UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN

COMPUTACION GRAFICA

Practica 9.1 y Practica 9.2

Alumnos:
Chayña Batallanes Josnick
Perez Rodriguez Angelo Aldo
Pucho Zevallos Kelvin Paul
Vilcapaza Flores Luis Felipe
Sihuinta Perez Luis Armando

Julio 2021

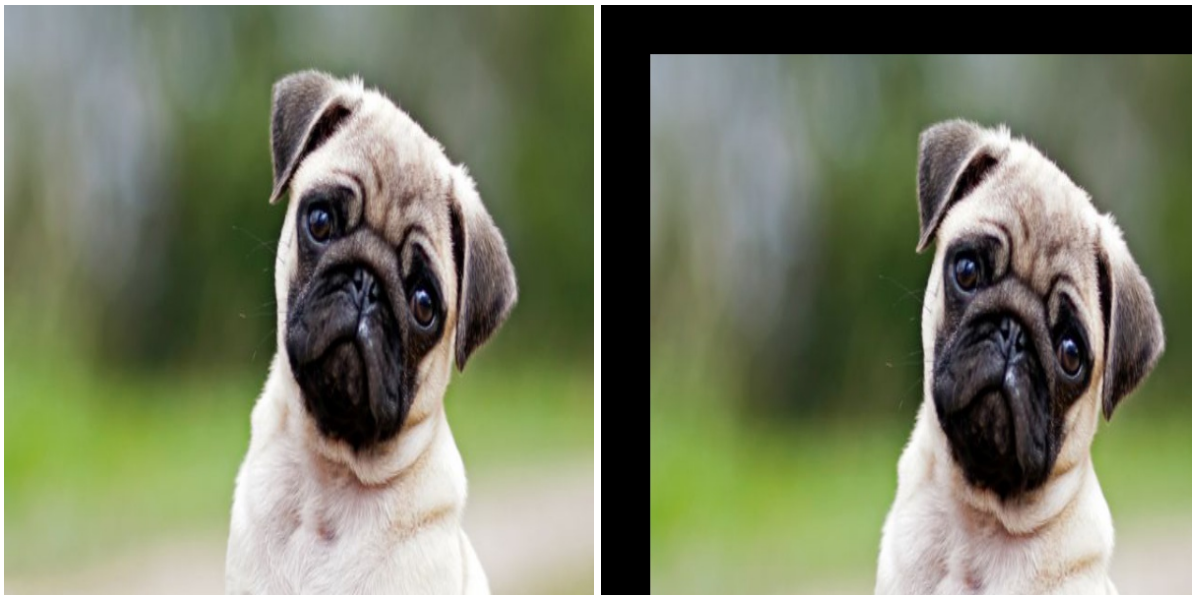
Índice

1. Algoritmo de computación gráfica Affine transformation (Practica 9.1)).

1.1. Utilice la función cv2.warpAffine(...) de OpenCV para obtener una traslación, escala, rotación y shear. Usted tiene que definir la matriz M y puede utilizar cualquier imagen.

- **Traslacion de la imagen usando la funcion cv2.warpAffine(...)**

  - **Imagen Original y Resultado de la imagen**:



(a) Imagen 1 (b) Imagen 2

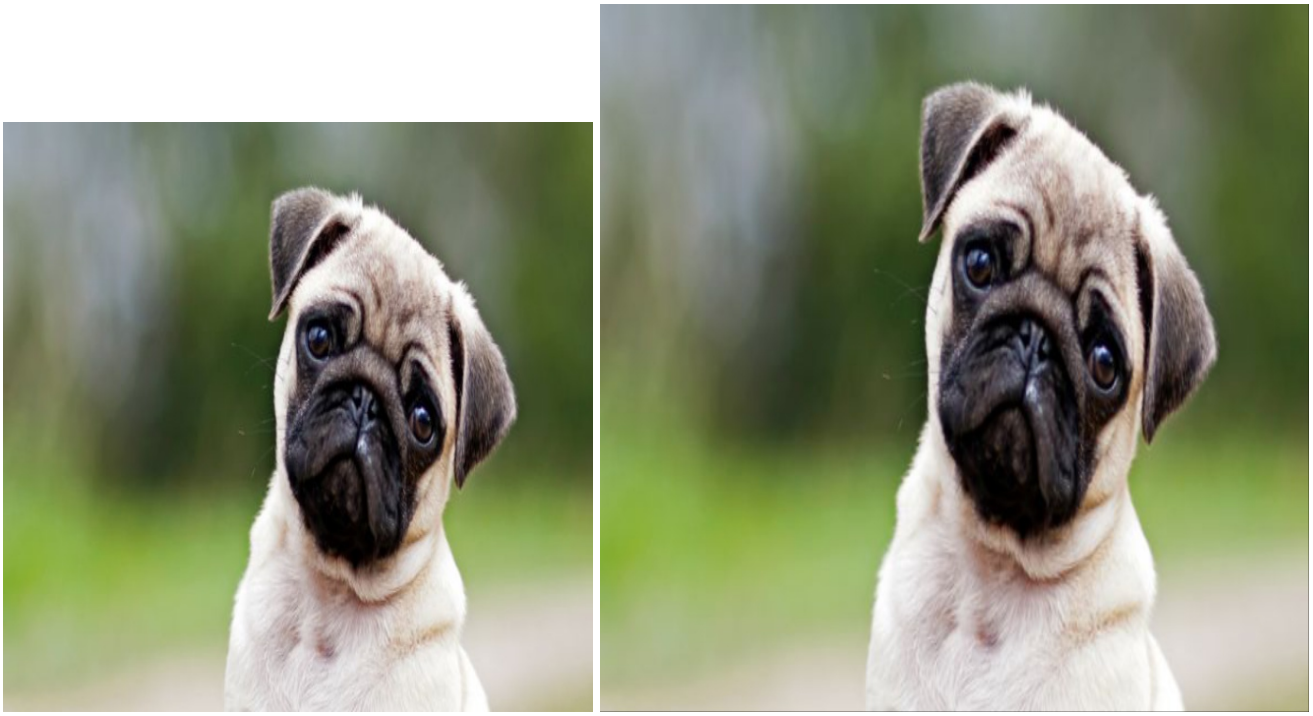
Figura 1: Traslación

- **Código**:

```
import cv2
import numpy as np

# Read the image
img = cv2.imread('pug.jpg',1)
img = cv2.resize(img,(600,600))
rows, cols = img.shape[:2]
cv2.imshow('Imagen Original', img)

# Matriz M

M = np.float32([[1,0,50], [0,1,50]])
print(M)

# Apply the affine transformation using cv2.warpAffine()
imgResult = cv2.warpAffine(img, M, (cols,rows))
 a
cv2.imshow('Result', imgResult)

filename = 'TraslacionPug.jpg'
cv2.imwrite(filename, imgResult)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

- **Escalamiento de la imagen usando la función cv2.warpAffine(...)**

  - **Imagen Original y Resultado de la imagen**:



(a) Imagen 1 (b) Imagen 2

Figura 2: Escalación

- **Código**:

```
import cv2
import numpy as np

# Read the image
img = cv2.imread('pug.jpg',1)
img = cv2.resize(img,(600,600))
rows, cols = img.shape[:2]
cv2.imshow('Imagen Original', img)

# Matriz M
M = np.float32([[2,0,0], [0,2,0]])
print(M)

# # Apply the affine transformation using cv2.warpAffine()
imgResult = cv2.warpAffine(img, M, (cols*2,rows*2))

cv2.imshow('Output', imgResult)
filename = 'EscalarPug.jpg'
cv2.imwrite(filename, imgResult)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

- **Rotación de la imagen usando la función cv2.warpAffine(...)**

- **Imagen Original y Resultado de la imagen**:

(a) Imagen 1 (b) Imagen 2

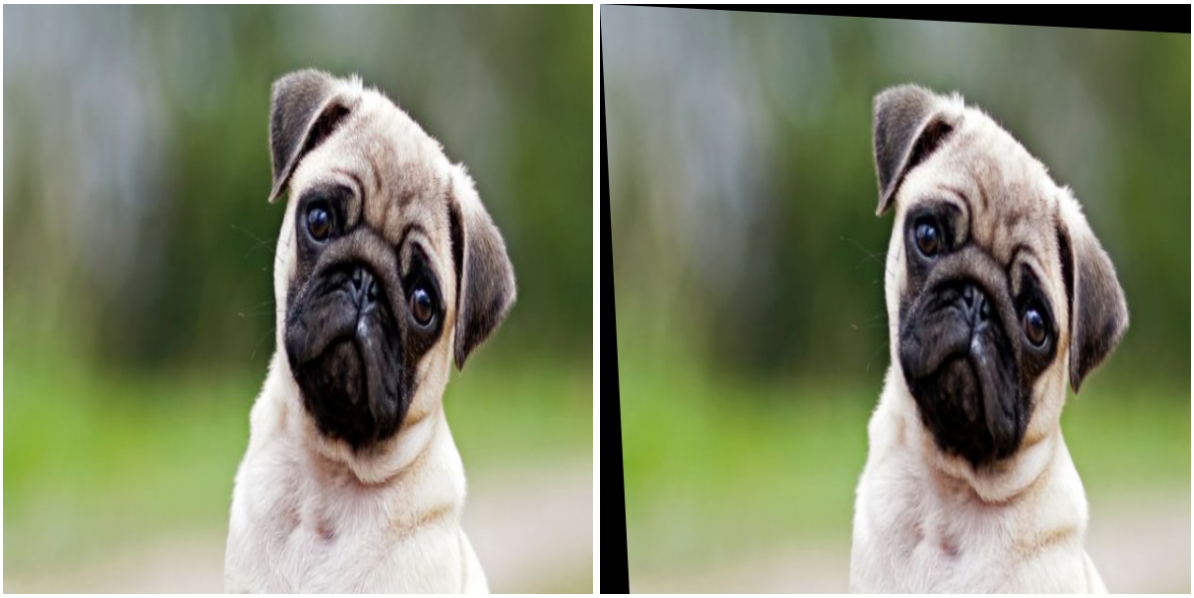
Figura 3: Rotación

- **Código**:

```
import cv2
import numpy as np
import math

# Reading the img
img = cv2.imread('pug.jpg',1)
img = cv2.resize(img,(600,600))

# matriz M
angulo = 45
px= 300.0
py= 300.0
M = np.float32([[math.cos(angulo),math.sin(angulo),((1-math.cos(
    angulo)) * px) - (math.sin(angulo) * py)],
                [-math.sin(angulo),math.cos(angulo),(math.sin(angulo)
                    * px) + ((1-math.cos(angulo)) * py)]])
# print(M)
# rotate the img using cv2.warpAffine
rotated_img = cv2.warpAffine(src=img, M=M, dsize=(width, height))

cv2.imshow('imgn Original ', img)
cv2.imshow('imgn Rotada ', rotated_img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

- **Shear de la imagen usando la funcion cv2.warpAffine(...)**

  - **Imagen Original y Resultado de la imagen**:



(a) Imagen 1 (b) Imagen 2

Figura 4: Shear

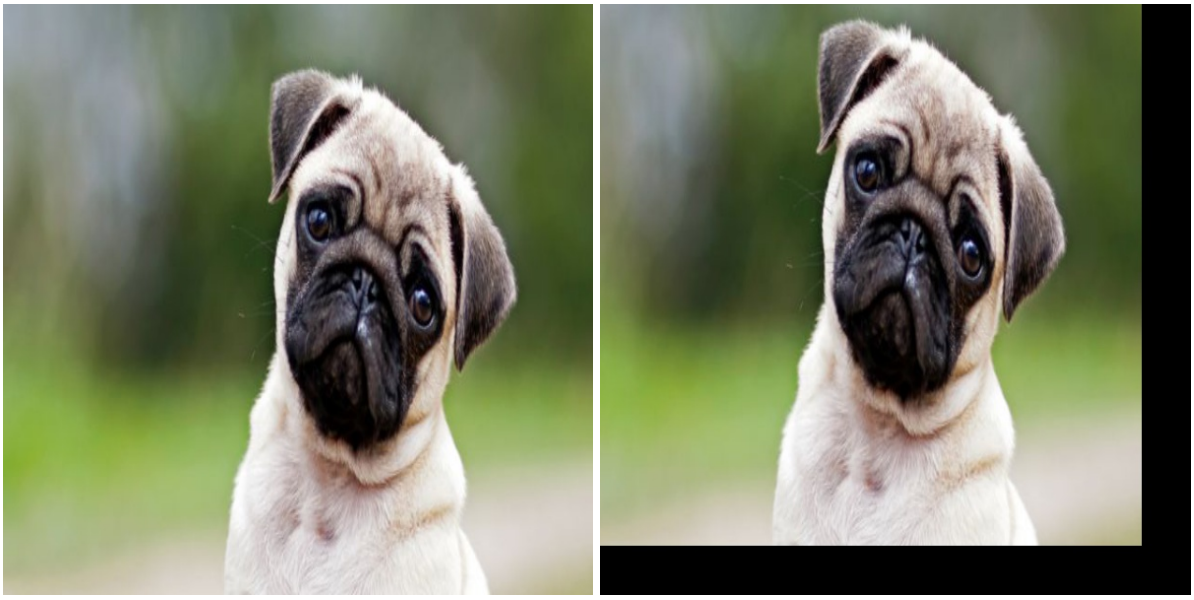
- **Codigo**:

```
import cv2
import numpy as np
import math

# Reading the img
img = cv2.imread('pug.jpg',1)
img = cv2.resize(img,(600,600))
# dividing height and width by 2 to get the center of the img
height, width = img.shape[:2]

shx = 0.05
shy = 0.05
M = np.float32([[1,shy,0],
                [shx,1,0]])
print(M)

# # Apply the affine transformation using cv2.warpAffine()
imgShear = cv2.warpAffine(img, M, (width,height))

cv2.imshow('Imagen Original ', img)
cv2.imshow('Imagen Shear',imgShear)

filename = 'ShearPug.jpg'
cv2.imwrite(filename, imgShear)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

1.2. Implemente su propia versión de la función cv2.warpAffine(...), realice una comparación con la versión de OpenCV.

- **Traslación de la imagen**

  - **Imagen Original y Resultado de la imagen**:



(a) Imagen 1 (b) Imagen 2

Figura 5: Traslación

- **Código**:

```
import cv2
import numpy as np

# Read the image
img = cv2.imread('pug.jpg',1)
height_ = 600
width_ = 600
img = cv2.resize(img,(width_,height_))
height, width = img.shape[:2]
cv2.imshow('Imagen Original', img)

A = np.array([[1,0],
              [0,1]], dtype='int32')
B = np.array([-50,-50], dtype='int32')



newImg = np.array(np.zeros(height_*width_*3).reshape(img.shape));
newImg = np.array(newImg, dtype=np.uint8)

for i in range(height):
    for j in range(width):
        position = np.array([i,j], dtype='int32')
        new_position = np.dot(A,position) + B # [new_i,new_j]

        if ((new_position[0] >= 0 and new_position[0] < height) and (
            new_position[1] >= 0 and new_position[1] < width)):
             newImg[new_position[0],new_position[1]] = img[i,j]

cv2.imshow('Imagen Trasladada', newImg)


filename = 'TraslacionPug2.jpg'
cv2.imwrite(filename, newImg)


cv2.waitKey(0)
cv2.destroyAllWindows()
```

- **Escalación de la imagen**

  - **Imagen Original y Resultado de la imagen sin implementar el metodo solve()**:

(a) Imagen 1 (b) Imagen 2

Figura 6: Escalación

(a) Imagen despues de implementar el metodo solve() para corregir el error

Figura 7: Escalación

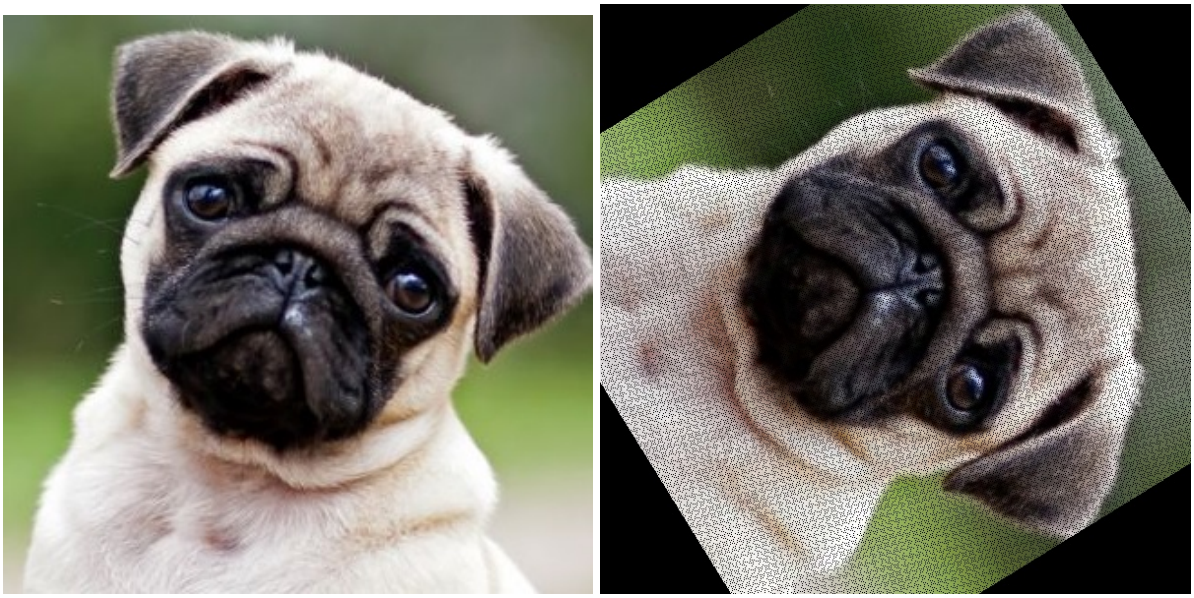
- **Código**:

```
        import cv2
import numpy as np
import math
# Read the image
img = cv2.imread('pugnew.jpg',1)
```

```
height_ = 600
width_ = 600
img = cv2.resize(img,(width_,height_))
height, width = img.shape[:2]
cv2.imshow('Imagen Original', img)

A = np.array([[2.0,0.0],
              [0.0,2.0]])
B = np.array([0.0,0.0])

# Imagen escalada sin solve
newImg = np.array(np.zeros((height_*2)*(width_*2)*3).reshape((img.
    shape[0]*2,img.shape[1]*2,img.shape[2])));
newImg = np.array(newImg, dtype=np.uint8)

for i in range(height):
    for j in range(width):
        position = np.array([i,j])
        new_position = np.dot(A,position) + B # [new_i,new_j]
        new_position = new_position.astype(int)
        if ((new_position[0] >= 0 and new_position[0] < height*2) and
             (new_position[1] >= 0 and new_position[1] < width*2)):
            newImg[new_position[0],new_position[1]] = img[i,j]
cv2.imshow('Imagen Escalada ', newImg)
filename = 'EscalacionPugERROR.jpg'
cv2.imwrite(filename, newImg)

# Imagen escalada con solve
newImg2 = np.array(np.zeros((height_)*(width_)*3).reshape((img.shape)
    ));
newImg2 = np.array(newImg2, dtype=np.uint8)

X = np.array([0,0])
for u in range(width):
    for v in range(height):
        Y = np.array([u,v]) - B
        solve_= cv2.solve(A, Y, X)
        X = solve_[1]
        X = X.astype(int)
        x = X[0]
        y = X[1]
        if ((x >= 0 and x < width) and (y>= 0 and y < height)):
            newImg2[v, u] = img[y, x]


cv2.imshow('Imagen Escalada con solve', newImg2)

filename = 'EscalacionPug2.jpg'
cv2.imwrite(filename, newImg2)


cv2.waitKey(0)
cv2.destroyAllWindows()
```

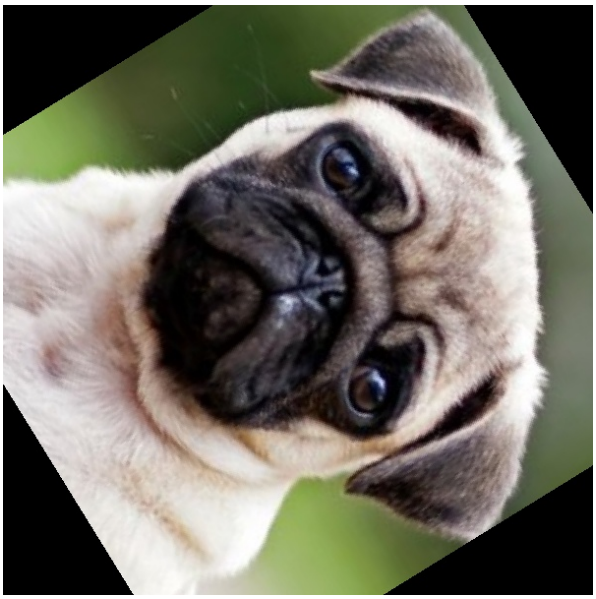
- **Rotación de la imagen**

- **Imagen Original y Resultado de la imagen sin implementar el metodo solve()**:



(a) Imagen 1 (b) Imagen 2

Figura 8: Escalación



(a) Imagen despues de implementar el metodo solve() para corregir el error

Figura 9: Rotación

- **Código**:

```
import cv2
import numpy as np
import math

# Read the image
img = cv2.imread('pugnew.jpg',1)
height_ = 600
width_ = 600
img = cv2.resize(img,(width_,height_))
height, width = img.shape[:2]
cv2.imshow('Imagen Original', img)

angulo = 45
px= height/2
py= width/2
A = np.array([[math.cos(angulo),math.sin(angulo)],
              [-math.sin(angulo),math.cos(angulo)]])
B = np.array([((1-math.cos(angulo)) * px) - (math.sin(angulo) * py),(
    math.sin(angulo) * px) + ((1-math.cos(angulo)) * py)])
print(A)
print(B)



newImg = np.array(np.zeros(height_*width_*3).reshape(img.shape));
newImg = np.array(newImg, dtype=np.uint8)
newImg2 = newImg.copy()
#Imagen rotada sin solve
for i in range(width):
    for j in range(height):
        position = np.array([i,j])
        new_position = np.dot(A,position) + B # [new_i,new_j]
        new_position = new_position.astype(int)
        x = new_position[0]
        y = new_position[1]
        if (( x>= 0 and x < height) and (y >= 0 and y < width)):
            newImg[new_position[0],new_position[1]] = img[i,j]
cv2.imshow('Imagen Rotada ', newImg)
filename = 'RotacionsPugERROR.jpg'
cv2.imwrite(filename, newImg)

#Imagen rotada con solve
X = np.array([0,0])
for u in range(width):
    for v in range(height):
        Y = np.array([u,v]) - B
        solve_= cv2.solve(A, Y, X)
        X = solve_[1]
        X = X.astype(int)
        x = X[0]
        y = X[1]
        if ((x >= 0 and x < width) and (y>= 0 and y < height)):
            # print(x)
```

```
                # print(y)
                # newImg2[v, u] = img[y, x]
                newImg2[u, v] = img[x, y]

cv2.imshow('Imagen Rotada con solve', newImg2)

filename = 'RotacionPug2.jpg'
cv2.imwrite(filename, newImg2)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

- **Shear de la imagen**

  - **Imagen Original y Resultado de la imagen**:

(a) Imagen 1 (b) Imagen 2

Figura 10: Shear

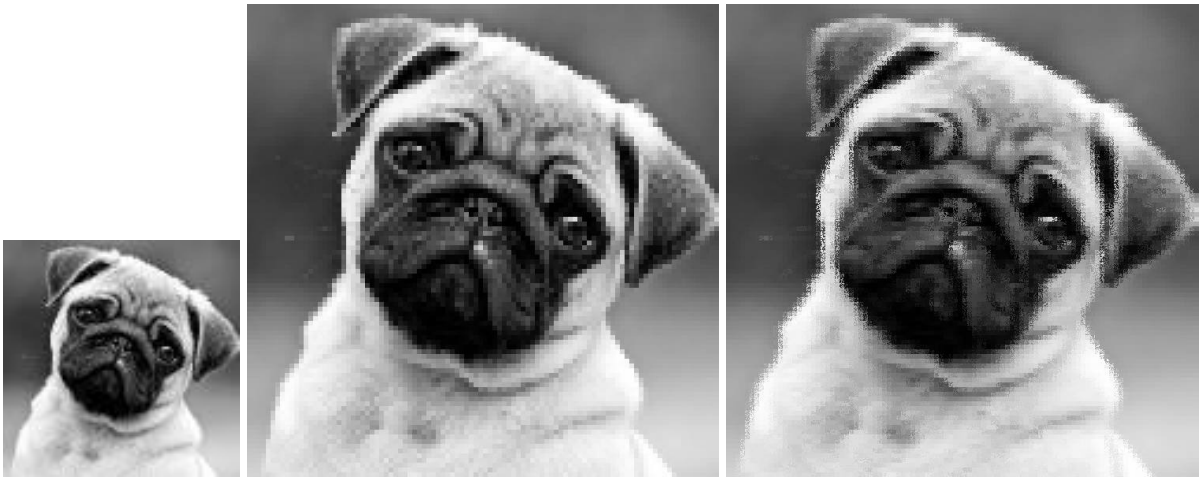
- **Código**:

```
import cv2
import numpy as np
import math

# Read the image
img = cv2.imread('pugnew.jpg',1)
height_ = 600
width_ = 600
img = cv2.resize(img,(width_,height_))
height, width = img.shape[:2]
cv2.imshow('Imagen Original', img)

shx = 0.09
shy = 0.09
A = np.array([[1,shy],
              [shx,1]])
B = np.array([0,0])

newImg = np.array(np.zeros(height_*width_*3).reshape(img.shape));
newImg = np.array(newImg, dtype=np.uint8)

for i in range(height):
    for j in range(width):
        position = np.array([i,j])
        new_position = np.dot(A,position) + B # [new_i,new_j]
        new_position = new_position.astype(int)

        if ((new_position[0] >= 0 and new_position[0] < height) and (
            new_position[1] >= 0 and new_position[1] < width)):
             newImg[new_position[0],new_position[1]] = img[i,j]

cv2.imshow('Imagen Shear', newImg)

filename = 'ShearPug2.jpg'
cv2.imwrite(filename, newImg)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

1.3. Implemente la interpolación de píxeles para aumentar el tamaño de una imagen. Además realice comparaciones con el método de Pixel replication.

- **Solución a la interpolación y replicacion de píxeles para aumentar el tamaño de una imagen y comparación con Pixel replication**

  - **Imagen Original y Resultado de la imagen**:



(a) Pug Original (b) Resultado de la interpolación (c) Resultado de la replicación

Figura 11: Comparación de Interpolación vs Replicación

- **Código**:

```
    import cv2
import numpy as np
import random

img=cv2.imread("pugnew.jpg",0)

height = 100
width = 100
imgReal1 = cv2.resize(img,(width,height))


height,width =imgReal1.shape
# height,width,channels =imgReal1.shape
newTam=6
# emptyImage=np.zeros((width*newTam,height*newTam,channels),np.uint8)

emptyImageReplication=np.zeros((width*newTam,height*newTam),np.uint8)

cv2.imshow("Imagen Original",img)

# ESCALACION CON REPLICATION
i_run = 0
j_run = 0

for i in range(width):
    for j in range(height):
        for x in range(newTam):
            for y in range(newTam):
                if(j == 0 or j == height-1):
                    emptyImageReplication[i_run+x,j_run+y] = imgReal1
                        [i,j]
                else:
```

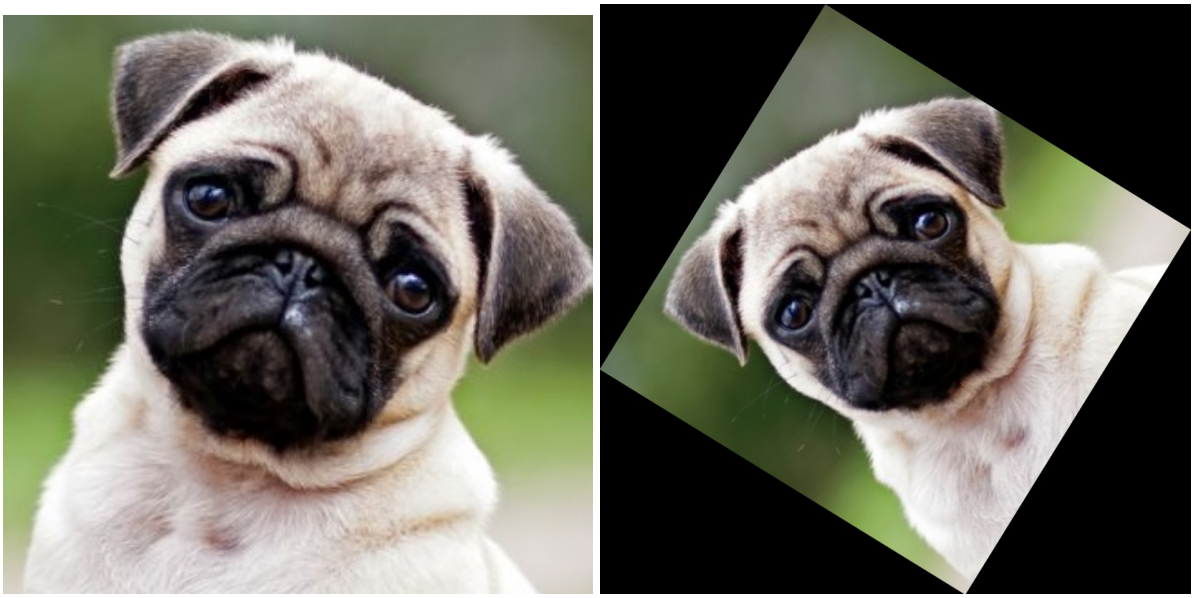
```
                    emptyImageReplication[i_run+x,j_run+y] = random.
                        choice([imgReal1[i,j],imgReal1[i,j+1]])
            j_run += newTam
        j_run = 0
        i_run += newTam

cv2.imshow("Resultado Replicacion",emptyImageReplication)

# ESCALACION CON INTERPOLATION
i_run = 0
j_run = 0

emptyImageInterpolation=np.zeros((width*newTam,height*newTam),np.
    uint8)
for i in range(width):
    for j in range(height):
        for x in range(newTam):
            for y in range(newTam):
                if(j == 0 or j == height-1):
                    emptyImageInterpolation[i_run+x,j_run+y] =
                        imgReal1[i,j]
                else:
                    value = (imgReal1[i,j].astype(np.intc) + imgReal1
                        [i,j].astype(np.intc))/2
                    emptyImageInterpolation[i_run+x,j_run+y] = np.
                        uint8(value)
            j_run += newTam
        j_run = 0
        i_run += newTam
cv2.imshow("Resultado Interpolacion",emptyImageInterpolation)



cv2.waitKey(0)
cv2.destroyAllWindows()
```

2. Algoritmo de computación gráfica Affine transformation (Practica 9.2)).

2.1. Solucione el problema de la rotación de imágenes al utilizar warpAffine.

- **Solución a la rotación de la imagen**

  - **Imagen Original y Resultado de la imagen**:



(a) Imagen 1 (b) Imagen 2

Figura 12: Rótacion

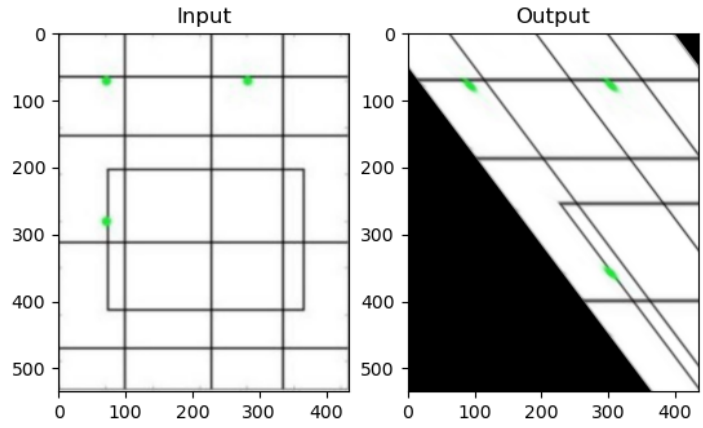
- **Código**:

```
import cv2
import numpy as np
import math

img = cv2.imread('pugnew.jpg',1)
img = cv2.resize(img,(600,600))

height, width = img.shape[:2]

# matriz M
angulo = 45
px= height/2
py= width/2
M = np.float32([[math.cos(angulo),math.sin(angulo),((1-math.cos(
    angulo)) * px) - (math.sin(angulo) * py)],
                [-math.sin(angulo),math.cos(angulo),(math.sin(angulo)
                    * px) + ((1-math.cos(angulo)) * py)]])

cos = np.abs(M[0, 0])
sin = np.abs(M[0, 1])

# calcular los nuevos limites
newWidth = int((height * sin) + (width * cos))
newHheight = int((height * cos) + (width * sin))

# ajustar la matriz de rotacion
M[0, 2] += (newWidth / 2) - px
M[1, 2] += (newHheight / 2) - py

rotated_img = cv2.warpAffine(src=img, M=M, dsize=(nW, nH))
print(len(rotated_img))
print(len(rotated_img[0]))

cv2.imshow('imgen Original ', img)
cv2.imshow('imgen Rotada ', rotated_img)

filename = 'RotadaCorrectPug.jpg'
cv2.imwrite(filename, rotated_img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

2.2. OpenCV utiliza la función cv2.getAffineTransform(pts1,pts2) para obtener la matriz M a partir de dos puntos. Implemente su propia versión de la función getAffineTransform

- **Implmentacion del metodo cv2.getAffineTransform(pts1,pts2)**

  - **Imagen Original y Resultado de la imagen**:



(a) Imagen Original y el resultado

Figura 13: Imagenes despues de aplicar getAffineTransform

- **Código con el metodo cv2.getAffineTransform(pts1,pts2)** :

```
 import cv2
import numpy as np
import math
from matplotlib import pyplot as plt

img = cv2.imread('test.jpg')
rows,cols,ch = img.shape

pts1 = np.float32([[50,50],[200,50],[50,200]])
pts2 = np.float32([[50,50],[200,50],[200,250]])


M = cv2.getAffineTransform(pts1,pts2)
print(M)

dst = cv2.warpAffine(img,M,(cols,rows))


plt.subplot(121),plt.imshow(img),plt.title('Input')
plt.subplot(122),plt.imshow(dst),plt.title('Output')
plt.savefig('ImagenAffineTransform.png')

plt.show()

cv2.waitKey(0)
cv2.destroyAllWindows()
```

- **Implementacion del metodo getAffineTransform(pts1,pts2) usando un solve** :

```
import cv2
import numpy as np
import math
from matplotlib import pyplot as plt

img = cv2.imread('test.jpg')
rows,cols,ch = img.shape


def getAffineTransform(pts1, pts2):
    p = []
    for x,y in pts1:
        p.append((x,y,1))
    return np.linalg.solve(p, pts2).T

pts1 = np.float32([[50,50],[200,50],[50,200]])
pts2 = np.float32([[50,50],[200,50],[200,250]])

M = getAffineTransform(pts1,pts2)
print(M)

dst = cv2.warpAffine(img,M,(cols,rows))

plt.subplot(121),plt.imshow(img),plt.title('Input')
plt.subplot(122),plt.imshow(dst),plt.title('Output')
plt.savefig('ImagenAffineTransform2.png')
plt.show()

cv2.waitKey(0)
cv2.destroyAllWindows()
```

3. Link de los códigos en github

https://github.com/kpzaolod6000/Graphics-Computing/tree/main/parcial_3/lab_10