CENTRO DE ENSEÑANZA TÉCNICA INDUSTRIAL

INGENIERÍA EN MECATRÓNICA

Visión artificial

Práctica 2:
Operaciones básicas

Alumna:
Vanessa Aguirre Diaz

Registro:
22310274

Fecha:
22 de marzo del 2025

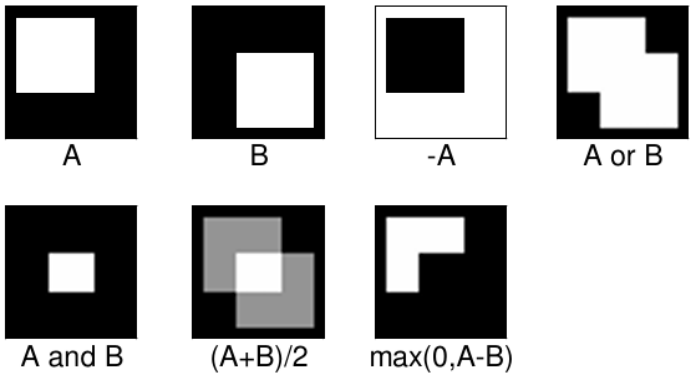
Desarrollo teórico: Operaciones Básicas entre píxeles.

Operaciones aritmético-lógicas

Estas operaciones son, con diferencia, las más usadas a cualquier nivel en un sistema de tratamiento de imágenes, ya que son las que se utilizan para dar valores a los píxeles de las imágenes. Las operaciones básicas son:

- Conjunción: Operación lógica AND entre los bits de dos imágenes.
- Disyunción: Operación lógica OR entre los bits de dos imágenes.
- Negación: Inversión de los bits que forman una imagen.
- Suma: Suma de los valores de los píxeles de dos imágenes.
- Resta: Resta de los valores de los píxeles de dos imágenes.
- Multiplicación: Multiplicación de los valores de los píxeles de una imagen por los de otra.
- División: División de los valores de los píxeles de una imagen entre los de otra.

Se ha visto que en imágenes en niveles de gris se suele utilizar el valor 255 para representar el blanco y el 0 para el negro. Cuando se realiza operaciones aritméticas se debe tener la precaución de verificar que el resultado R de una operación cae dentro del dominio de valores permitidos.



Operaciones geométricas

Si se expresa los puntos en coordenadas homogéneas, todas las transformaciones se pueden tratar mediante multiplicación de matrices. Las operaciones geométricas más usuales son:

- Traslación: Movimiento de los píxeles de una imagen según un vector de movimiento. La siguiente transformación muestra el resultado de trasladar el punto (x, y) según el vector (dx, dy), obteniendo el punto (x', y').

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Escalado: Cambio del tamaño de una imagen. La siguiente transformación muestra el resultado de escalar el punto (x, y) en un factor (sx, sy), obteniendo el punto (x', y').

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Rotación: Giro de los píxeles de una imagen en torno al origen de coordenadas. La siguiente transformación muestra el resultado de rotar el punto (x, y) un ángulo θ, obteniendo el punto (x', y').

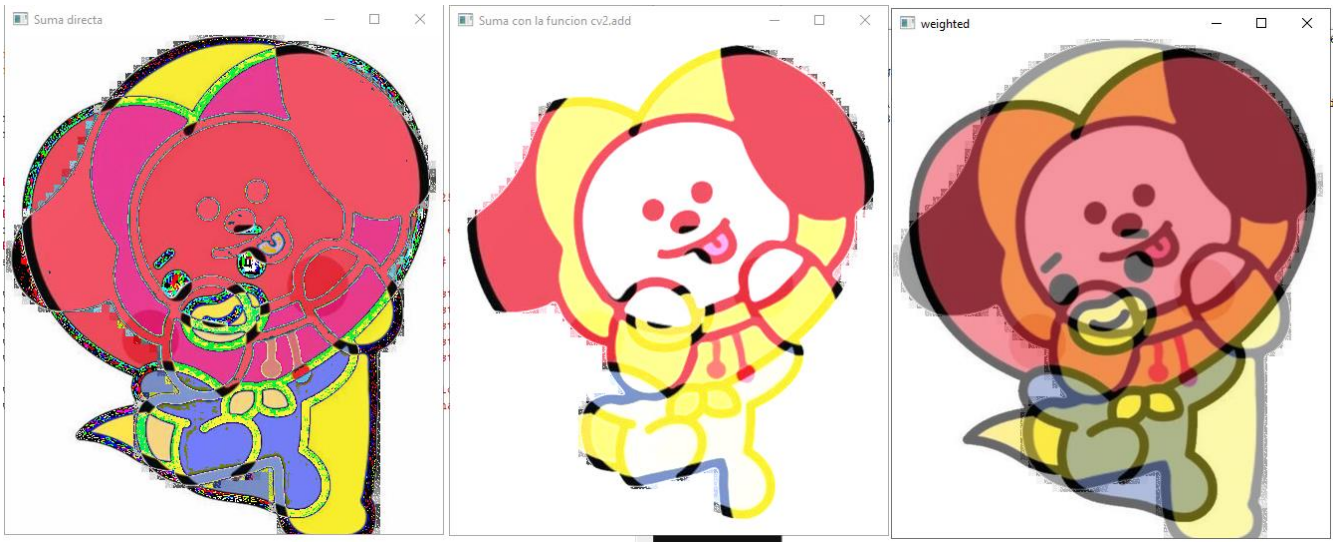
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -sen(\theta) & 0 \\ sen(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Desarrollo práctico

1. Suma

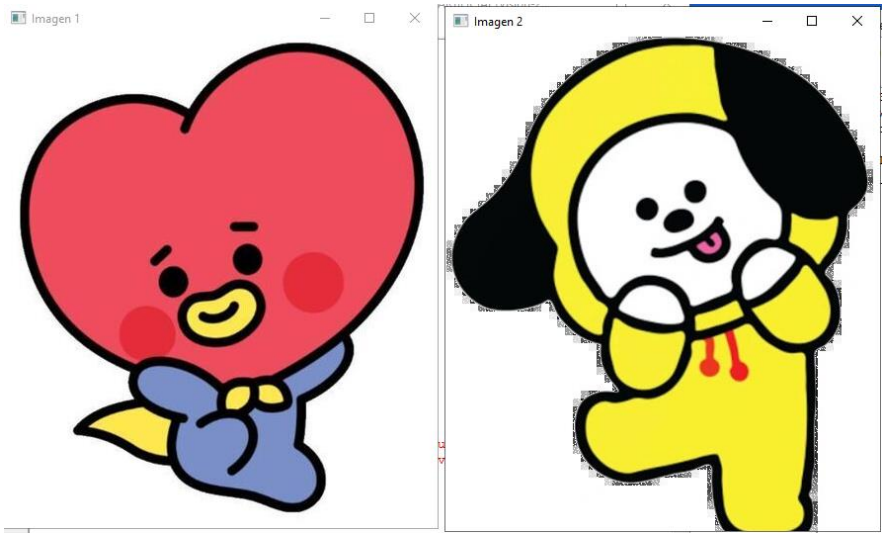
```
# Practica 2: Operaciones Aritmeticas
# Vanessa Aguirre Diaz

# Importar librerias:
import cv2
import numpy as np

# 440 x 500
img1 = cv2.imread('tata.jpg')       # Cargar imagen 1
img2 = cv2.imread('chimmy.jpg')     # Cargar imagen 2

# Sumar ambas imagenes pixel por pixel
#Pt.1:
add = img1+img2                 # Si la suma supera los 255 habrá un desbordamiento
#Pt.2:
add_cv2 = cv2.add(img1,img2)    # Si la suma excede 255, el valor se limita a 255 (no
hay desbordamiento).
#Pt.3:
weighted = cv2.addWeighted(img1, 0.6, img2, 0.4, 0)     # Cada imagen tendrá un peso
diferente
```

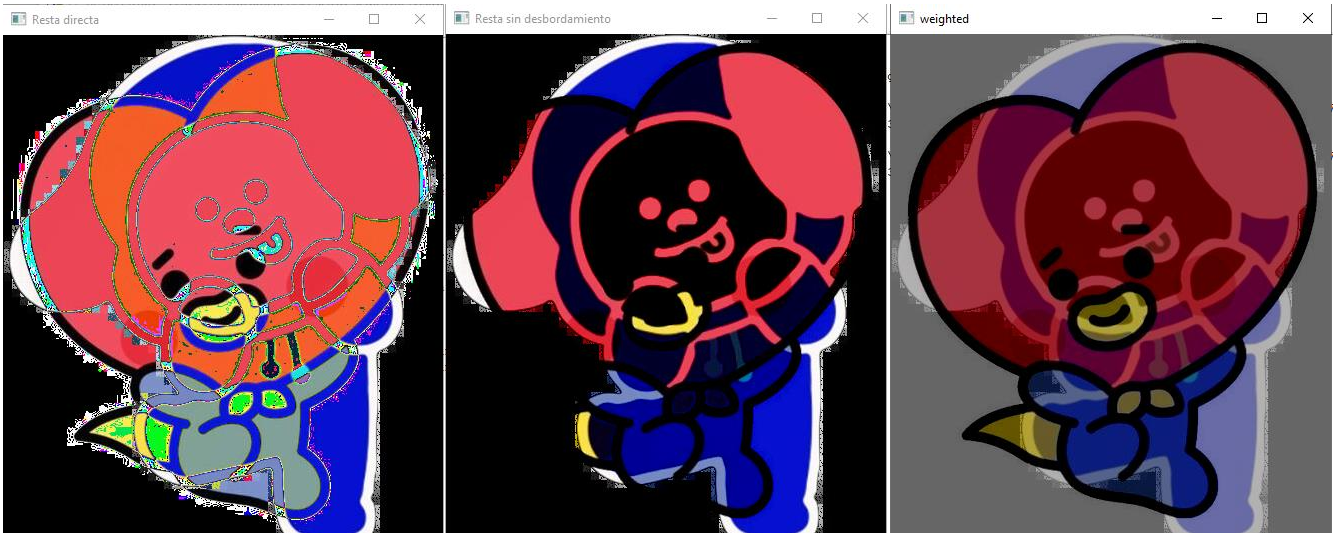
```
cv2.imshow('Imagen 1',img1)                                  # Mostrar la imagen 1
cv2.imshow('Imagen 2',img2)                                  # Mostrar la imagen 2
cv2.imshow('Suma directa',add)                               # Mostrar la imagen que resulta de
la suma directa con signo
cv2.imshow('Suma con la funcion cv2.add',add_cv2)    # Mostrar la imagen que resulta de
la suma con la funcion cv2.add
cv2.imshow('weighted',weighted)                              # Mostrar la imagen con diferentes
pesos

cv2.waitKey(0)                  # Esperar a que el usuario presione una tecla para
cerrar la ventana
cv2.destroyAllWindows()         # Cerrar todas las ventanas de OpenCV.
```



2. Resta

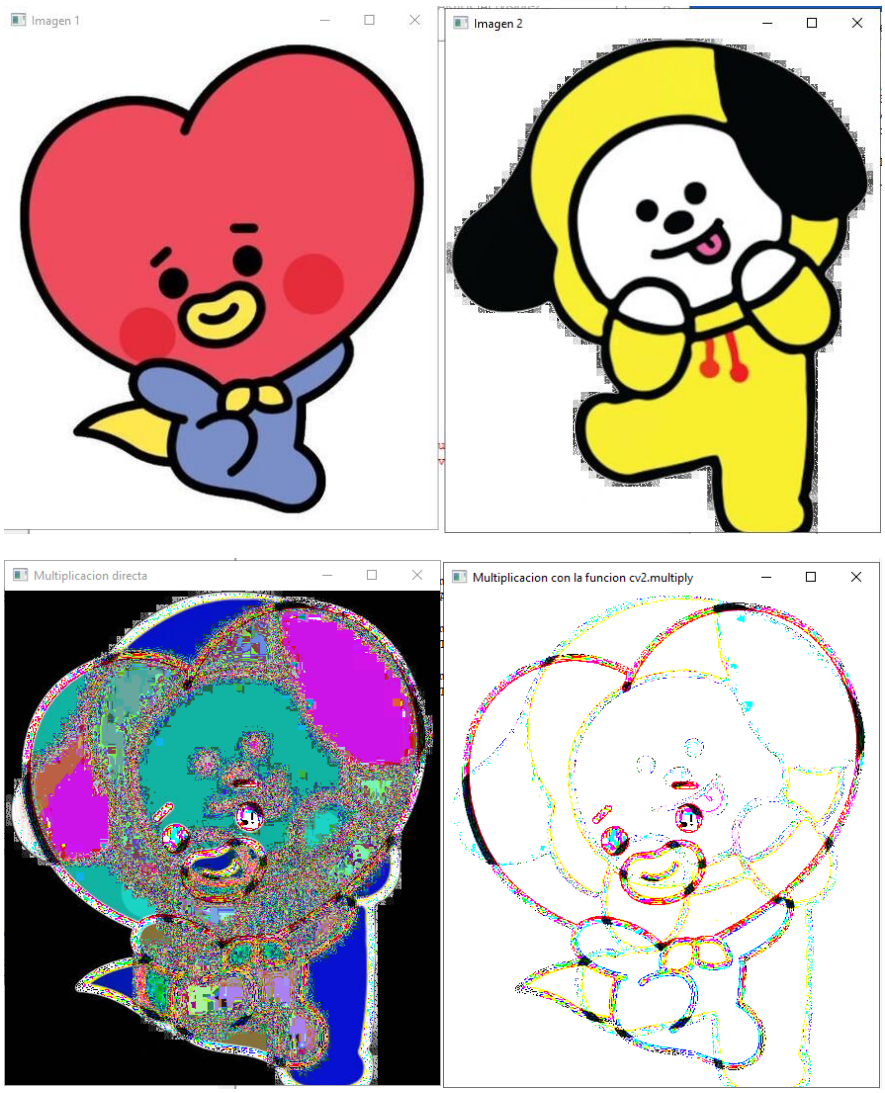
```
# Practica 2: Operaciones Aritmeticas... Resta
# Vanessa Aguirre Diaz

# Importar librerias:
import cv2
import numpy as np

# 440 x 500
img1 = cv2.imread('tata.jpg')       # Cargar imagen 1
img2 = cv2.imread('chimmy.jpg')     # Cargar imagen 2

# Sumar ambas imagenes pixel por pixel
#Pt.1:
add = img1-img2                 # Si la resta supera los 0 habrá un desbordamiento
negativo
#Pt.2:
add_cv2 = cv2.subtract(img1,img2)    # Si la resta es menor a 0, el valor se limita a 0
(no hay desbordamiento negativo).
#Pt.3:
weighted = cv2.addWeighted(img1, 0.7, img2, -0.3, 0)     # Cada imagen tendrá un peso
diferente

cv2.imshow('Imagen 1',img1)         # Mostrar la imagen 1
cv2.imshow('Imagen 2',img2)         # Mostrar la imagen 2
cv2.imshow('Resta directa',add)                     # Mostrar la imagen que resulta de
la suma directa con signo
cv2.imshow('Resta sin desbordamiento',add_cv2)      # Mostrar la imagen que resulta de
la resta con la funcion cv2.subtract
cv2.imshow('weighted',weighted)                     # Mostrar la imagen con diferentes
pesos

cv2.waitKey(0)                                      # Esperar a que el usuario presione
una tecla para cerrar la ventana
cv2.destroyAllWindows()                             # Cerrar todas las ventanas
```





3. Multiplicación

```
# Practica 2: Operaciones Aritmeticas... Multiplicacion
# Vanessa Aguirre Diaz

# Importar librerias:
import cv2
import numpy as np

# 440 x 500
img1 = cv2.imread('tata.jpg')       # Cargar imagen 1
img2 = cv2.imread('chimmy.jpg')     # Cargar imagen 2

# Multiplicar ambas imagenes pixel por pixel
#Pt.1:
multiplicacion = img1*img2              # Si la multiplicacion supera los 255 habrá un
desbordamiento
#Pt.2:
mult_cv2 = cv2.multiply(img1,img2)      # Si la multiplicacion excede 255, el valor se
limita a 255 (no hay desbordamiento).
                                        # Ideal para aplicar máscaras o efectos de
fusión

cv2.imshow('Imagen 1',img1)                                         # Mostrar la imagen
1
cv2.imshow('Imagen 2',img2)                                         # Mostrar la imagen
2
cv2.imshow('Multiplicacion directa',multiplicacion)                 # Mostrar la imagen
que resulta de la multiplicacion directa con signo
cv2.imshow('Multiplicacion con la funcion cv2.multiply',mult_cv2)   # Mostrar la imagen
que resulta de la multiplicacion con la funcion cv2.multiply

cv2.waitKey(0)                  # Esperar a que el usuario presione una tecla para
cerrar la ventana
cv2.destroyAllWindows()         # Cerrar todas las ventanas de OpenCV.
```



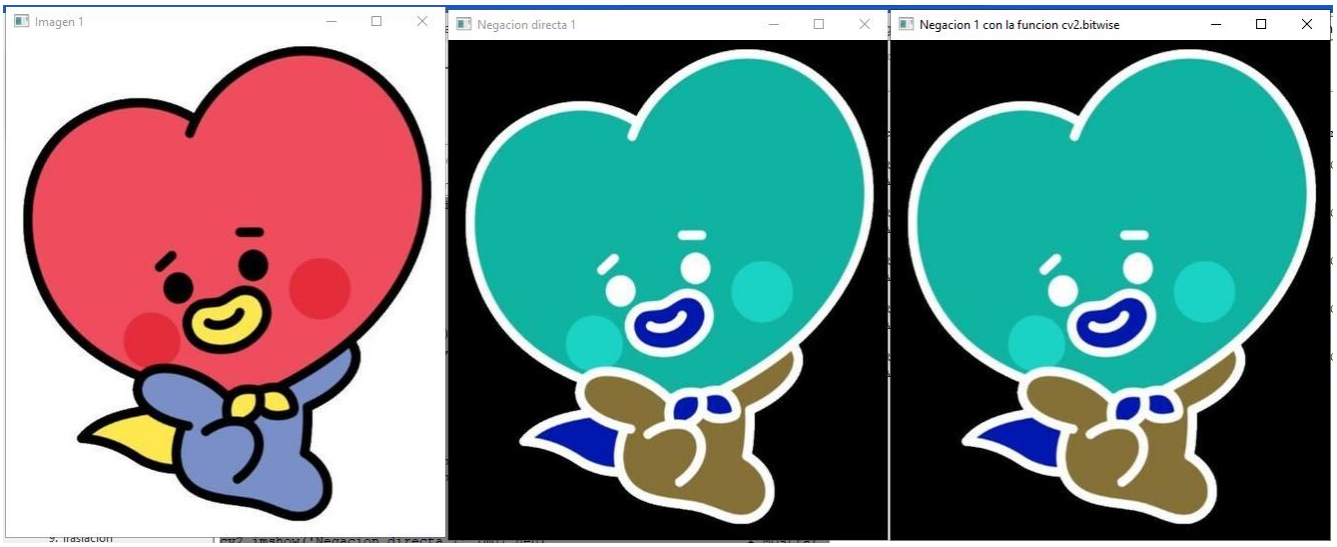
4. División

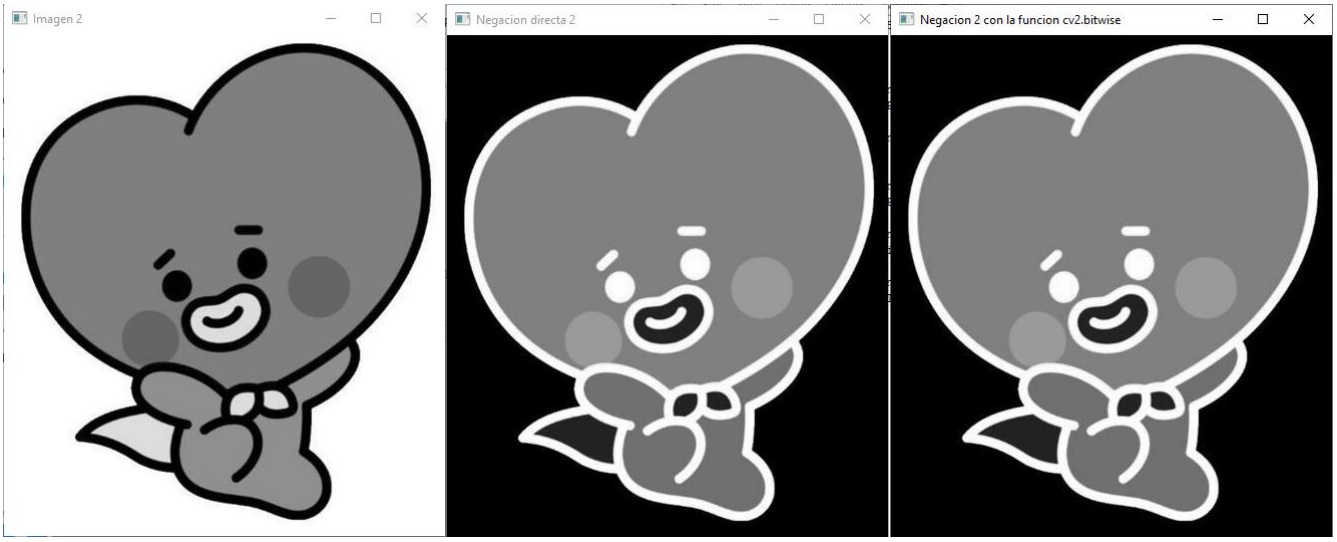
```
# Practica 2: Operaciones Aritmeticas... Division
# Vanessa Aguirre Diaz

# Importar librerias:
import cv2
import numpy as np

# 440 x 500
img1 = cv2.imread('tata.jpg')       # Cargar imagen 1
img2 = cv2.imread('chimmy.jpg')     # Cargar imagen 2

# Multiplicar ambas imagenes pixel por pixel
#Pt.1:
division = img1 / (img2 + 1e-5)     # Se añade epsilon para evitar división por 0
                                    # Realiza la división, pero si un píxel en img2 es
0, generará valores muy altos
#Pt.2:
div_cv2 = cv2.divide(img1,img2)     # Evita divisiones por 0 automáticamente.
```

```
cv2.imshow('Imagen 1',img1)         # Mostrar la imagen 1
cv2.imshow('Imagen 2',img2)         # Mostrar la imagen 2
cv2.imshow('Division directa',division)                          # Mostrar la imagen que
resulta de la division directa con signo
cv2.imshow('Division con la funcion cv2.divide',div_cv2)        # Mostrar la imagen que
resulta de la division con la funcion cv2.divide

cv2.waitKey(0)                  # Esperar a que el usuario presione una tecla para
cerrar la ventana
cv2.destroyAllWindows()         # Cerrar todas las ventanas de OpenCV.
```

5. Negación

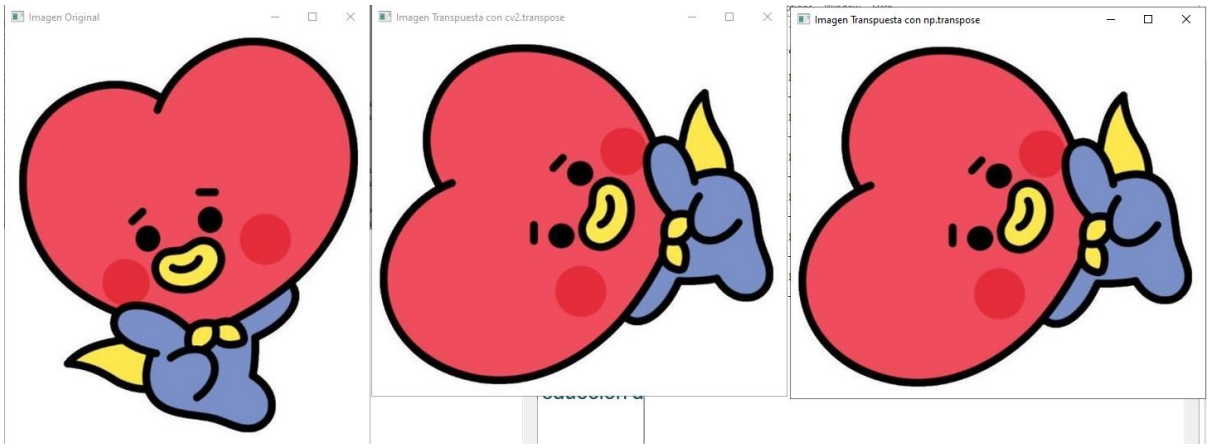
```
# Practica 2: Operaciones Aritmeticas... Negacion
# Vanessa Aguirre Diaz

# Importar librerias:
import cv2
import numpy as np

# 440 x 500
img1 = cv2.imread('tata.jpg')    # Cargar imagen
img2 = cv2.imread('tata.jpg', cv2.IMREAD_GRAYSCALE)    # Misma imagen a escala de grises

#Pt.1:
img1_neg = 255- img1                # Operacion directa
img2_neg = 255- img2                # Operacion directa

#Pt.2:
neg1_cv2 = cv2.bitwise_not(img1)
neg2_cv2 = cv2.bitwise_not(img2)

cv2.imshow('Imagen 1',img1)             # Mostrar la imagen
cv2.imshow('Imagen 2',img2)             # Mostrar la imagen
cv2.imshow('Negacion directa 1',img1_neg)                           # Mostrar la imagen
que resulta de la negacion directa
cv2.imshow('Negacion directa 2',img2_neg)                           # Mostrar la imagen
que resulta de la negacion directa
cv2.imshow('Negacion 1 con la funcion cv2.bitwise',neg1_cv2)       # Mostrar la imagen
que resulta de la negacion con la funcion cv2.bitwise
cv2.imshow('Negacion 2 con la funcion cv2.bitwise',neg2_cv2)       # Mostrar la imagen
que resulta de la negacion con la funcion cv2.bitwise

cv2.waitKey(0)                  # Esperar a que el usuario presione una tecla para
cerrar la ventana
cv2.destroyAllWindows()         # Cerrar todas las ventanas de OpenCV.
```





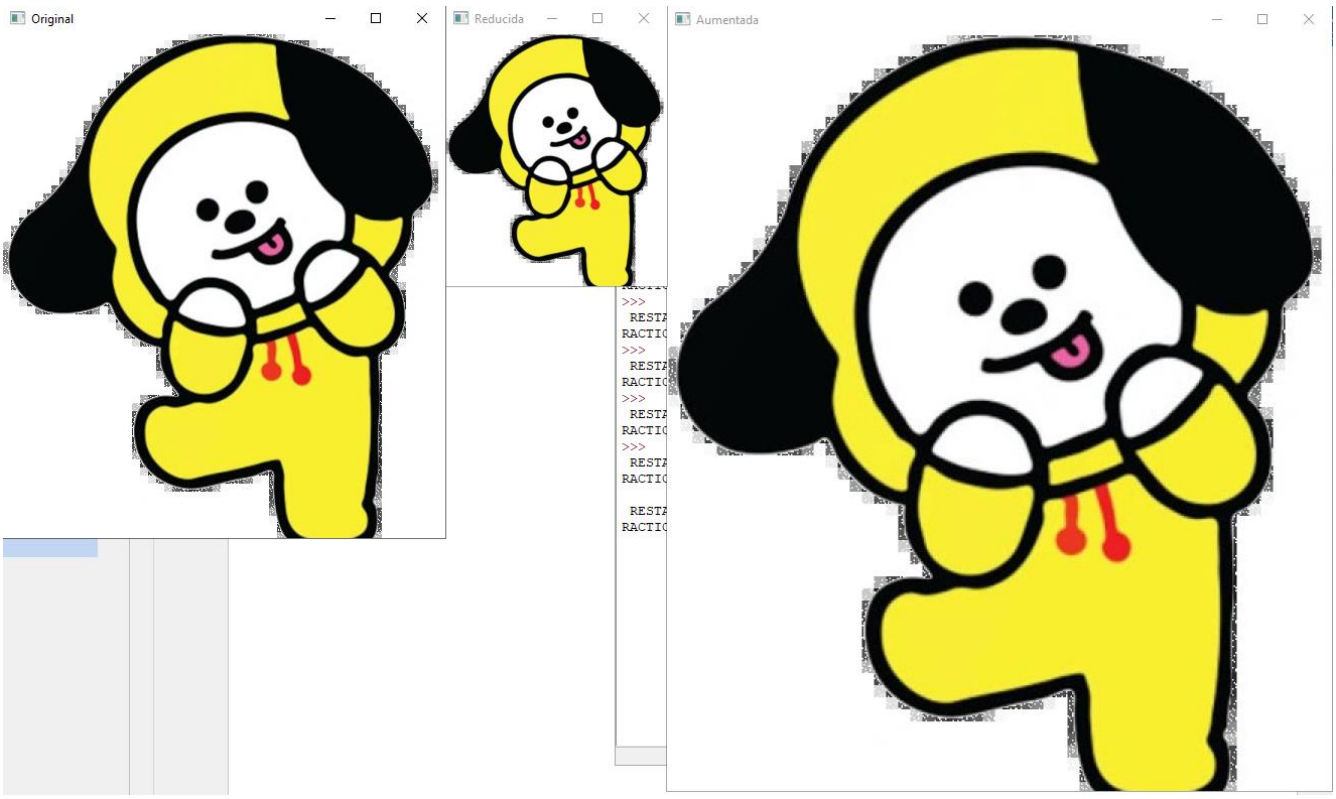
6. Transpuesta

```
# Practica 2: Operaciones Aritmeticas... Transpuesta
# Vanessa Aguirre Diaz

# Importar librerias:
import cv2
import numpy as np

img = cv2.imread('tata.jpg')  # Cargar imagen

img_trans1 = cv2.transpose(img)    # Aplicar transposición
img_trans2 = np.transpose(img, (1, 0, 2))  # Intercambia alto y ancho

# Mostrar resultados
cv2.imshow('Imagen Original', img)
cv2.imshow('Imagen Transpuesta con cv2.transpose', img_trans1)
cv2.imshow('Imagen Transpuesta con np.transpose', img_trans2)

cv2.waitKey(0)                  # Esperar a que el usuario presione una tecla para
cerrar la ventana
cv2.destroyAllWindows()         # Cerrar todas las ventanas de OpenCV.
```



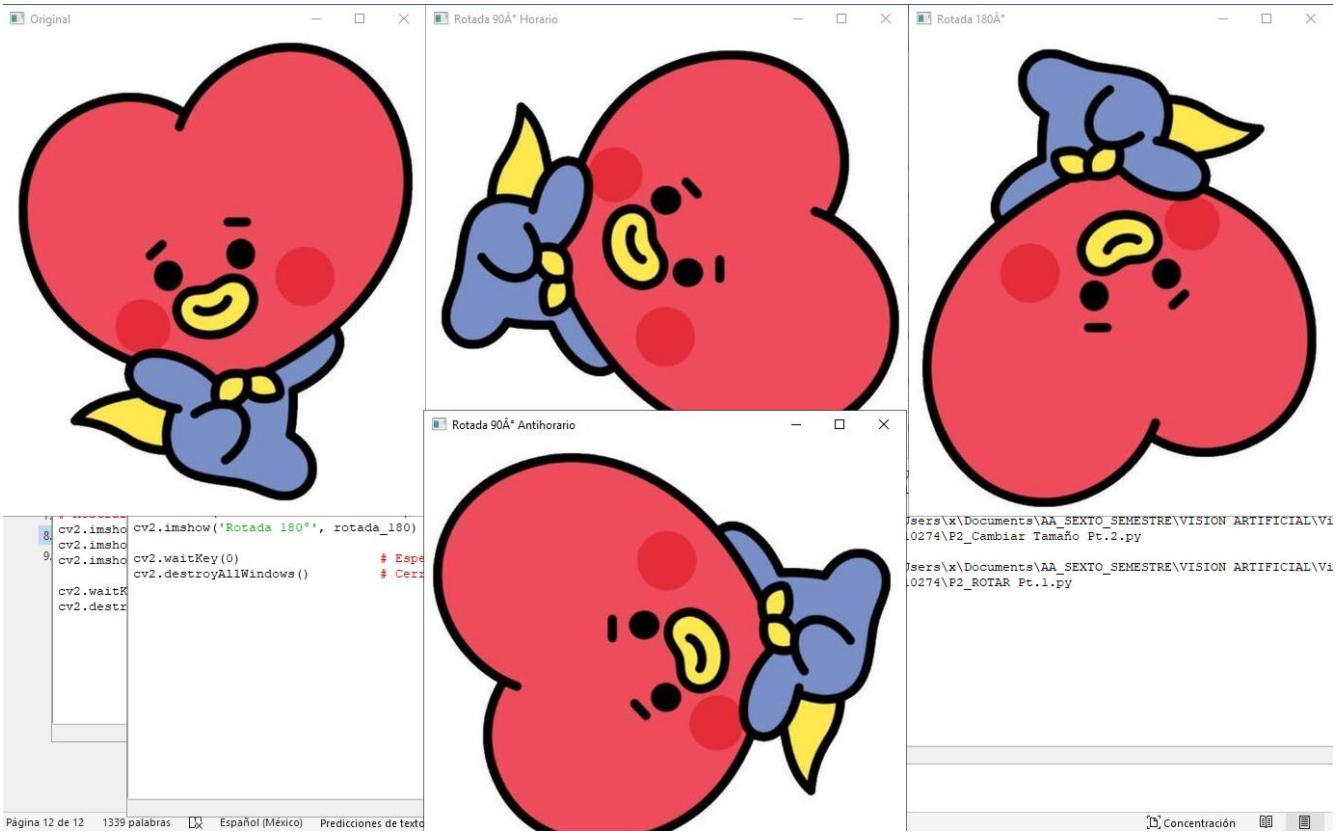
7. Aumento o reducción de tamaño

```
# Practica 2: Operaciones Aritmeticas... Redimencionar una imagen
# Vanessa Aguirre Diaz

# Importar librerias:
import cv2

# Cargar imagen
img = cv2.imread('chimmy.jpg')

# Obtener dimensiones originales
alto, ancho = img.shape[:2]

# Nuevo tamaño manteniendo la relación de aspecto (reducir a 50%)
factor1 = 0.5
factor2 = 1.5

nuevo_ancho1 = int(ancho * factor1)
nuevo_alto1 = int(alto * factor1)

nuevo_ancho2 = int(ancho * factor2)
nuevo_alto2 = int(alto * factor2)

# Redimensionar
img_reducida = cv2.resize(img, (nuevo_ancho1, nuevo_alto1),
interpolation=cv2.INTER_AREA)
img_aumentada = cv2.resize(img, (nuevo_ancho2, nuevo_alto2),
interpolation=cv2.INTER_AREA)

# Mostrar imágenes
cv2.imshow('Original', img)
cv2.imshow('Reducida', img_reducida)
cv2.imshow('Aumentada', img_aumentada)

cv2.waitKey(0)                  # Esperar a que el usuario presione una tecla para
cerrar la ventana
cv2.destroyAllWindows()         # Cerrar todas las ventanas de OpenCV.
```



8. Rotación

```
# Practica 2: Operaciones Aritmeticas... Rotar una imagen
# Vanessa Aguirre Diaz

# Importar librerias:
import cv2

# Cargar imagen
img = cv2.imread('tata.jpg')

# Rotar 90° en sentido horario
rotada_90 = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)

# Rotar 90° en sentido antihorario
rotada_90_anti = cv2.rotate(img, cv2.ROTATE_90_COUNTERCLOCKWISE)

# Rotar 180°
rotada_180 = cv2.rotate(img, cv2.ROTATE_180)

# Mostrar imágenes
cv2.imshow('Original', img)
cv2.imshow('Rotada 90° Horario', rotada_90)
cv2.imshow('Rotada 90° Antihorario', rotada_90_anti)
cv2.imshow('Rotada 180°', rotada_180)

cv2.waitKey(0)                  # Esperar a que el usuario presione una tecla para
cerrar la ventana
cv2.destroyAllWindows()         # Cerrar todas las ventanas de OpenCV.
```



9. Traslación

```
# Practica 2: Operaciones Aritmeticas... Traslacion de una imagen
# Vanessa Aguirre Diaz

# Importar librerias:
import cv2
import numpy as np

# Cargar imagen
img = cv2.imread('tata.jpg')

# Dimensiones originales
alto, ancho = img.shape[:2]

# Definir traslación (mueve la imagen 150 píxeles a la derecha y 100 abajo)
Tx, Ty = 150, 100

# Crear matriz de traslación
M = np.float32([[1, 0, Tx], [0, 1, Ty]])

# Ajustar el tamaño del lienzo
nuevo_ancho = ancho + Tx
nuevo_alto = alto + Ty

# Aplicar traslación con nuevo tamaño
```

```
img_trasladada = cv2.warpAffine(img, M, (nuevo_ancho, nuevo_alto))

# Mostrar imágenes
cv2.imshow('Imagen Original', img)
cv2.imshow('Imagen Trasladada (Lienzo Ajustado)', img_trasladada)

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