max(max(im2))? ¿Por qué?

```

### 1.1 Manipulación básica de imágenes

- Escribir una función (ej11()), que lee la imagen *lena.jpg* y la visualiza (*figure, imshow*). ¿Qué tamaño tiene la imagen? Cuántos canales tiene? Convertir la imagen en escala de grises (Help: *rgb2grey*) (Figura 1a).
- Obtener información sobre su tamaño (Help: mirar el workspace y el comando *size*). Calcular el 10% de su tamaño.
- “Incrustar” la imagen original en una imagen negra con un borde negro de 10% de anchura/altura de la imagen original. ¿De qué tamaño ha de ser la imagen negra?
- Mostrar la imagen resultante y guardarla como “*lena\_frame.jpg*”.

**(Opcional)**

- Crear un borde rojo y guardar la imagen resultante como “*lena\_red\_frame.jpg*”.
- Crear un borde superior para la imagen, cuyo grosor sea el 10% de su tamaño. Para esto se ha de crear una matriz de tipo ‘*uint8*’, donde cada pixel tenga valor 0 (Help: *zeros()*).
- Concatenar el borde superior con la imagen original. (Ver funciones *vertcat* y *horzcat*).
- Crear el resto de los bordes, inferior y laterales (Figura 3b).

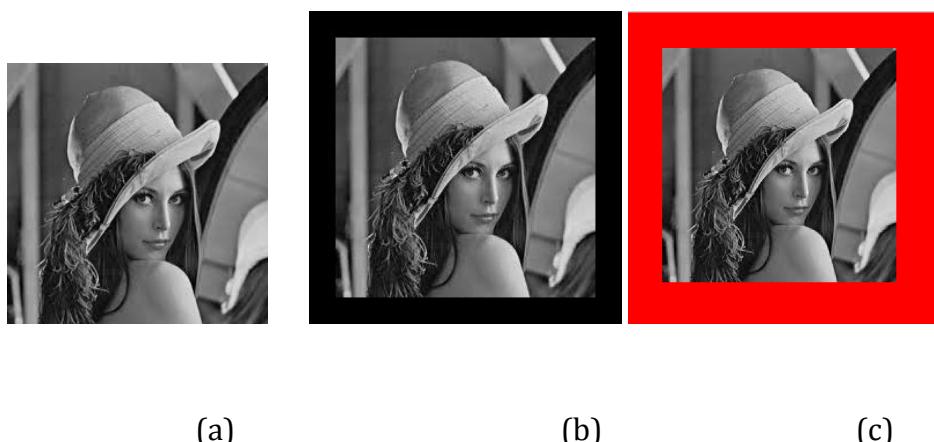


Figura 1 La imagen *lena.jpg* y la imagen con borde negro

## 1.2 Cambios de color y contraste

- a) Crear la función ej12() que abre la imagen *car\_lowContrast.jpg* y la aclara de forma constante sin saturarla, de forma que el píxel más claro sea blanco (Figura 3 a)). Guardar la imagen como “*car\_bright.jpg*”. Visualiza y compara el histograma (help: *imhist()*) de las dos imágenes y explica en qué se diferencian y en qué se parecen.
- b) Abre la imagen *car\_lowContrast.jpg* y la oscurece de forma constante sin saturarla, de forma que el píxel más oscuro sea negro (Figura 3 b)). Guarda la imagen como “*car\_dark.jpg*”. Visualiza y compara el histograma (help: *imhist()*) de las dos imágenes y explica en qué se diferencian y en qué se parecen.
- c) A continuación aumenta el contraste de la imagen *car\_lowContrast.jpg*. Primeramente, abre la imagen y la convierte a tipo double con valores entre 0 y 1 (dividir todos los valores entre 255). Finalmente aumenta el contraste (Figura 3 c)). Guarda la imagen como “*car\_contrast.jpg*”. Visualiza y compara el histograma (help: *imhist()*) de las dos imágenes y explica en qué se diferencian y en qué se parecen.

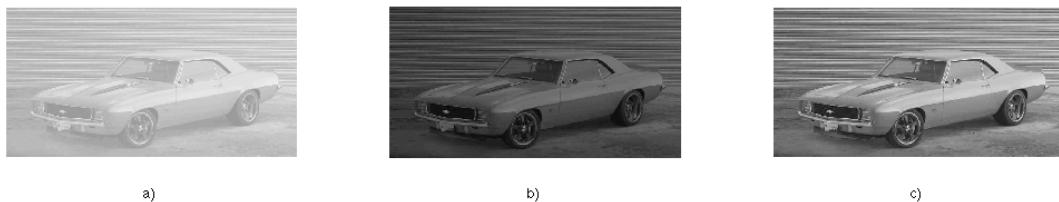


Figura 4 La imagen *car\_lowContrast.jpg*

## 1.3 Binarización de imágenes

La binarización  $B_I(x,y)$  de una imagen  $I(x,y)$  a partir de un umbral ( $Th$ ) consiste en convertir la imagen en una imagen binaria (de 0s y 1s), dependiendo de si el nivel de intensidad de la imagen original es mayor o menor al umbral  $Th$ .

$$B_I(x,y) = \begin{cases} 0, & \forall(x,y): I(x,y) < Th \\ 1, & \forall(x,y): I(x,y) \geq Th \end{cases}$$



**Figura 5: La imagen b) representa la binarización de a) utilizando un umbral de Th=128**

Dada la imagen *car\_gray.jpg*, crear la función *ej13()* que implemente los siguientes puntos:

- Crea la versión binaria de la imagen original poniendo un umbral de valor 130 y usando la indexación lógica. ¿Qué pasa si utilizamos un valor de umbral diferente? ¿Por qué? Modificar la función que se pueda llamar con diferentes umbrales.
- Visualiza las dos imágenes en la misma figura (Help: subplot).

Ejemplo: figure, subplot(1,2,1), imshow(...), subplot(1,2,2), imshow(...);

#### (Opcional)

- Junta las dos imágenes como una imagen más grande compuesta de las dos y guárdala como *car\_binary.jpg*.
- Encuentra el comando de binarización (*thresholding*) en Matlab y reimplementa con este comando la función *ej13()*. Help: puedes usar el *lookfor*.

### 1.4 Trabajar con imágenes binarias y de escalas de gris

Crear la función *ej14()* que haga:

- Abrir la imagen *circles.bmp*, visualizarla y convertirla a escala de grises (Figura 6. a). Con *imtool(im)* puedes ver el valor de los píxeles de los círculos.
- Crear 3 imágenes tipo binario a partir de la imagen inicial. En cada una de ellas, se debe representar el fondo negro y cada uno de los 3 círculos en blanco individualmente, utilizando los umbrales de valor correspondiente a los tres círculos (Figura 6. b). Visualizar las tres imágenes en la misma figura usando el subplot.
- Utiliza las matrices binarias creadas en b) para generar 3 imágenes, donde aparezcan cada uno de los círculos con su nivel de gris original. (Figura 6. c)
- Combina las 3 imágenes de forma que se obtenga la imagen mostrada en la Figura 6. c y visualízalos.

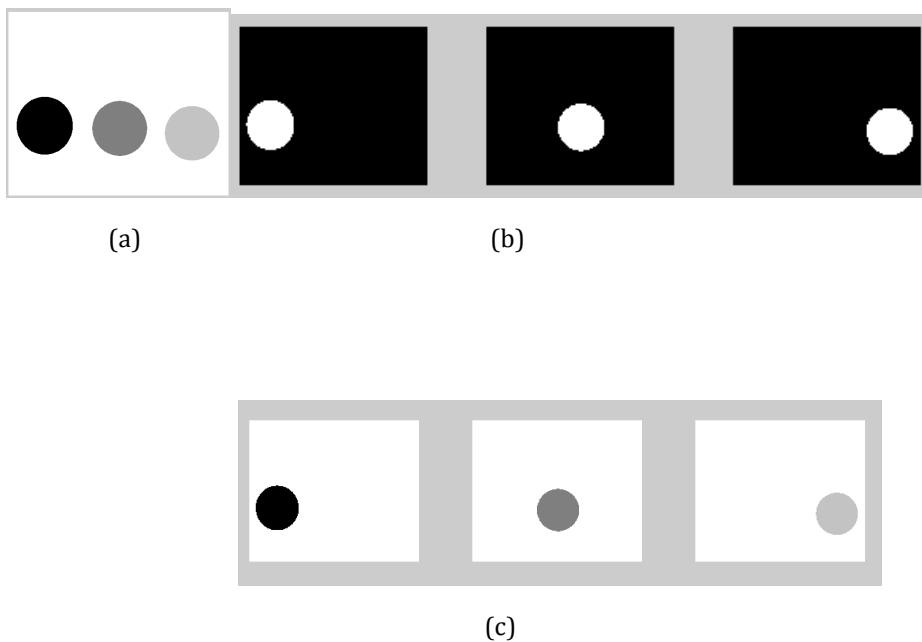


Figura 6 La imagen circles.bmp

## 1.5 Visualización de imágenes en color

Ler la imagen 'sillas.jpg'. Visualizar los diferentes canales y explicar la diferencia entre los valores de los píxeles. ¿Qué sucede si en la imagen en color intercambiamos los canales? ¿Qué sucede si en la imagen en color multiplicamos uno de los canales por 0?

```

function [ ] = ej15(file)

%This function reads a color image and displays its channels


im=imread(file);

imR=... % define the Red channel
imG=... % define the Green channel
imB=... % define the Blue channel


figure, subplot(2,3,2), imshow(im),...
subplot(2,3,4), imshow(imR),...
subplot(2,3,5), imshow(imG),...
subplot(2,3,6), imshow(imB);

...
end

```



Figura 7 La imagen sillas.jpg

### 1.6 Creación de imágenes de 3 canales (en color)

Las imágenes RGB están formadas por 3 matrices, llamadas comúnmente canales.

- a) Crea las 3 imágenes en escala de grises mostradas en la figura 8 (arriba).
- b) Combina las 3 imágenes de forma que se obtenga la imagen mostrada en la figura 8 (abajo).
- c) Guarda la imagen resultante como *3channels.jpg*  
*(Opcional)*
- d) Leer la imagen altalena.png
- e) Extraer sus componentes
- f) Visualizar la imagen RGB en una figura y sus imágenes componentes en otra con los títulos apropiados
- g) Concatenar las 3 imágenes componentes para formar una imagen RGB. Explorar lo que ocurre al cambiar el orden de la concatenación.

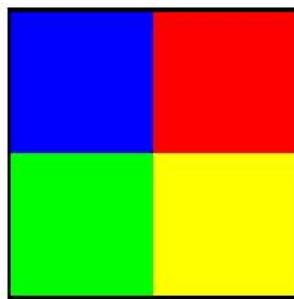
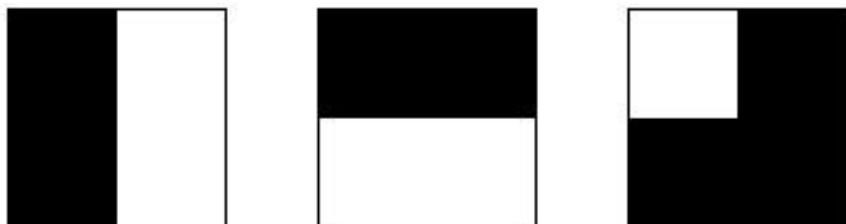


Figura 8 Imagen sintética en color



Figura 9 Sobreposición de imágenes

### 1.7 (Opcional) Tratamiento de imágenes en color RGB: modificación de una imagen

Dada la imagen logo.jpg, crear la función *ej17()* que implemente los siguientes puntos:

- Leer la imagen logo.jpg
- Encontrar el vector de todos los pixeles que tiene color verde (6,118,85). Para ello utilizar la función **find**.
- Asignar a todos estos pixeles color rojo (255,0,0)
- Visualizar las dos imágenes en una figura de la forma siguiente:

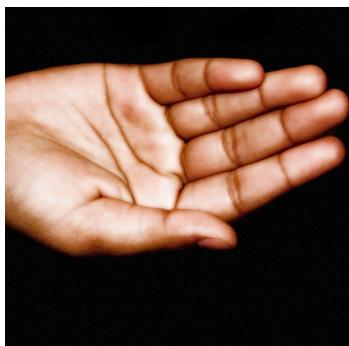


Figura 10 La imagen logo

### 1.8 Tratamiento de imágenes en color RGB

Dados los archivos de imagen *hand\_rgb.jpg* (figura 9 a) y *mapfre.jpg* (figura 9 b), crear la función *ej18()* que implemente los siguientes puntos:

- a) Abrir los 2 archivos en MATLAB. ¿Cuántas dimensiones tienen las imágenes?
- b) Convertir la imagen *hand\_rgb.jpg* en escala de grises utilizando la función de MATLAB apropiada.
- c) Realizar una binarización sobre la imagen resultante en b) para conseguir 2 regiones: una perteneciente a la mano (foreground) y la otra al fondo (background). Crear otra imagen binaria invirtiendo las zonas de foreground y background.
- d) Utilizar las matrices binarias creadas en c) para fusionar la imágenes *hand\_rgb* y *Mapfre* (fig. 11c)
- e) Guardar la imagen resultante como *hand\_mapfre\_3C.jpg*



(a)



(b)



(c)

Figura 11 Sobreposición de imágenes

### Entrega de la práctica

Tiempo máximo de entrega: 7 de octubre a las 13:00h.

Material para entregar: archivo “nombres\_apellidos\_P1.zip” que contenga:

- Archivos .m de cada ejercicio junto con las funciones creadas. **El código debe estar comentado en inglés.**