César Soltero Pérez

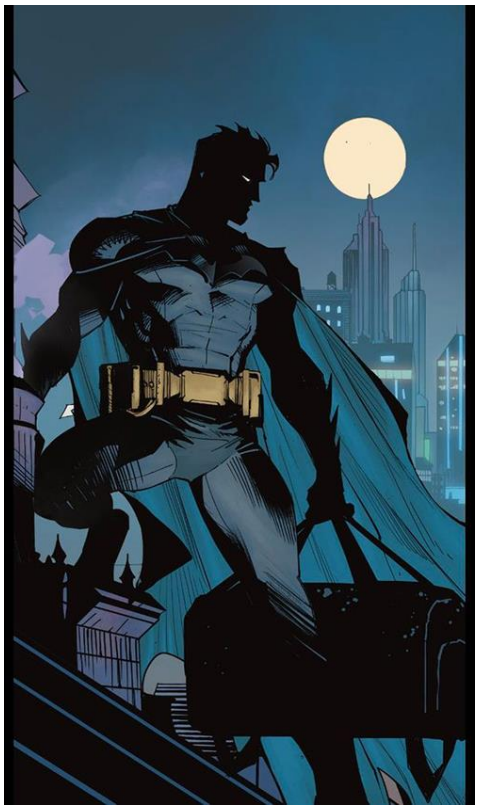
18310460

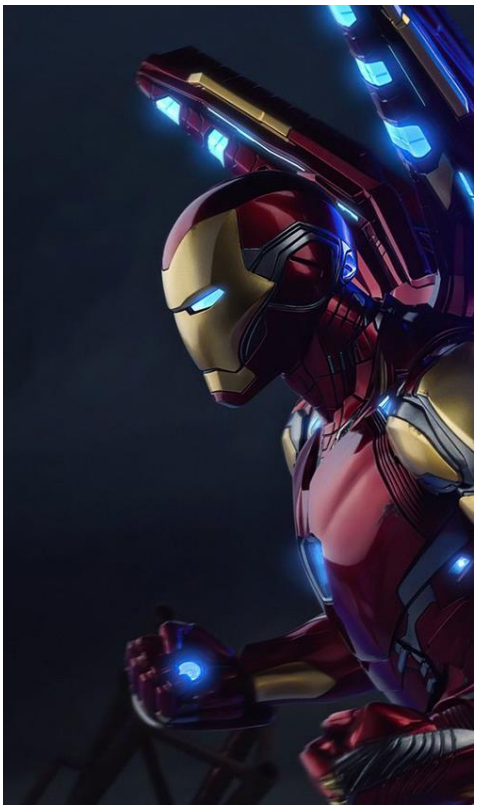
7E1

Sistemas de procesamiento de imágenes y visión artificial

Práctica 2

Con base en las siguientes imágenes:



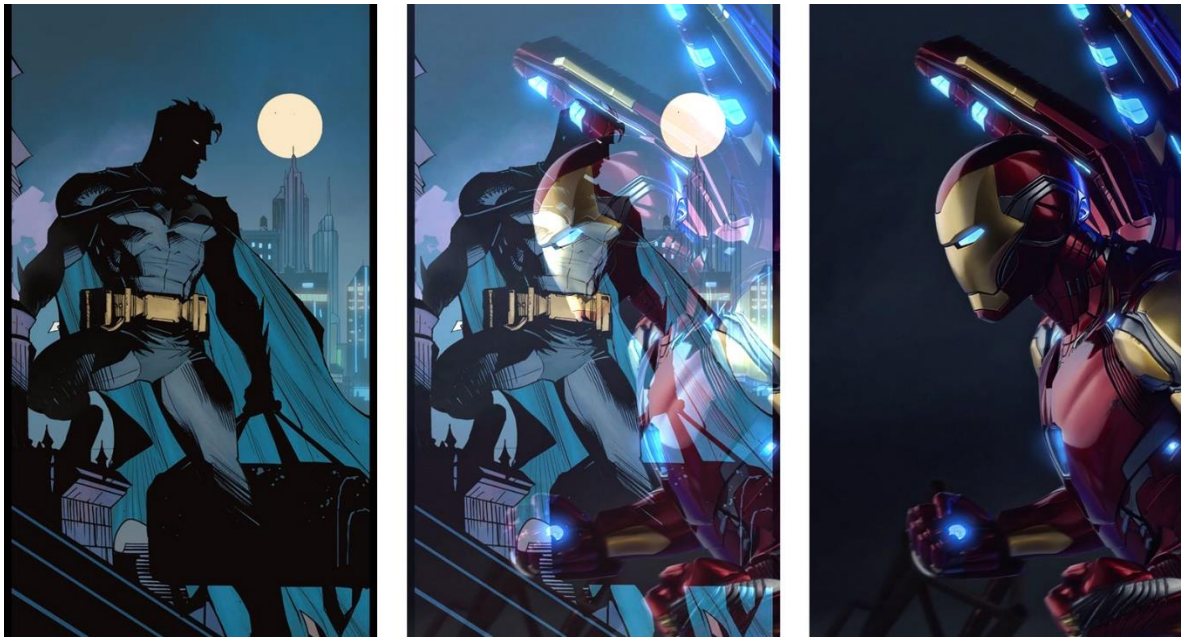


Se desarrollarán, por medio de la librería OpenCV, las operaciones de:

- Suma
- Resta
- Multiplicación
- División
- Logaritmo natural
- Raíz cuadrada
- Derivada
- Potencia
- Conjunción
- Disyunción
- Negación
- Escalado
- Rotación
- Translación a fin
- Transpuesta

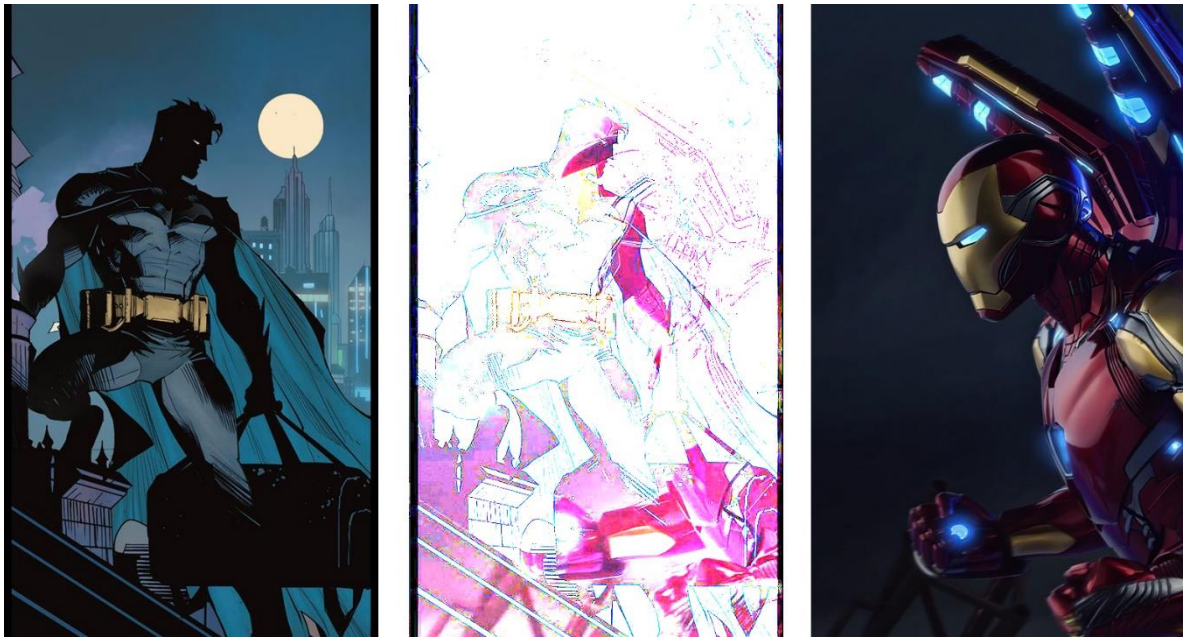
Demostración

- **Suma**

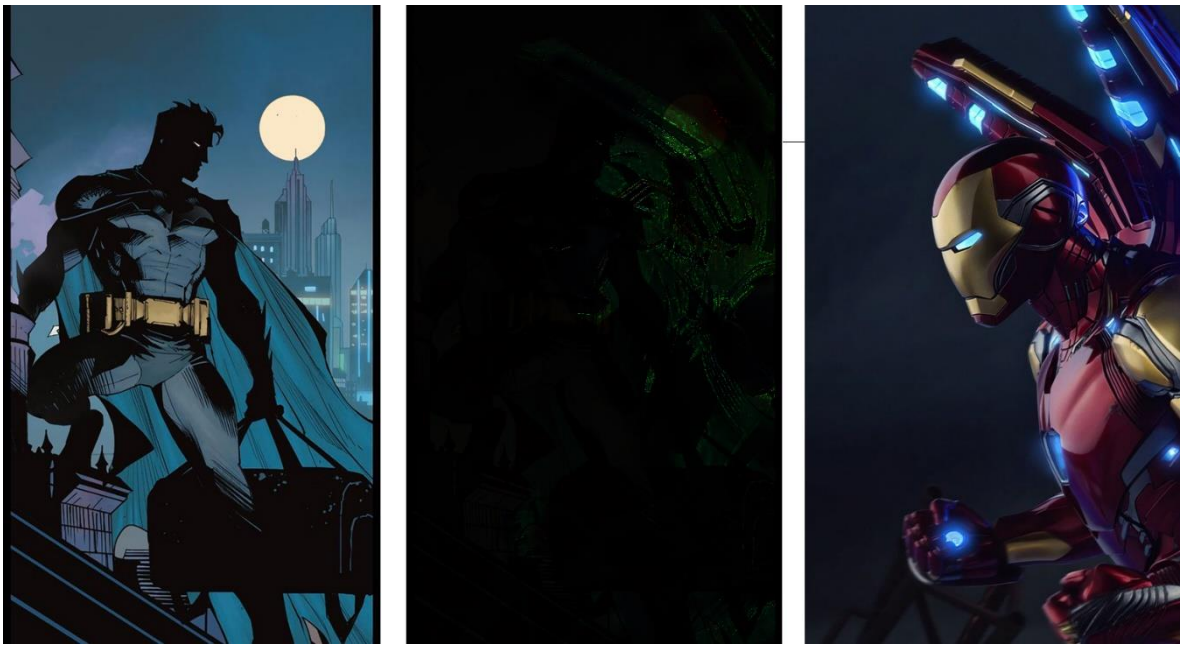


- **Resta**

- **Multiplicación**

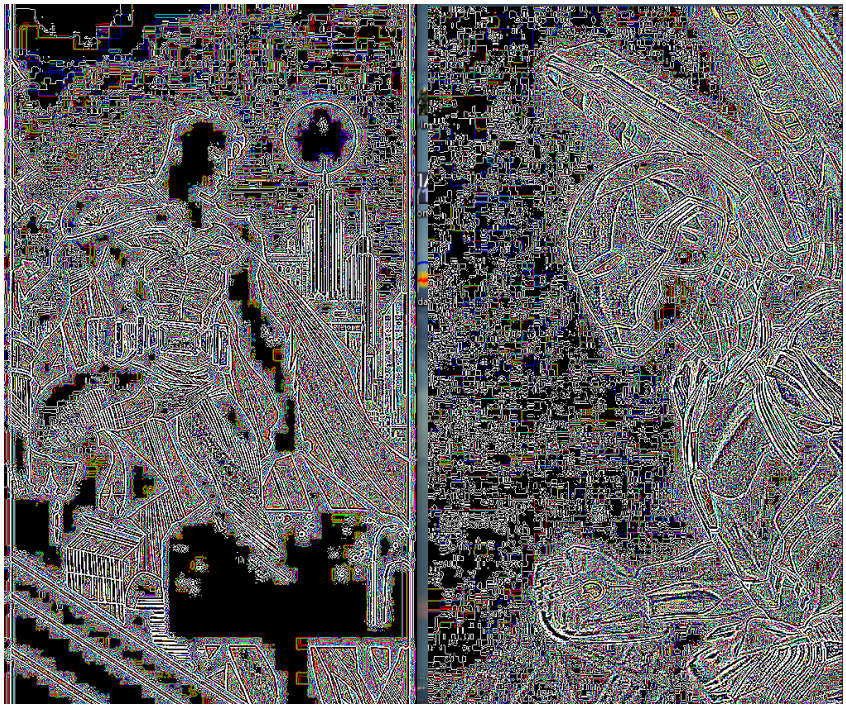


- **División**



- **Logaritmo natural**

- **Derivada**



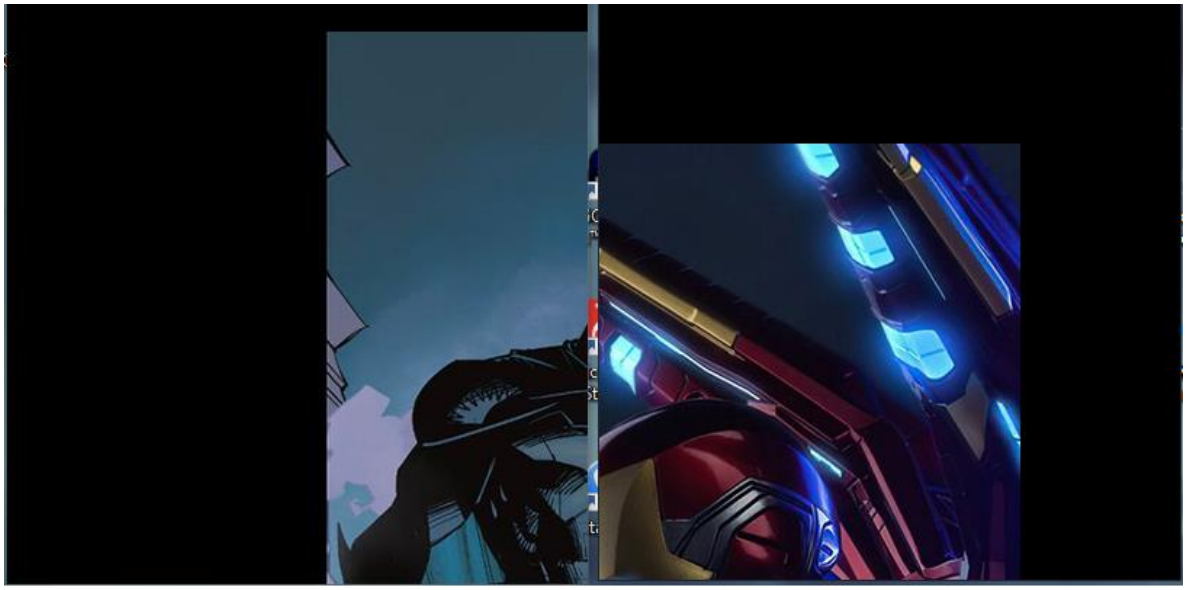
- **Potencia**

- **Conjunción**

- **Disyunción**

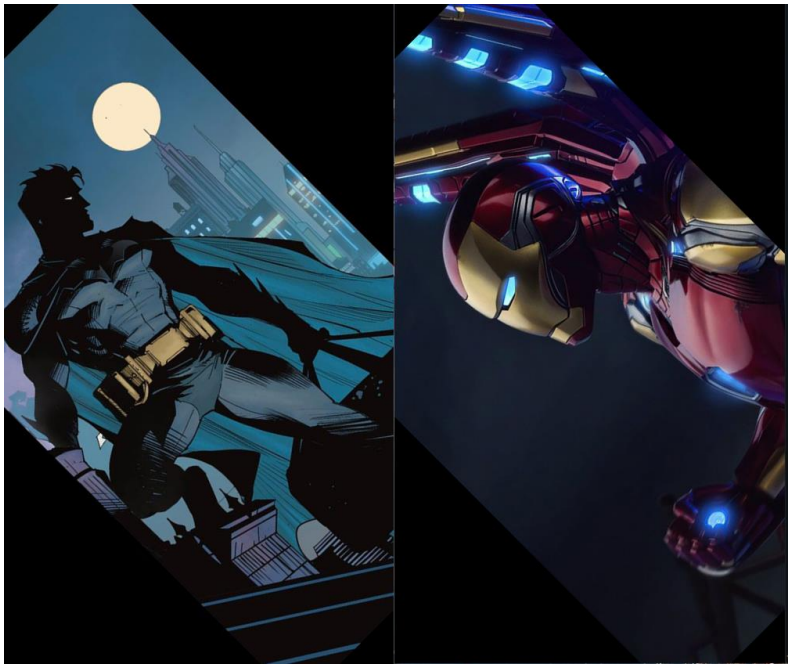
- **Negación**

- **Traslación**



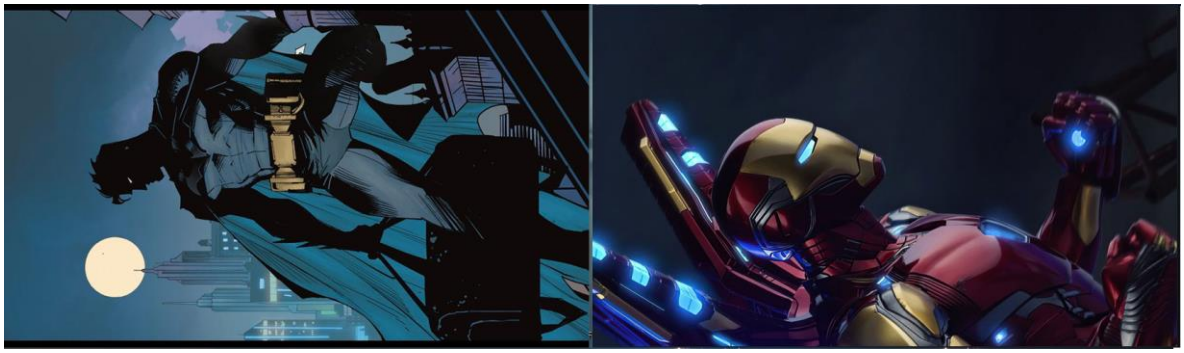
- **Escalado**

- **Rotación**



- **Translación a fin**

- **Transpuesta**



Código:

```
#Librerías
import numpy as np
from matplotlib import pyplot as plt
import cv2

#Cargar imagenes (Resolucion de 500 x 850)
imagen1 = cv2.imread('bat1.jpg',1)
imagen2 = cv2.imread('im1.jpg',1)

#Mostrar imagenes
cv2.imshow("Batman",imagen1)
cv2.moveWindow("Batman",0,200)
cv2.imshow("Iron Man",imagen2)
cv2.moveWindow("Iron Man",1985,200)

#Guardar tecla pulsada
resultado = cv2.waitKey(0)

#Suma
if resultado == 99:
    suma = cv2.add(imagen1,imagen2)
    cv2.imshow("Suma",suma)
    cv2.moveWindow("Suma",900,200)

#Resta
resultado = cv2.waitKey(0)
if resultado == 99:
    cv2.destroyWindow("Suma")
    resta = cv2.subtract(imagen1,imagen2)
    cv2.imshow("Resta",resta)
    cv2.moveWindow("Resta",900,200)

#Multiplicacion
resultado = cv2.waitKey(0)
```

```
if resultado == 99:
    cv2.destroyWindow("Resta")
    multiplicacion = cv2.multiply(imagen1,imagen2)
    cv2.imshow("Multiplicacion",multiplicacion)
    cv2.moveWindow("Multiplicacion",900,200)

#Division
resultado = cv2.waitKey(0)
if resultado == 99:
    cv2.destroyWindow("Multiplicacion")
    division = cv2.divide(imagen1,imagen2)
    cv2.imshow("Division",division)
    cv2.moveWindow("Division",900,200)

#Logaritmo
resultado = cv2.waitKey(0)
if resultado == 99:
    cv2.destroyAllWindows()

    c = 255 / np.log(1 + np.max(imagen1))
    log_image1 = c * (np.log(imagen1 + 1))
    log_image1 = np.array(log_image1, dtype = np.uint8)

    d = 255 / np.log(1 + np.max(imagen2))
    log_image2 = d * (np.log(imagen2 + 1))
    log_image2 = np.array(log_image2, dtype = np.uint8)

    cv2.imshow("Logaritmo 1",log_image1)
    cv2.moveWindow("Logaritmo 1",0,200)
    cv2.imshow("Logaritmo 2",log_image2)
    cv2.moveWindow("Logaritmo 2",1985,200)

#Derivada
resultado = cv2.waitKey(0)
if resultado == 99:
    cv2.destroyAllWindows()
    derivada1 = cv2.Laplacian(imagen1,cv2.CV_64F)
    derivada2 = cv2.Laplacian(imagen2,cv2.CV_64F)
    cv2.imshow("Derivada 1",derivada1)
    cv2.moveWindow("Derivada 1",0,200)
    cv2.imshow("Derivada 2",derivada2)
    cv2.moveWindow("Derivada 2",1985,200)


#Potencia
resultado = cv2.waitKey(0)
if resultado == 99:
    cv2.destroyAllWindows()
    potencia1 = cv2.pow(imagen1,2)
    potencia2 = cv2.pow(imagen2,2)
    cv2.imshow("Potencia 1",potencia1)
```

```
    cv2.moveWindow("Potencia 1",0,200)
    cv2.imshow("Potencia 2",potencia2)
    cv2.moveWindow("Potencia 2",1985,200)

#Conjuncion
resultado = cv2.waitKey(0)
if resultado == 99:
    cv2.destroyAllWindows()
    cv2.imshow("Batman",imagen1)
    cv2.moveWindow("Batman",0,200)
    cv2.imshow("Iron Man",imagen2)
    cv2.moveWindow("Iron Man",1985,200)
    conjuncion = cv2.bitwise_and(imagen1,imagen2)
    cv2.imshow("Conjuncion",conjuncion)
    cv2.moveWindow("Conjuncion",900,200)

#Disyuncion
resultado = cv2.waitKey(0)
if resultado == 99:
    cv2.destroyWindow("Conjuncion")
    disyuncion = cv2.bitwise_or(imagen1,imagen2)
    cv2.imshow("Disyuncion",disyuncion)
    cv2.moveWindow("Disyuncion",900,200)

#Negacion
resultado = cv2.waitKey(0)
if resultado == 99:
    cv2.destroyWindow("Disyuncion")
    negacion1 = cv2.bitwise_not(imagen1)
    negacion2 = cv2.bitwise_not(imagen2)
    cv2.destroyWindow("Batman")
    cv2.destroyWindow("Iron Man")
    cv2.imshow("Negacion 1",negacion1)
    cv2.moveWindow("Negacion 1",0,200)
    cv2.imshow("Negacion 2",negacion2)
    cv2.moveWindow("Negacion 2",1985,200)

#Traslacion
resultado = cv2.waitKey(0)
if resultado == 99:
    cv2.destroyWindow("Negacion 1")
    cv2.destroyWindow("Negacion 2")
    T1 = np.float32([[1,0,210],[0,1,20]])
    traslacion1 = cv2.warpAffine(imagen1,T1,(400,400))
    cv2.imshow("Traslacion 1",traslacion1)
    cv2.moveWindow("Traslacion 1",0,200)
    T2 = np.float32([[1,0,-210],[0,1,100]])
    traslacion2 = cv2.warpAffine(imagen2,T2,(400,400))
    cv2.imshow("Traslacion 2",traslacion2)
    cv2.moveWindow("Traslacion 2",1985,200)
```

```
#Escalado
resultado = cv2.waitKey(0)
if resultado == 99:
    cv2.destroyWindow("Traslacion 1")
    cv2.destroyWindow("Traslacion 2")
    escalado1 = cv2.resize(imagen1, (700,1050), interpolation =
cv2.INTER_AREA)
    escalado2 = cv2.resize(imagen2, (700,1050), interpolation =
cv2.INTER_AREA)
    cv2.imshow("Escalado 1",escalado1)
    cv2.moveWindow("Escalado 1",0,200)
    cv2.imshow("Escalado 2",escalado2)
    cv2.moveWindow("Escalado 2",1785,200)

#Rotacion
resultado = cv2.waitKey(0)
if resultado == 99:
    cv2.destroyWindow("Escalado 1")
    cv2.destroyWindow("Escalado 2")
    height, width = imagen1.shape[:2]
    center = (width/2, height/2)
    rotate_matrix = cv2.getRotationMatrix2D(center=center, angle=45,
scale=1)
    rotada1 = cv2.warpAffine(src=imagen1, M=rotate_matrix, dsize=(width,
height))
    rotada2 = cv2.warpAffine(src=imagen2, M=rotate_matrix, dsize=(width,
height))
    cv2.imshow("Rotada 1",rotada1)
    cv2.moveWindow("Rotada 1",0,200)
    cv2.imshow("Rotada 2",rotada2)
    cv2.moveWindow("Rotada 2",2035,200)

#Traslación a fin
resultado = cv2.waitKey(0)
if resultado == 99:
    cv2.destroyWindow("Rotada 1")
    cv2.destroyWindow("Rotada 2")

    rowsA,colsA,chA = imagen1.shape
    p1 = np.float32([[50,300],[400,100],[100,100]])
    p2 = np.float32([[300,100],[300,50],[80,150]])
    A1 = cv2.getAffineTransform(p1,p2)
    Affine1 = cv2.warpAffine(imagen1,A1,(colsA,rowsA))

    rowsB,colsB,chB = imagen2.shape
    p3 = np.float32([[300,100],[300,50],[80,150]])
    p4 = np.float32([[50,300],[400,100],[100,100]])
    A2 = cv2.getAffineTransform(p3,p4)
    Affine2 = cv2.warpAffine(imagen2,A2,(colsB,rowsB))
```

```
        cv2.imshow("Traslacion 1",Affine1)
        cv2.moveWindow("Traslacion 1",0,200)
        cv2.imshow("Traslacion 2",Affine2)
        cv2.moveWindow("Traslacion 2",1985,200)


    #Transpuesta
    resultado = cv2.waitKey(0)
    if resultado == 99:
        cv2.destroyWindow("Traslacion 1")
        cv2.destroyWindow("Traslacion 2")
        transpuesta1 = cv2.transpose(imagen1)
        transpuesta2 = cv2.transpose(imagen2)
        cv2.imshow("Transpuesta 1",transpuesta1)
        cv2.moveWindow("Transpuesta 1",0,400)
        cv2.imshow("Transpuesta 2",transpuesta2)
        cv2.moveWindow("Transpuesta 2",1685,400)
```

Repositorio:

https://github.com/Cesarsp41/Practica-2