

UNIVERSIDAD INDUSTRIAL DE SANTANDER.

PABLO JOSUE ROJAS YEPES.

PROCESAMIENTO DE IMÁGENES DIGITALES.

LABORATORIO 1: FUNDAMENTO DE PROCESAMIENTO DIGITAL DE IMÁGENES.

1. Objetivo

Comprender los principales conceptos del entorno MATLAB así como los comandos básicos IPT Toolbox de procesamiento de imágenes (Image Processing Toolbox por sus siglas en ingles).

2. Introducción.

3. Experimentos:

- 1) Cargue la imagen coins.png, ejecutando el siguiente comando:

```
I = imread('coins.png');
```

Encontrar: Tamaño en bytes. Numero de pixeles. Numero de bits por pixel.

```
5 %%-----
6 %%Punto 1.
7 - I=imread('coins.png');
8 - whos I
9 % imwrite(I, 'coins.png');
10 % imwrite(I, 'coins.jpg');
11 % imwrite(I, 'coins5.jpg', 'quality', 5);
12
13
14 %%-----
```

Command Window




Name	Size	Bytes	Class	Attributes
I	246x300	73800	uint8	

Ahora: Ejecute el siguiente script y compare los tamaños de archivos

```
imwrite(I, 'coins.png');
```

```
imwrite(I, 'coins.jpg');
```

```
imwrite(I, 'coins5.jpg', 'quality', 5);
```

Nombre	Tipo	Tamaño	Dimensiones
 coins	Imagen JPEG	9 KB	300 x 246
 coins	Imagen PNG	37 KB	300 x 246
 coins5	Imagen JPEG	2 KB	300 x 246

- 2) Cargue la imagen 'parrots.tif' y examine las características de la imagen con el comando whos. **Nota:** leer acerca del comando **colormap**.

```
[I, map] = imread('parrots.tif');
```

- Almacene cada canal de color en diferentes variables y gráfíquelas con el comando **imagesc** y defina un **colormap** para cada canal.
- Genere un **colormap** de N entradas mediante la aproximación del **colormap** de la imagen original almacenado en la variable **map**. Grafique la imagen con **imagesc** y el nuevo **colormap**. ¿Cuántos colores son

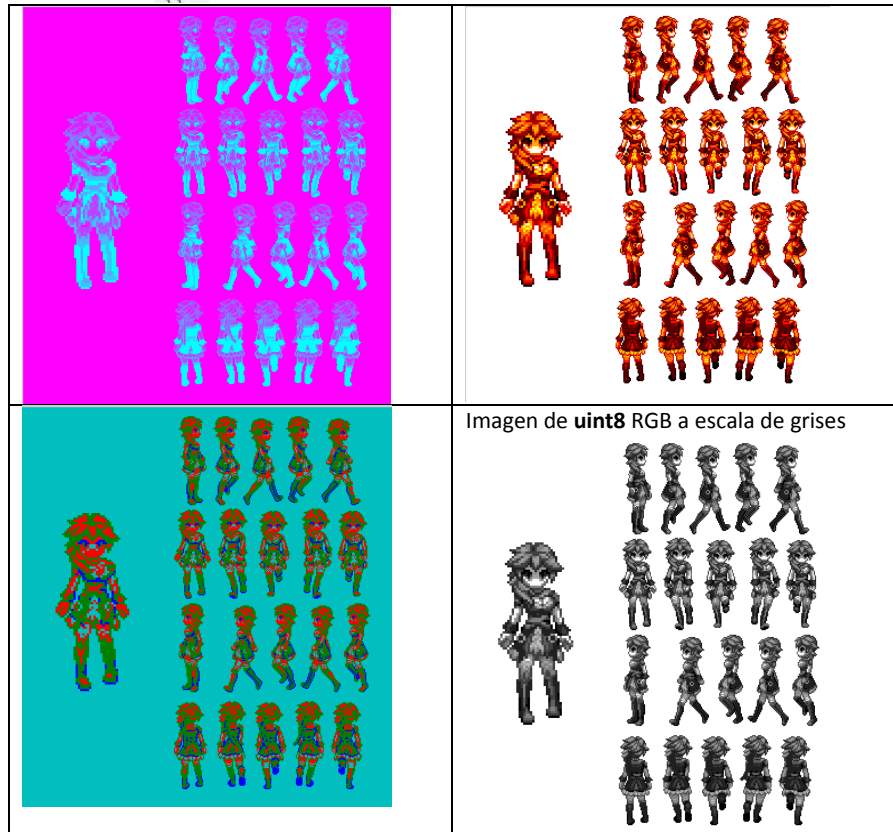
necesarios para ver una buena imagen? Por lo que entendí con dos colores se puede generar una img.

- Convertir la imagen de **uint8** RGB a escala de grises y grafique con el comando **imagesc** y **colormap(gray)**.
- Ahora cambiar el rango [0 255] a un rango más bajo [0 N]. Grafique y examine el resultado para diferentes valores de N con el comando **whos**. ¿Hasta que valor de N no existe distorsión en la imagen? Un poco más de la mitad.

```

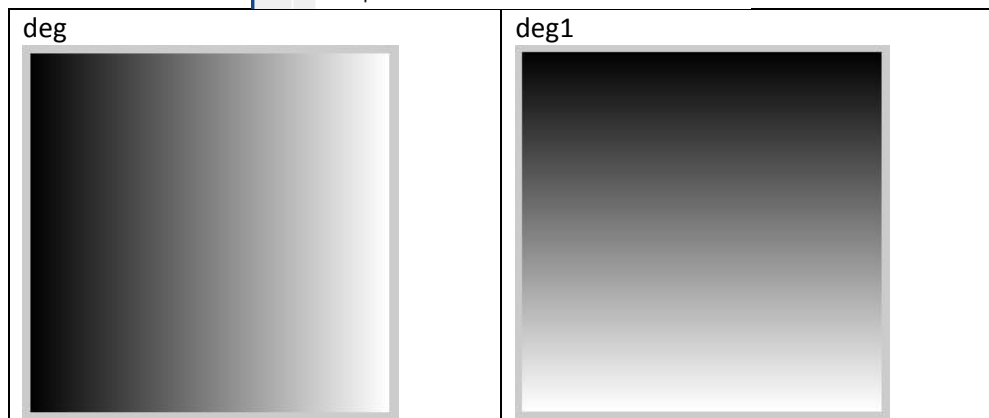
13 %-----
14 %%Punto 2.
15 [I, map] = imread('sprite.tif');
16 whos I
17 R=I(:,:,1);
18 G=I(:,:,2);
19 B=I(:,:,3);
20 figure,imshow(R);
21 colormap('cool');
22 figure,imshow(G);
23 colormap('hot');
24 figure,imshow(B);
25 colormap('lines');
26 figure,imshow(G);
27 colormap('grey');
28 cmap1=zeros(150,3);
29 cmap1(:,1)=(0:149)';
30 cmap=im2double(cmap1);
31 figure,imshow(G);
32 colormap(cmap./149);
33

```



- 3) Realizar una función en matlab que construya y visualice dos imágenes de 256x256 con variación de N niveles de grises en las y columnas.

```
34 %-----
35 %%Punto 3.
36 deg=ones(256,256);
37 deg1=ones(256,256);
38 for i=1:256
39     for j=1:256
40         deg(i,j)=j/256;
41     end
42 end
43 for j=1:256
44     for i=1:256
45         deg1(i,j)=i/256;
46     end
47 end
48 figure,imshow(deg);
49 figure,imshow(deg1);
50
```



- 4) Escribir una función en Matlab que cree una figura sobre una matriz binaria. La figura puede ser un cuadrado, un rectángulo, un triángulo o un círculo. La función debe cumplir el siguiente formato:

```
Function [I] = figura(stringfigura)
% Genera una imagen binaria con la forma 'stringfigura';
% -----
% stringfigura = 'cuadrado', 'circulo', 'triangulo', 'rectangulo' o 'todos';
end
```

La salida es de la forma:

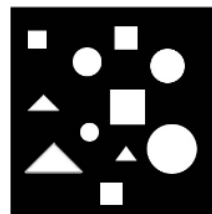
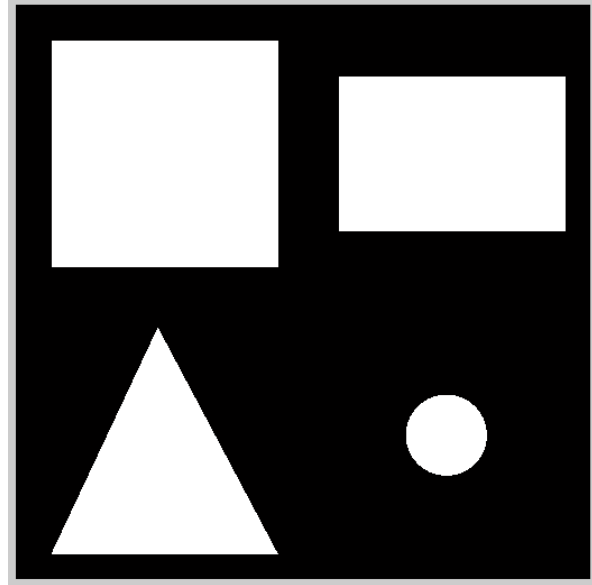


Figura 2: Matriz binaria

```

1 function [I]=figura(stringfigura)
2
3 switch stringfigura
4     case 'cuadrado'
5         x=[32 234 234 32];
6         y=[32 32 234 234];
7         I=poly2mask(x,y,256,256);
8     case 'rectangulo'
9         x=[32 234 234 32];
10        y=[64 64 202 202];
11        I=poly2mask(x,y,256,256);
12    case 'triangulo'
13        x=[127 234 32];
14        y=[32 234 234];
15        I=poly2mask(x,y,256,256);
16    case 'circulo'
17        r=200;
18        n=256;
19        imagen=zeros(n);
20        x=linspace(-50,50,n);
21        y=linspace(-50,50,n);

```



- 5) Usando la función creada en el punto anterior, genere las mismas figuras pero sin relleno (solo contorno).

```

1 function [I]=figura2(stringfigura)
2
3 switch stringfigura
4     case 'cuadrado'
5         x1=[32 234 234 32];
6         y1=[32 32 234 234];
7         x2=[35 231 231 35];
8         y2=[35 35 231 231];
9         I1=poly2mask(x1,y1,256,256);
10        I2=poly2mask(x2,y2,256,256);
11        I=I1-I2;
12    case 'rectangulo'
13        x1=[32 234 234 32];
14        y1=[64 64 202 202];
15        x2=[35 231 231 35];
16        y2=[67 67 199 199];
17        I1=poly2mask(x1,y1,256,256);
18        I2=poly2mask(x2,y2,256,256);
19        I=I1-I2;
20    case 'triangulo'
21        x1=[127 234 32];

```

