Catálogo de Algoritmos

Integrantes:

- Daniel Castro Campos - Carnet 2016165624
- Gustavo Segura Umaña - Carnet 2016089824
- Joseph Vargas Blanco - Carnet 2016200170

1 Semana 3: SVD e imágenes

1.1 Compresión de imágenes

```
clear;
clc;
close all;
pkg load image;

I_color = imread("WingedFigure.jpg");
A = I_color(:,:,1);

A =im2double(A);
val_sing=diag(A);
[m,n]=size(A);

[U,S,V] =svd(A);

r=50;
Ur= U(:,1:r);
Vr= V(:,1:r);
Sr= S(1:r,1:r);

B=Ur*Sr;
C= Vr';
Ar=B*C;
Ar = im2uint8(Ar);


subplot(1,2,1);
imshow(A)
title('Imagen original');
subplot(1,2,2);
imshow(Ar)
title('Imagen reducida');

error=norm(A-im2double(Ar),'fro');
```

1.2 Compresión de imágenes con Proyección Bilateral

```python
import numpy as np
from PIL import Image, ImageOps
import matplotlib.pyplot as plt

def imageToArray(filename:str, rgb:bool=False):
    img = Image.open(filename)
    if not rgb:
        img = ImageOps.grayscale(img)
    img_array = np.array(img)
    return img_array

def bilateral(A, r, p):
    if(type(p)==int and p>=2):
        Y2=np.array(np.random.randint(255, size=A.shape), dtype=float)
        for k in range(1,p+1):
            Y1=np.matmul(A,Y2)
            Y2=np.matmul(A.T,Y1)
        Q, R = np.linalg.qr(Y2)
        Qr=Q[ : , :r]
        B=np.matmul(A,Qr)
        C=Qr.T
        return (B,C)


A = imageToArray("WingedFigure.jpg")
B,C=bilateral(A,30,p=5)
Ar=np.matmul(B,C)

plt.subplot(1, 2, 1)
plt.title('Imagen original')
plt.imshow(A, cmap='gray', vmin=0, vmax=255)

plt.subplot(1, 2, 2)
plt.title('Imagen resultante')
plt.imshow(Ar, cmap='gray', vmin=0, vmax=255)
plt.show()
```

2 Semana 4: Operaciones geométricas e interpolación

2.1 Filtro de la mediana

```
clc; clear;
pkg load image;

A = imread('WingedFigureNoise.jpg');
A = A(:,:,1);

[m,n] = size(A);
```

```
B = zeros(m,n);


B(1, 1) = median(A(1:2, 1:2)(:));
B(m, 1) = median(A(m-1:m, 1:2)(:));
B(1, n) = median(A(1:2, n-1:n)(:));
B(m, n) = median(A(m-1:m, n-1:n)(:));

for x = 2:m-1
  B(x, 1) = median(A(x-1:x+1, 1:2)(:));
  B(x, n) = median(A(x-1:x+1, n-1:n)(:));
endfor

for y = 2:n-1
  B(1, y) = median(A(1:2,y-1:y+1)(:));
  B(1, y) = median(A(m-1:m,y-1:y+1)(:));
endfor

for x = 2:m-1
  for y = 2:n-1
    neighborhood = A(x-1:x+1, y-1:y+1);
    temp = neighborhood(:);
    B(x, y) = median(temp);
  end
end
B = uint8(B);


subplot(1, 2, 1);
imshow(A);
title('Imagen con ruido');

subplot(1, 2, 2);
imshow(B);
title('Imagen resultante');
```

2.2 Transformadas Afines Lineales: Rotación

```
import numpy as np
from PIL import Image, ImageOps
import matplotlib.pyplot as plt

def imageToArray(filename:str, rgb:bool=False):
    img = Image.open(filename)
    if not rgb:
        img = ImageOps.grayscale(img)
    img_array = np.array(img)
    return img_array

def rotation(A:np.array, angle:int):
    angle = np.radians(angle)
    result = np.zeros(A.shape, dtype=np.uint8)
```

```
    M, N = A.shape[0],A.shape[1]
    xc= M//2
    yc= N//2
    a0 = np.cos(angle)
    a1 = np.sin(angle)
    b0 = -np.sin(angle)
    b1 = np.cos(angle)
    for i in range(M):
        for j in range(N):
            new_i=round(b0*(j - xc) + b1 * (i - yc) + yc)
            new_j=round(a0*(j - xc) + a1 * (i - yc) + xc)
            if (new_i>=0 and  new_i<M and new_j>=0 and  new_j<N ):
                result[new_i][new_j]=A[i][j]
    return result
def medianFilterBN(A:np.array):
    result = np.zeros(A.shape, dtype=np.uint8)
    M, N = A.shape[0],A.shape[1]
    # esquinas
    result[0][N-1] = np.median([A[0][N-1],
                                A[1][N-1],
                                A[0][N-2]])
    result[M-1][0] = np.median([A[M-2][0],
                                A[M-1][0],
                                A[M-1][1]])
    result[M-1][N-1] = np.median([A[M-1][N-1],
                                  A[M-2][N-1],
                                  A[M-1][N-2]])
    result[0][0] = np.median([A[0][0],
                              A[0][1],
                              A[1][0]])
    for y in range(1,N-1):
        result[0][y] = np.median(A[0:2,y-1:y+1])
        result[M-1][y] = np.median(A[M-2:M,y-1:y+1])
    for x in range(1,M-1):
        result[x][0] = np.median(A[x-1:x+2,:2])
        result[x][N-1] = np.median(A[x-1:x+2,-2:])
        for y in range(1,N-1):
            result[x][y] = np.median(A[x-1:x+2,y-1:y+2])
    return result


A = imageToArray("WingedFigure.jpg")
B = rotation(A, 35)
C = medianFilterBN(B)


plt.subplot(1, 3, 1)
plt.title('Imagen original')
plt.imshow(A, cmap='gray', vmin=0, vmax=255)

plt.subplot(1, 3, 2)
plt.title('Imagen rotada')
plt.imshow(B, cmap='gray', vmin=0, vmax=255)

```

```
plt.subplot(1, 3, 3)
plt.title('Imagen rotada y filtrada')
plt.imshow(C, cmap='gray', vmin=0, vmax=255)
plt.show()
```

2.3 Transformadas Afines Lineales: Traslación

```
import numpy as np
from PIL import Image, ImageOps
import matplotlib.pyplot as plt

def imageToArray(filename:str, rgb:bool=False):
    img = Image.open(filename)
    if not rgb:
        img = ImageOps.grayscale(img)
    img_array = np.array(img)
    return img_array

def traslation(A:np.array, x:int, y:int):
    result = np.zeros(A.shape, dtype=np.uint8)
    M, N = A.shape[0],A.shape[1]
    for i in range(M):
        for j in range(N):
            new_i=i+y
            new_j=j+x
            if (new_i>=0 and  new_i<M and new_j>=0 and  new_j<N ):
                result[new_i][new_j]=A[i][j]
    return result

A = imageToArray("WingedFigure.jpg")
B = traslation(A, 50,200)


plt.subplot(1, 2, 1)
plt.title('Imagen original')
plt.imshow(A, cmap='gray', vmin=0, vmax=255)

plt.subplot(1, 2, 2)
plt.title('Imagen trasladada')
plt.imshow(B, cmap='gray', vmin=0, vmax=255)
plt.show()
```

2.4 Transformadas Afines No Lineales: Efecto Rippling

```
import numpy as np
from PIL import Image, ImageOps
import matplotlib.pyplot as plt

def imageToArray(filename:str, rgb:bool=False):
    img = Image.open(filename)
```

```
    if not rgb:
        img = ImageOps.grayscale(img)
    img_array = np.array(img)
    return img_array

def rippling(A:np.array,Ax=15,Ay=15,Lx=75,Ly=75):
    result = np.zeros(A.shape,dtype=np.uint8)
    M, N = A.shape[0],A.shape[1]
    for x in range(M):
        for y in range(N):
            x_new=round(x+Ax*np.sin(2*np.pi*y/Lx))
            y_new=round(y+Ay*np.sin(2*np.pi*x/Ly))
            if (x_new>=0 and  x_new<M and y_new>=0 and  y_new<N ):
                result[x_new][y_new]=A[x][y]
    return result

A = imageToArray("WingedFigure.jpg")
B=rippling(A,Ax=15,Ay=15,Lx=200,Ly=200)


plt.subplot(1, 2, 1)
plt.title('Imagen original')
plt.imshow(A, cmap='gray', vmin=0, vmax=255)

plt.subplot(1, 2, 2)
plt.title('Imagen transformada')
plt.imshow(B, cmap='gray', vmin=0, vmax=255)
plt.show()
```

```
import numpy as np
from PIL import Image, ImageOps
import matplotlib.pyplot as plt

def imageToArray(filename:str, rgb:bool=False):
    img = Image.open(filename)
    if not rgb:
        img = ImageOps.grayscale(img)
    img_array = np.array(img)
    return img_array

def rippling(A:np.array,Ax=15,Ay=15,Lx=75,Ly=75):
    result = np.zeros(A.shape,dtype=np.uint8)
    M, N = A.shape[0],A.shape[1]
    for x in range(M):
        for y in range(N):
            x_new=round(x+Ax*np.sin(2*np.pi*y/Lx))
            y_new=round(y+Ay*np.sin(2*np.pi*x/Ly))
            if (x_new>=0 and  x_new<M and y_new>=0 and  y_new<N ):
                result[x_new][y_new]=A[x][y]
    return result

def ripplingAnimation(A, filename="out.gif", minA=0, maxA=50, step=5,Lx=200,Ly=200):
    images=[]
```

```
    for a in range(minA, maxA, step):
        images.append(rippling(A,Ax=a,Ay=a,Lx=Lx,Ly=Ly))
    B_img=list(map(Image.fromarray, images))
    B_img[0].save(filename,save_all=True, append_images=B_img[1:], loop=0)

A_color=imageToArray("WingedFigure.jpg", rgb=True)
ripplingAnimation(A_color, filename="rippling.gif", minA=0, maxA=100, step=5,Lx=200,Ly=200)
```

3 Semana 5: Transformaciones a escala de grises

3.1 Transformacion Lineal

```
import cv2
import matplotlib.pyplot as plt
import numpy as np


image0 = cv2.imread("boat.jpg",0)
plt.subplot(1,2,1)
plt.imshow(image0, cmap='gray')

new_image = np.zeros(image0.shape, image0.dtype)
alpha = 1.8
beta = 0

for y in range(image0.shape[0]):
  for x in range(image0.shape[1]):
    new_image[y,x] = np.clip(alpha*image0[y,x] + beta, 0, 255)

plt.subplot(1,2,2)
plt.imshow(new_image, cmap='gray')
plt.show()
```

3.2 Transformacion Autocontraste

```
clc;
clear;
close all;

A = imread('boat_new.jpg');
subplot(1,2,1);
imshow(A)
title('Imagen original');

A = double(A);
[m,n] = size(A);
B = zeros(m,n);

% Valores del nuevo contraste
```

```
rmin = min(min(A)); % Extrae valor de las columnas
rmax = max(max(A));

alpha = 255 / (rmax - rmin);
beta = rmin;

B = alpha * (A - beta);


B = uint8(B);
subplot(1,2,2);
imshow(B)
title('Imagen Modificada');
```

3.3 Transformacion Negativo

```
import cv2
import matplotlib.pyplot as plt
import numpy as np


image0 = cv2.imread("boat.jpg",0)
plt.subplot(1,2,1)
plt.imshow(image0, cmap='gray')

new_image = np.zeros(image0.shape, image0.dtype)
a = -1
b = 255

for y in range(image0.shape[0]):
  for x in range(image0.shape[1]):
    new_image[y,x] = np.clip(a*image0[y,x] + b, 0, 255)

plt.subplot(1,2,2)
plt.imshow(new_image, cmap='gray')
plt.show()
```

3.4 Transformacion Gamma

```
import cv2
import matplotlib.pyplot as plt
import numpy as np

image0 = cv2.imread("boat.jpg")
plt.subplot(1,2,1)
plt.imshow(image0, cmap='gray')

image0.astype(np.double)
new_image = np.zeros(image0.shape, image0.dtype)
gamma = 1.9
```

```
c = 1

for y in range(image0.shape[0]):
  for x in range(image0.shape[1]):
    new_image[y,x] = np.clip(c * (image0[y,x] ** gamma), 0, 255)

new_image.astype(np.uint8)
plt.subplot(1,2,2)
plt.imshow(new_image, cmap='gray')
plt.show()
```

3.5 Transformacion Logaritmica

```
clc;
clear;
close all;

A = imread('log.jpg');
subplot(1,2,1);
imshow(A)
title('Imagen original');

A = im2double(A);
[m,n] = size(A);
B = zeros(m,n);

c = 0
for c = 0.1:0.1:5
  B = c.*log(1+A);
  subplot(1,2,2);
  imshow(B);
  title(['Imagen Modificada: c = ' num2str(c)]);
  pause(0.05);
endfor
```

4 Semana 6: Histogramas

4.1 Mostrar el histograma

```
from skimage import io
import matplotlib.pyplot as plt

# Cargar imagen
image = io.imread('peppers.jpg')

# Mostrar imagen
plt.subplot(2, 1, 1)
plt.title('Imagen')
plt.imshow(image, cmap='gray', vmin=0, vmax=255)
```

```
11
12  # Mostrar histograma
13  plt.subplot(2, 1, 2)
14  plt.title('Histograma')
15  plt.hist(image.ravel(), bins = 256, range=(0, 255))
16
17  plt.show()
```

4.2 Ecualización de histograma

```
1   clc
2   clear
3   close all
4
5   A = imread('peppers.jpg');
6   subplot(2,2,1);
7   imshow(A)
8   title('Imagen original');
9
10
11  % Calcular el histograma
12  h2 = zeros(256, 1);
13  [m, n] = size(A);
14  for i = 1:255
15    h2(i + 1) = sum(sum(A == i));
16  endfor
17  subplot(2,2,2);
18  bar(0:255, h2)
19  title('Histograma');
20  xlim([0 255])
21
22  % Calcular la distribucion acumulativa
23  [m, n] = size(A);
24  v_ac = zeros(256, 1);
25  for i = 0:255
26    v_ac(i + 1) = sum(h2(1:i+1)/(m*n));
27  endfor
28  subplot(2,2,3)
29  bar(0:255, v_ac)
30  title('Distribucion acumulativa');
31  xlim([0 255])
32
33  % Metodos de ecualizacion
34  B = zeros(m, n);
35  A = double(A);
36  for x = 1:m
37    for y = 1:n
38      B(x, y) = round(v_ac(A(x, y) + 1) * 255);
39    endfor
40  endfor
41
42  B = uint8(B);
```

```
subplot(2,2,4);
imshow(B)
title('Imagen ecualizada');
```

4.3 Estiramiento del histograma

```
from skimage import io
import matplotlib.pyplot as plt

# Cargar imagen
image = io.imread('sydney.jpg', True)
# Estirar imagen
image_stretched = [(255 - 0) / (image.max() - image.min())] * (image - image.min())

# Mostrar imagen original
plt.subplot(2, 2, 1)
plt.title('Imagen original')
plt.imshow(image, cmap='gray', vmin=0, vmax=255)

# Mostrar imagen con histograma estirado
plt.subplot(2, 2, 2)
plt.title('Imagen con histograma estirado')
plt.imshow(image_stretched, cmap='gray', vmin=0, vmax=255)

# Histograma original
plt.subplot(2, 2, 3)
plt.title('Histograma original')
plt.hist(image.ravel(), bins = 256, range=(0, 255))

# Histograma estirado
plt.subplot(2, 2, 4)
plt.title('Histograma estirado')
plt.hist(image_stretched.ravel(), bins = 256, range=(0, 255))

plt.show()
```

4.4 Reducción del histograma

```
from skimage import io
import matplotlib.pyplot as plt

# Cargar imagen
image = io.imread('peppers.jpg', True)
# Reducir imagen
max = 175
min = 100
image_stretched = ([(max - min) / (image.max() - image.min())]
                   * (image - image.min()) + min)

# Mostrar imagen original
```

```
plt.subplot(2, 2, 1)
plt.title('Imagen original')
plt.imshow(image, cmap='gray', vmin=0, vmax=255)

# Mostrar imagen con histograma estirado
plt.subplot(2, 2, 2)
plt.title('Imagen con histograma reducido')
plt.imshow(image_stretched, cmap='gray', vmin=0, vmax=255)

# Histograma original
plt.subplot(2, 2, 3)
plt.title('Histograma original')
plt.hist(image.ravel(), bins=256, range=(0, 255))

# Histograma estirado
plt.subplot(2, 2, 4)
plt.title('Histograma reducido')
plt.hist(image_stretched.ravel(), bins=256, range=(0, 255))

plt.show()
```

4.5 Especificación de histograma

```
clc
clear
close all

A = imread('peppers.jpg');
subplot(4,2,1);
imshow(A)
title('Imagen original');

B = imread('sydney.jpg');
subplot(4,2,2);
imshow(B)
title('Imagen estandar');


% Calcular el histograma original
h1 = zeros(256, 1);
[m, n] = size(A);
for i = 1:255
  h1(i + 1) = sum(sum(A == i));
endfor
subplot(4,2,3);
bar(0:255, h1)
title('Histograma original');
xlim([0 255])

% Calcular el histograma estandar
h2 = zeros(256, 1);
[m, n] = size(B);
```

```
for i = 1:255
  h2(i + 1) = sum(sum(B == i));
endfor
subplot(4,2,4);
bar(0:255, h2)
title('Histograma estandar');
xlim([0 255])

% Calcular la distribucion acumulativa
[m, n] = size(A);
v_ac = zeros(256, 1);
for i = 0:255
  v_ac(i + 1) = sum(h1(1:i+1)/(m*n));
endfor
subplot(4,2,5)
bar(0:255, v_ac)
title('Distribucion acumulativa original');
xlim([0 255])

% Calcular la distribucion acumulativa
[m2, n2] = size(B);
v2_ac = zeros(256, 1);
for i = 0:255
  v2_ac(i + 1) = sum(h2(1:i+1)/(m2*n2));
endfor
subplot(4,2,6)
bar(0:255, v2_ac)
title('Distribucion acumulativa estandar');
xlim([0 255])

% Metodos de ecualizacion
C = zeros(m, n);
A = double(A);
for x = 1:m
  for y = 1:n
    C(x, y) = round(v2_ac(A(x, y) + 1) * 255);
  endfor
endfor

C = uint8(C);
subplot(4,1,4);
imshow(C)
title('Imagen original con especificacion de histograma estandar');
```