PROCESAMIENTO DE IMÁGENES

PRÁTICA 0: INTRODUCCIÓN A MATLAB

**Santiago Ponce Arrocha**

**INSTRUCCIONES**

Entrega un pdf con los apartados resueltos (captura del script y los resultados) de esta práctica.

**OPERACIONES GENERALES**

1. Comentarios

% esto es un comentario

%% permite la división del código en partes que se pueden ejecutar por separado

1. Operaciones aritméticas con números

pi\*10^2

1. Definición de variables

a = 2+5;

La podemos visualizar en la sección “Workspace” donde se indica también el tamaño y tipo.

1. Variables con cadenas de texto y salida por consola:

tempText = “Temperature is “ + a + “ºC”

disp(‘hello world’)

disp(“Temperature is “ + a + “ºC”)

clc % limpia la ventana de comandos

1. Comando help

help

1. Guardar y recuperar variables:

save(“a.mat”,”a”)

load(“a.mat”)

1. Creación de matrices:

ones, zeros, magic

**Busca información de cómo usar los comandos y pruébalos. ¿Qué diferencias hay?**

Ones / zeros crea matrices enteras de 1s y 0s. Magic crea una matriz cuadrada de tamaño especificado donde la suma de los elementos en cada fila, cada columna y cada diagonal es la misma.

1. Recorrido de matrices: secuencial, por intervalos, columnas, filas…

A = [1 2 3 4; 5 6 7 8];

A(1:2,2)

A(2,:)

B = 0:10:100

1. Submatrices y concatenación de matrices:

C = [A; A]

C = [A A]

C(:,5:8) = []

1. Operaciones aritméticas con matrices: +,-,\*, .\*, ./, .^

B = 3\*A

B = A.\*A

C = A(1:2,3:4)\* A(1:2,1:2)

**¿Qué diferencia hay entre el operador \* y el .\*?**

‘\*’ multiplica matrices, ‘.\*’ multiplica elemento por elemento.

1. Estructuras de control:

if-elseif-else-end

while-end

for i=ini:paso:fin end

**Prueba a crear un bucle donde se muestre el resultado de una operación en la ventana de comandos.**

**A screenshot of a computer code

Description automatically generated**

1. Operaciones lógicas:

&& || ~ < <= == > >=

**Prueba a crear un bucle donde se muestre el resultado de una operación en la ventana de comandos cuando se cumpla una condición.**

A screenshot of a computer code

Description automatically generated

**Prueba a crear un bucle donde se muestre el resultado de una operación en la ventana de comandos cuando NO se cumpla una condición.**

**A white background with black and blue text

Description automatically generated**

1. Borrado:

clear variables

clear all

**¿Qué diferencia hay entre usar “all” y “variables”?**

En la primera borramos variables simplemente, mientras que con ‘all’ borramos todas aquellas tal que cumplan una condición. EJ: clear all([1,2,3] > 0) devolverá true y borrará.

close all % cierra las figuras (ventanas emergentes) que haya

1. La instrucción ver indicará las Toolboxes instaladas y las versiones correspondientes:

**MANEJO BÁSICO DE IMÁGENES**

En este punto es conveniente guardar las imágenes que vamos a utilizar en la carpeta de trabajo actual. Ya que la primera instrucción que se empleará será la lectura de ficheros de imágenes (imread( )):

imgEnt = imread('cameraman.tif');

imgEnt representa el identificador para la imagen que queremos leer. Para buscar ayuda sobre las funciones o comandos se emplea la instrucción help:

help imread

Si queremos obtener más información sobre las variables, esta puede obtenerse visualizando la ventana WORKSPACE. También puede utilizarse la instrucción whos:

whos

Podemos comprobar el formato de la imagen, niveles de grises, la clase así como el tamaño de la misma, a modo de ejemplo la información será del tipo (256 valores diferentes (de la clase uint8) y tamaño de 256 x256 píxeles). Para la visualización de la imagen utilizamos la instrucción imshow:

imshow(imgEnt)



El tipo de dato matriz que contendrá una imagen puede ser de varios tipos (según el tipo de dato de cada pixel):

* **double**: Doble precisión, números en punto flotante que varían en un rango aproximado de -10308 a 10308 (8 bytes por elemento)
* **uint8**: Enteros de 8 bits en el rango de [0,255] (1 byte por elemento)
* **uint16**: Enteros de 16 bits en el rango de [0, 65535] (2 bytes por elemento)
* **uint32**: Enteros de 32 bits en el rango de [0, 4294967295] (4 bytes por elemento)
* **int8**: Enteros de 8 bits en el rango de [-128, 127] (1 byte por elemento)
* **int16**: Enteros de 16 bits en el rango de [-32768, 32767] (2 bytes por elemento)
* **int32**: Enteros de 32 bits en el rango de [-2147483648,2147483647] (4 bytes por elemento)
* **logical**: Los valores son 0 o 1 (1 bit por elemento)

**EJERCICIOS**

**Ejercicio 1. Herramienta imtool**

Repita estas operaciones con otra imagen ‘bacteria.tif’. **¿Cuál es su tamaño? ¿Cuántos niveles de grises puede tener? ¿Cuál es el valor mínimo y el máximo?** Si la imagen fuese en color, normalmente quedará definida por tres matrices correspondiente a los tres colores básicos (rojo, verde y azul).

A computer screen shot of a computer code

Description automatically generated

*La imagen tiene un tamaño cuadrado de 178x178, tiene 8 bits en el rango de [0,255] grises. A white background with text and numbers

Description automatically generated*

*Sus valores mínimos y máximos de grises son 0 y 239 respectivamente.*

**Vuelva a realizar las mismas operaciones de: a) lectura, b) tamaño y clase de la imagen y c) visualización sobre una imagen de color 'flowers.tif'.**

**A white background with black text

Description automatically generated**

***Tamaño****:362x500* ***Tipo****: uint8*

Utilizando la notación de matrices de Matlab se pueden visualizar las tres componentes del color. El operador : hace referencia a todos los elementos de esa dimensión, luego el nivel de gris para cada parte del espectro de la luz será definido por (:,:,i).

Indica que todas las filas y las columnas para la componente i, i=1,2 o 3 según se trate de los colores (rojo, verde o azul):

imshow([imgEnt(:,:,1),imgEnt(:,:,2),imgEnt(:,:,3)]);

El operador [ ] permitirá construir una matriz de N x (3\*M), siendo N el número de filas y M el número de columnas.

Emplee el comando imtool para ver el nivel de gris de la imagen de ‘nombre.tif’ y los colores en ‘nombre.tif’. Utilice el inspector de valores de los píxeles: imtool('flowers.tif');

**Prueba a visualizar e inspeccionar 3 imágenes distintas.**

**A screenshot of a black and white photo of flowers

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

**Ejercicio 2. Conversión de imágenes**

Otra opción interesante para tratamiento de imágenes es el formato en binario. Normalmente se emplea el ‘0’ para indicar el fondo y ‘1’ para el objeto. En muchas ocasiones habrá que hacer una conversión.

Conversión entre tipos de datos: Para ciertas operaciones es necesario convertir una imagen de su tipo original a otro tipo de imagen que facilite su procesamiento. Algunos métodos son:

* **gray2ind** Crea una imagen indexada a partir de una imagen de intensidad en escala de gris.
* **im2bw** Crea una imagen binaria a partir de una imagen de intensidad, imagen indexada o RGB basada en el umbral de luminancia.
* **ind2rgb** Crea una imagen RGB a partir de una imagen indexada.
* **rgb2gray** Crea una imagen de intensidad en escala de gris a partir de una imagen RGB
* **rgb2ind** Crea una imagen indexada a partir de una imagen RGB.

Por lo tanto, se emplea una técnica de umbralización para convertir las imágenes en binarias (im2bw()):

imgEntGris = imread(fullfile(cd,path,"rice.tif"));

figure(1);

imshow(imgEntGris);

impixelinfo

imgBW = im2bw(imgEntGris);

figure(2);

imshow(imgBW);

impixelinfo

**Realice la misma operación de binarización (umbralización) con la imagen ‘coins.png’.**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

A screenshot of a computer program

Description automatically generated

**Convierta a imagen indexada una imagen en color:**

A computer code with colorful text

Description automatically generated**A screenshot of a computer

Description automatically generated**

**Ejercicio 3. Generando un fichero \*.m**

En este apartado se tratará de realizar la primera función (\*.m) de procesamiento de imágenes con Matlab. Consistirá en leer un fichero de imagen 2D, cuyo nombre es pasado por parámetro, se visualizará y se aplicará una umbralización automática, la cual es también visualizada.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Modifica la función para que, además, devuelva ambas imágenes.**

**A screenshot of a computer

Description automatically generated** **A screenshot of a computer

Description automatically generated**

A screenshot of a computer

Description automatically generated

**Ejercicio 4. Construcción de imágenes**

**Construir una imagen binaria de 120 x 200 píxeles que tenga franjas horizontales de 20 píxeles de anchura, distanciada por cada 20 píxeles:**

A white background with black text

Description automatically generatedA screenshot of a computer

Description automatically generated

**Si queremos que las franjas sean verticales sólo habría que emplear el operador traspuesta de las matrices:**

A white background with black text

Description automatically generatedA screenshot of a computer

Description automatically generated

**Ejercicio 5.**

**Realizar una función que construya y visualice dos imágenes de 256x256 con variación del nivel de gris en filas y columna as.**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated