



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

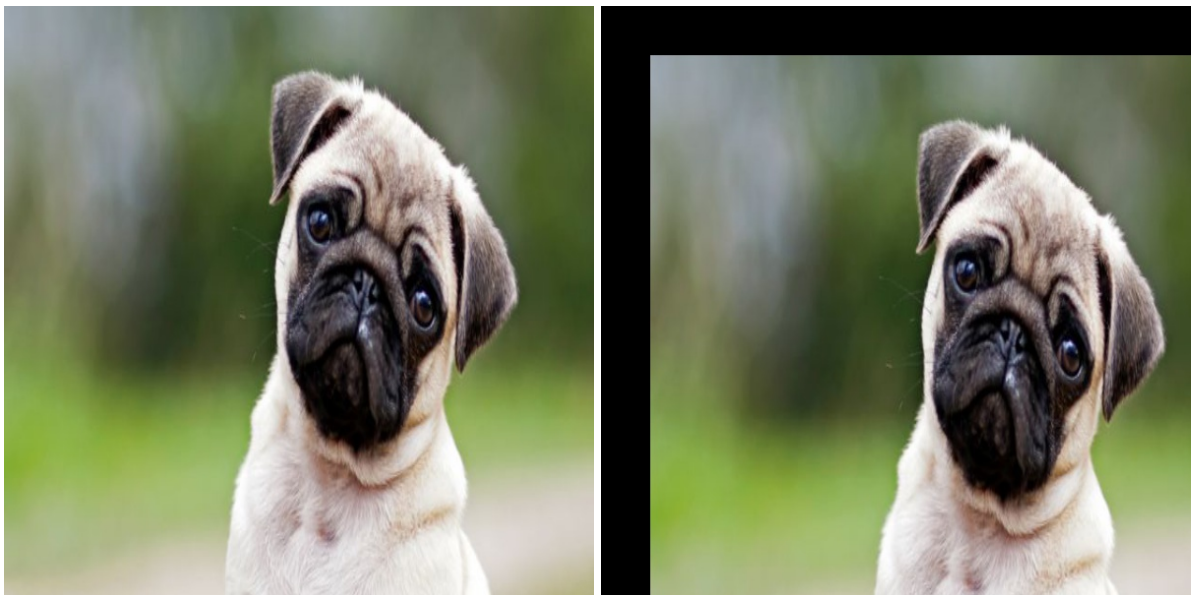
	1
1. Algoritmo de computación gráfica Affine transformation (Practica 9.1)).	2
1.1. Utilice la función <code>cv2.warpAffine(...)</code> de OpenCV para obtener una traslación, escala, rotación y shear. Usted tiene que definir la matriz M y puede utilizar cualquier imagen.	2
1.2. Implemente su propia versión de la función <code>cv2.warpAffine(...)</code> , realice una comparación con la versión de OpenCV.	8
1.3. Implemente la interpolación de píxeles para aumentar el tamaño de una imagen. Además realice comparaciones con el método de Pixel replication. . . .	15
2. Algoritmo de computación gráfica Affine transformation (Practica 9.2)).	18
2.1. Solucione el problema de la rotación de imágenes al utilizar <code>warpAffine</code>	18
2.2. OpenCV utiliza la función <code>cv2.getAffineTransform(pts1,pts2)</code> para obtener la matriz M a partir de dos puntos. Implemente su propia versión de la función <code>getAffineTransform</code>	19
3. Link de los códigos en github	21

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:

```
1 import cv2
2 import numpy as np
3
4 # Read the image
5 img = cv2.imread('pug.jpg',1)
6 img = cv2.resize(img, (600,600))
7 rows, cols = img.shape[:2]
8 cv2.imshow('Imagen Original', img)
9
10 # Matriz M
11
12 M = np.float32([[1,0,50], [0,1,50]])
13 print(M)
14
15 # Apply the affine transformation using cv2.warpAffine()
16 imgResult = cv2.warpAffine(img, M, (cols,rows))
17 a
18 cv2.imshow('Result', imgResult)
19
20 filename = 'TraslacionPug.jpg'
21 cv2.imwrite(filename, imgResult)
22
23 cv2.waitKey(0)
24 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:

```

1  import cv2
2  import numpy as np
3
4  # Read the image
5  img = cv2.imread('pug.jpg',1)
6  img = cv2.resize(img, (600,600))
7  rows, cols = img.shape[:2]
8  cv2.imshow('Imagen Original', img)
9
10 # Matriz M
11 M = np.float32([[2,0,0], [0,2,0]])
12 print(M)
13
14 # # Apply the affine transformation using cv2.warpAffine()
15 imgResult = cv2.warpAffine(img, M, (cols*2,rows*2))
16
17 cv2.imshow('Output', imgResult)
18 filename = 'EscalarPug.jpg'
19 cv2.imwrite(filename, imgResult)
20
21 cv2.waitKey(0)
22 cv2.destroyAllWindows()

```

■ Rotación de la imagen usando la función cv2.warpAffine(...)

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



(a) Imagen 1



(b) Imagen 2

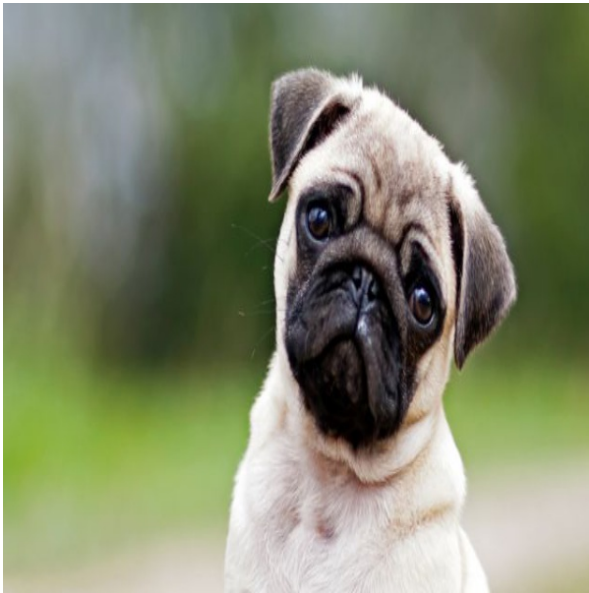
Figura 3: Rotación

■ Código:

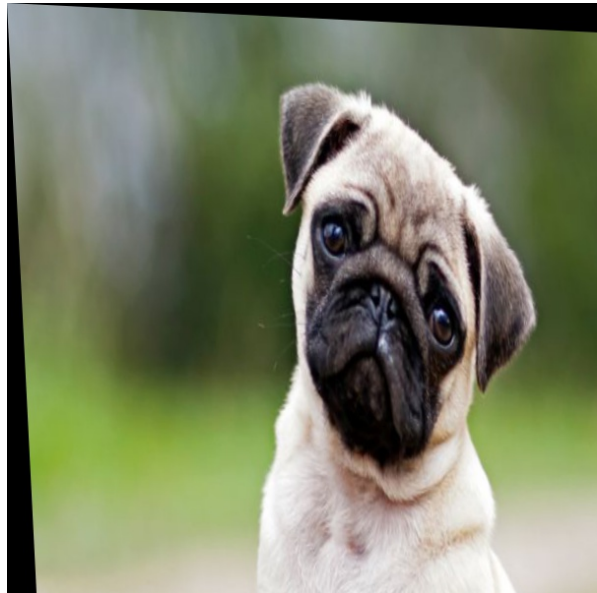
```
1 import cv2
2 import numpy as np
3 import math
4
5 # Reading the img
6 img = cv2.imread('pug.jpg',1)
7 img = cv2.resize(img, (600,600))
8
9 # matriz M
10 angulo = 45
11 px= 300.0
12 py= 300.0
13 M = np.float32([[math.cos(angulo),math.sin(angulo), ((1-math.cos(
14     angulo)) * px) - (math.sin(angulo) * py)],
15     [-math.sin(angulo),math.cos(angulo), (math.sin(angulo)
16     * px) + ((1-math.cos(angulo)) * py)])])
17
18 # print(M)
19 # rotate the img using cv2.warpAffine
20 rotated_img = cv2.warpAffine(src=img, M=M, dsize=(width, height))
21
22 cv2.imshow('imgn Original ', img)
23 cv2.imshow('imgn Rotada ', rotated_img)
24
25 cv2.waitKey(0)
26 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

■ Código:

```

1  import cv2
2  import numpy as np
3  import math
4
5  # Reading the img
6  img = cv2.imread('pug.jpg',1)
7  img = cv2.resize(img, (600,600))
8  # dividing height and width by 2 to get the center of the img
9  height, width = img.shape[:2]
10
11  shx = 0.05
12  shy = 0.05
13  M = np.float32([[1, shy, 0],
14                  [shx, 1, 0]])
15  print(M)
16
17  # # Apply the affine transformation using cv2.warpAffine()
18  imgShear = cv2.warpAffine(img, M, (width,height))
19
20  cv2.imshow('Imagen Original ', img)
21  cv2.imshow('Imagen Shear',imgShear)
22
23  filename = 'ShearPug.jpg'
24  cv2.imwrite(filename, imgShear)
25
26  cv2.waitKey(0)
27  cv2.destroyAllWindows()

```

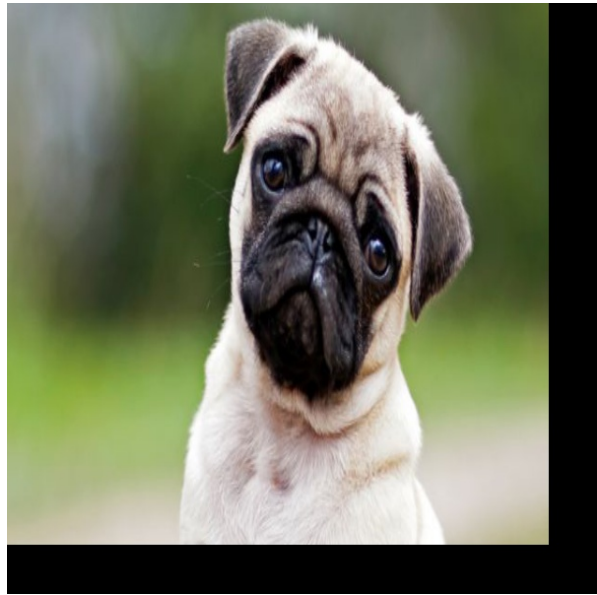

1.2. Implemente su propia versión de la función `cv2.warpAffine(...)`, realice una comparación con la versión de OpenCV.

■ **Traslación de la imagen**

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



(a) Imagen 1



(b) Imagen 2

Figura 5: Traslación

■ Código:

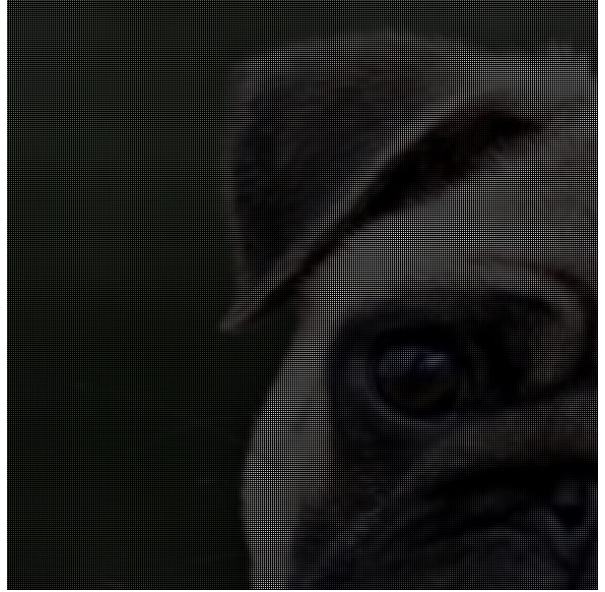
```
1  import cv2
2  import numpy as np
3
4  # Read the image
5  img = cv2.imread('pug.jpg',1)
6  height_ = 600
7  width_ = 600
8  img = cv2.resize(img, (width_,height_))
9  height, width = img.shape[:2]
10 cv2.imshow('Imagen Original', img)
11
12 A = np.array([[1,0],
13               [0,1]], dtype='int32')
14 B = np.array([-50,-50], dtype='int32')
15
16
17
18 newImg = np.array(np.zeros(height_*width_*3).reshape(img.shape));
19 newImg = np.array(newImg, dtype=np.uint8)
20
21 for i in range(height):
22     for j in range(width):
23         position = np.array([i,j], dtype='int32')
24         new_position = np.dot(A,position) + B # [new_i,new_j]
25
26         if ((new_position[0] >= 0 and new_position[0] < height) and (
27             new_position[1] >= 0 and new_position[1] < width)):
28             newImg[new_position[0],new_position[1]] = img[i,j]
29
30 cv2.imshow('Imagen Traslada', newImg)
31
32 filename = 'TraslacionPug2.jpg'
33 cv2.imwrite(filename, newImg)
34
35
36 cv2.waitKey(0)
37 cv2.destroyAllWindows()
```

■ Escalación de la imagen

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

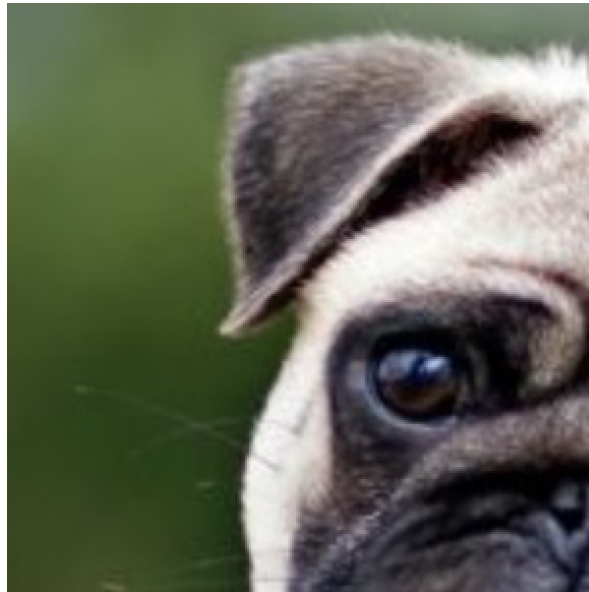


(a) Imagen 1



(b) Imagen 2

Figura 6: Escalación



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

Figura 7: Escalación

■ Código:

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

```

6 height_ = 600
7 width_ = 600
8 img = cv2.resize(img, (width_, height_))
9 height, width = img.shape[:2]
10 cv2.imshow('Imagen Original', img)
11
12 A = np.array([[2.0, 0.0],
13              [0.0, 2.0]])
14 B = np.array([0.0, 0.0])
15
16 # Imagen escalada sin solve
17 newImg = np.array(np.zeros((height_*2)*(width_*2)*3).reshape((img.
18     shape[0]*2, img.shape[1]*2, img.shape[2])));
19 newImg = np.array(newImg, dtype=np.uint8)
20
21 for i in range(height):
22     for j in range(width):
23         position = np.array([i, j])
24         new_position = np.dot(A, position) + B # [new_i, new_j]
25         new_position = new_position.astype(int)
26         if ((new_position[0] >= 0 and new_position[0] < height*2) and
27             (new_position[1] >= 0 and new_position[1] < width*2)):
28             newImg[new_position[0], new_position[1]] = img[i, j]
29 cv2.imshow('Imagen Escalada ', newImg)
30 filename = 'EscalacionPugERROR.jpg'
31 cv2.imwrite(filename, newImg)
32
33 # Imagen escalada con solve
34 newImg2 = np.array(np.zeros((height_)*(width_)*3).reshape((img.shape)
35     ));
36 newImg2 = np.array(newImg2, dtype=np.uint8)
37
38 X = np.array([0, 0])
39 for u in range(width):
40     for v in range(height):
41         Y = np.array([u, v]) - B
42         solve_ = cv2.solve(A, Y, X)
43         X = solve_[1]
44         X = X.astype(int)
45         x = X[0]
46         y = X[1]
47         if ((x >= 0 and x < width) and (y >= 0 and y < height)):
48             newImg2[v, u] = img[y, x]
49
50 cv2.imshow('Imagen Escalada con solve', newImg2)
51
52 filename = 'EscalacionPug2.jpg'
53 cv2.imwrite(filename, newImg2)
54
55 cv2.waitKey(0)
56 cv2.destroyAllWindows()

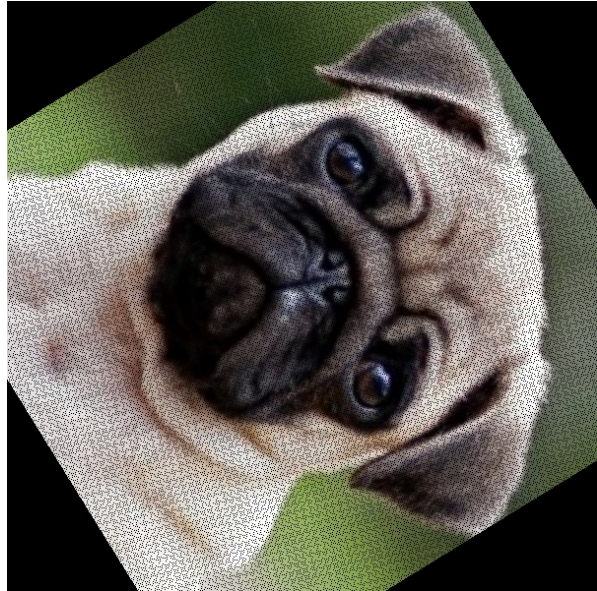
```

■ Rotación de la imagen

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

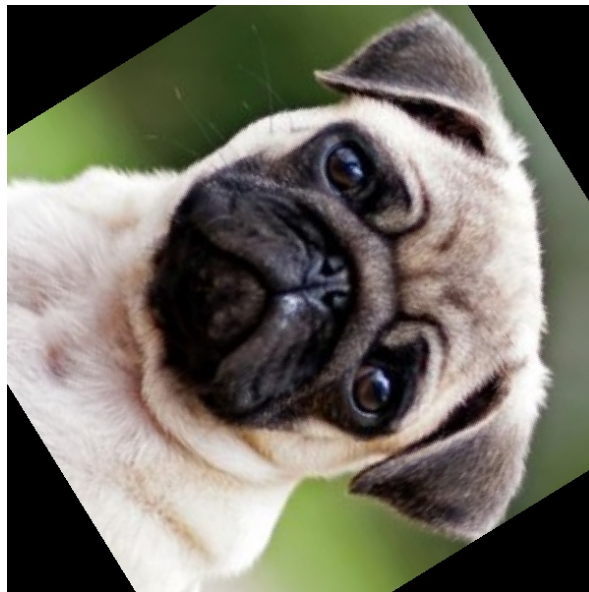


(a) Imagen 1



(b) Imagen 2

Figura 8: Escalación



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

Figura 9: Rotación

■ Código:

```
1 import cv2
2 import numpy as np
3 import math
4
5 # Read the image
6 img = cv2.imread('pugnew.jpg',1)
7 height_ = 600
8 width_ = 600
9 img = cv2.resize(img, (width_,height_))
10 height, width = img.shape[:2]
11 cv2.imshow('Imagen Original', img)
12
13 angulo = 45
14 px= height/2
15 py= width/2
16 A = np.array([[math.cos(angulo),math.sin(angulo)],
17               [-math.sin(angulo),math.cos(angulo)]])
18 B = np.array([(1-math.cos(angulo)) * px - (math.sin(angulo) * py), (
19               math.sin(angulo) * px) + ((1-math.cos(angulo)) * py)])
20 print(A)
21 print(B)
22
23
24 newImg = np.array(np.zeros(height_*width_*3).reshape(img.shape));
25 newImg = np.array(newImg, dtype=np.uint8)
26 newImg2 = newImg.copy()
27 #Imagen rotada sin solve
28 for i in range(width):
29     for j in range(height):
30         position = np.array([i,j])
31         new_position = np.dot(A,position) + B # [new_i,new_j]
32         new_position = new_position.astype(int)
33         x = new_position[0]
34         y = new_position[1]
35         if ((x >= 0 and x < height) and (y >= 0 and y < width)):
36             newImg[new_position[0],new_position[1]] = img[i,j]
37 cv2.imshow('Imagen Rotada ', newImg)
38 filename = 'RotacionsPugERROR.jpg'
39 cv2.imwrite(filename, newImg)
40
41 #Imagen rotada con solve
42 X = np.array([0,0])
43 for u in range(width):
44     for v in range(height):
45         Y = np.array([u,v]) - B
46         solve_ = cv2.solve(A, Y, X)
47         X = solve_[1]
48         X = X.astype(int)
49         x = X[0]
50         y = X[1]
51         if ((x >= 0 and x < width) and (y >= 0 and y < height)):
52             # print(x)
```

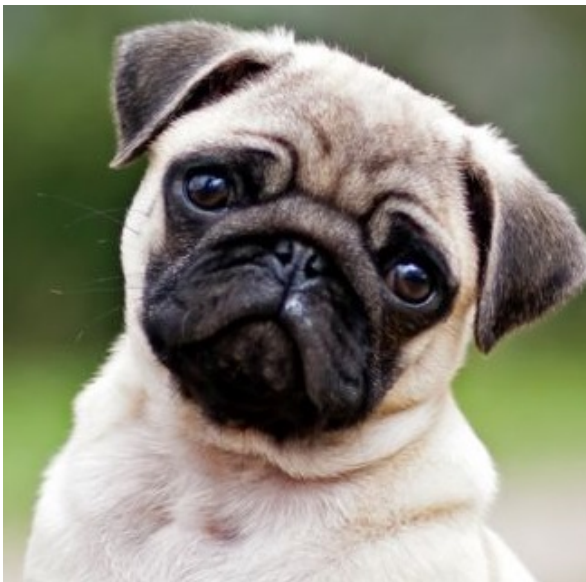
```

53         # print(y)
54         # newImg2[v, u] = img[y, x]
55         newImg2[u, v] = img[x, y]
56
57     cv2.imshow('Imagen Rotada con solve', newImg2)
58
59     filename = 'RotacionPug2.jpg'
60     cv2.imwrite(filename, newImg2)
61
62     cv2.waitKey(0)
63     cv2.destroyAllWindows()

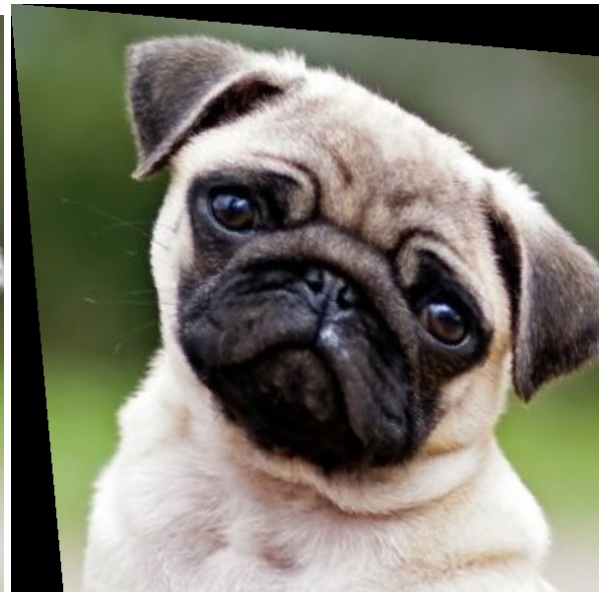
```

■ Shear de la imagen

- Imagen Original y Resultado de la imagen:



(a) Imagen 1



(b) Imagen 2

Figura 10: Shear

■ Código:

```
1 import cv2
2 import numpy as np
3 import math
4
5 # Read the image
6 img = cv2.imread('pugnew.jpg',1)
7 height_ = 600
8 width_ = 600
9 img = cv2.resize(img, (width_,height_))
10 height, width = img.shape[:2]
11 cv2.imshow('Imagen Original', img)
12
13 shx = 0.09
14 shy = 0.09
15 A = np.array([[1, shy],
16               [shx, 1]])
17 B = np.array([0,0])
18
19 newImg = np.array(np.zeros(height_*width_*3).reshape(img.shape));
20 newImg = np.array(newImg, dtype=np.uint8)
21
22 for i in range(height):
23     for j in range(width):
24         position = np.array([i,j])
25         new_position = np.dot(A,position) + B # [new_i,new_j]
26         new_position = new_position.astype(int)
27
28         if ((new_position[0] >= 0 and new_position[0] < height) and (
29             new_position[1] >= 0 and new_position[1] < width)):
30             newImg[new_position[0],new_position[1]] = img[i,j]
31
32 cv2.imshow('Imagen Shear', newImg)
33
34 filename = 'ShearPug2.jpg'
35 cv2.imwrite(filename, newImg)
36
37 cv2.waitKey(0)
38 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 replicación de píxeles para aumentar el tamaño de una imagen y comparación con Pixel replication

- Imagen Original y Resultado de la imagen:



Figura 11: Comparación de Interpolación vs Replicación

■ Código:

```

1     import cv2
2     import numpy as np
3     import random
4
5     img=cv2.imread("pugnew.jpg",0)
6
7     height = 100
8     width = 100
9     imgReall = cv2.resize(img, (width,height))
10
11
12     height,width =imgReall.shape
13     # height,width,channels =imgReall.shape
14     newTam=6
15     # emptyImage=np.zeros((width*newTam,height*newTam,channels),np.uint8)
16
17     emptyImageReplication=np.zeros((width*newTam,height*newTam),np.uint8)
18
19     cv2.imshow("Imagen Original",img)
20
21     # ESCALACION CON REPLICATION
22     i_run = 0
23     j_run = 0
24
25     for i in range(width):
26         for j in range(height):
27             for x in range(newTam):
28                 for y in range(newTam):
29                     if(j == 0 or j == height-1):
30                         emptyImageReplication[i_run+x,j_run+y] = imgReall
31                                     [i,j]
32                     else:

```

```

32         emptyImageReplication[i_run+x,j_run+y] = random.
           choice([imgReall[i,j],imgReall[i,j+1]])
33     j_run += newTam
34     j_run = 0
35     i_run += newTam
36
37 cv2.imshow("Resultado Replicacion",emptyImageReplication)
38
39 # ESCALACION CON INTERPOLATION
40 i_run = 0
41 j_run = 0
42
43 emptyImageInterpolation=np.zeros((width*newTam,height*newTam),np.
   uint8)
44 for i in range(width):
45     for j in range(height):
46         for x in range(newTam):
47             for y in range(newTam):
48                 if(j == 0 or j == height-1):
49                     emptyImageInterpolation[i_run+x,j_run+y] =
                       imgReall[i,j]
50                 else:
51                     value = (imgReall[i,j].astype(np.intc) + imgReall
                       [i,j].astype(np.intc))/2
52                     emptyImageInterpolation[i_run+x,j_run+y] = np.
                       uint8(value)
53             j_run += newTam
54         j_run = 0
55         i_run += newTam
56 cv2.imshow("Resultado Interpolacion",emptyImageInterpolation)
57
58
59
60 cv2.waitKey(0)
61 cv2.destroyAllWindows()

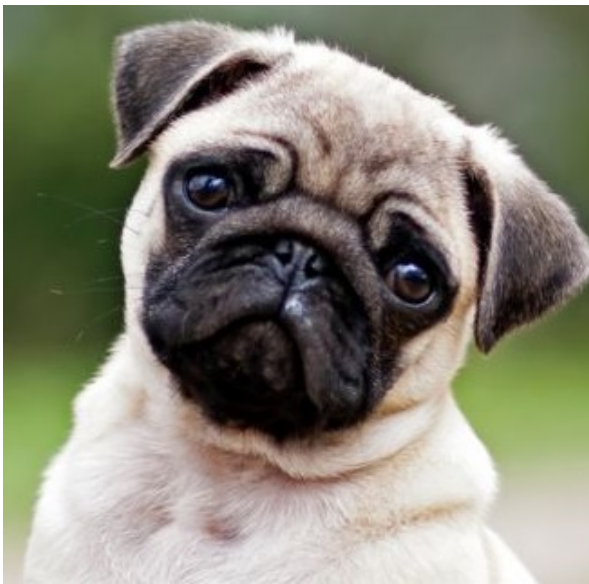
```

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

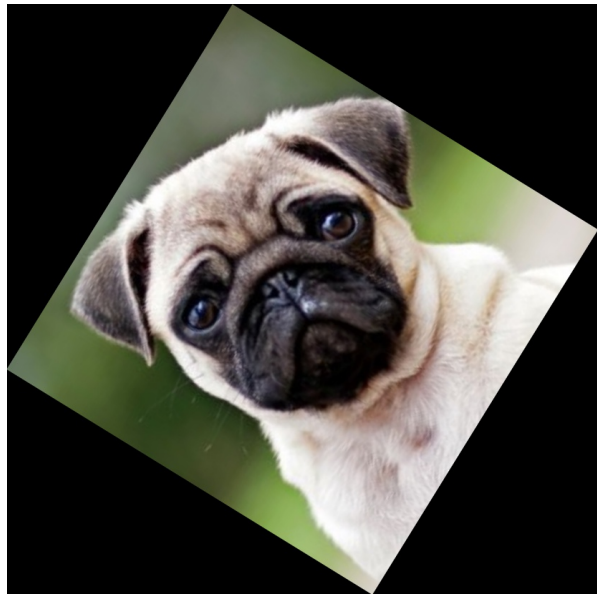
2.1. Solucione el problema de la rotación de imágenes al utilizar warpAffine.

- Solución a la rotación de la imagen

- Imagen Original y Resultado de la imagen:



(a) Imagen 1



(b) Imagen 2

Figura 12: Rótacion

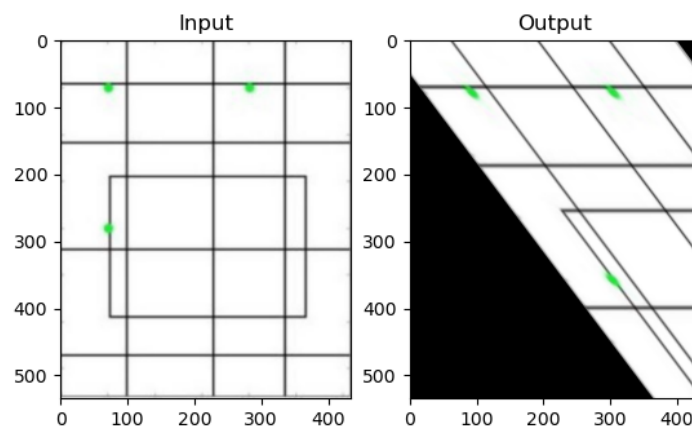
■ Código:

```
1  import cv2
2  import numpy as np
3  import math
4
5  img = cv2.imread('pugnew.jpg',1)
6  img = cv2.resize(img, (600,600))
7
8  height, width = img.shape[:2]
9
10 # matriz M
11 angulo = 45
12 px= height/2
13 py= width/2
14 M = np.float32([[math.cos(angulo),math.sin(angulo), ((1-math.cos(
15     angulo)) * px) - (math.sin(angulo) * py)],
16     [-math.sin(angulo),math.cos(angulo), (math.sin(angulo)
17     * px) + ((1-math.cos(angulo)) * py) ]])
18
19 cos = np.abs(M[0, 0])
20 sin = np.abs(M[0, 1])
21
22 # calcular los nuevos limites
23 newWidth = int((height * sin) + (width * cos))
24 newHeight = int((height * cos) + (width * sin))
25
26 # ajustar la matriz de rotacion
27 M[0, 2] += (newWidth / 2) - px
28 M[1, 2] += (newHeight / 2) - py
29
30 rotated_img = cv2.warpAffine(src=img, M=M, dsize=(nW, nH))
31 print(len(rotated_img))
32 print(len(rotated_img[0]))
33
34 cv2.imshow('imagen Original ', img)
35 cv2.imshow('imagen Rotada ', rotated_img)
36
37 filename = 'RotadaCorrectPug.jpg'
38 cv2.imwrite(filename, rotated_img)
39
40 cv2.waitKey(0)
41 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)` :

```

1  import cv2
2  import numpy as np
3  import math
4  from matplotlib import pyplot as plt
5
6  img = cv2.imread('test.jpg')
7  rows,cols,ch = img.shape
8
9  pts1 = np.float32([[50,50],[200,50],[50,200]])
10 pts2 = np.float32([[50,50],[200,50],[200,250]])
11
12
13 M = cv2.getAffineTransform(pts1,pts2)
14 print(M)
15
16 dst = cv2.warpAffine(img,M,(cols,rows))
17
18
19 plt.subplot(121),plt.imshow(img),plt.title('Input')
20 plt.subplot(122),plt.imshow(dst),plt.title('Output')
21 plt.savefig('ImagenAffineTransform.png')
22
23 plt.show()
24
25 cv2.waitKey(0)
26 cv2.destroyAllWindows()
```

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

```
1  import cv2
2  import numpy as np
3  import math
4  from matplotlib import pyplot as plt
5
6  img = cv2.imread('test.jpg')
7  rows,cols,ch = img.shape
8
9
10 def getAffineTransform(pts1, pts2):
11     p = []
12     for x,y in pts1:
13         p.append((x,y,1))
14     return np.linalg.solve(p, pts2).T
15
16 pts1 = np.float32([[50,50],[200,50],[50,200]])
17 pts2 = np.float32([[50,50],[200,50],[200,250]])
18
19 M = getAffineTransform(pts1,pts2)
20 print(M)
21
22 dst = cv2.warpAffine(img,M,(cols,rows))
23
24 plt.subplot(121),plt.imshow(img),plt.title('Input')
25 plt.subplot(122),plt.imshow(dst),plt.title('Output')
26 plt.savefig('ImagenAffineTransform2.png')
27 plt.show()
28
29 cv2.waitKey(0)
30 cv2.destroyAllWindows()
```

3. Link de los códigos en github

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