ING. Mecatrónica

visión Artificial

Saul Isaac Limon Bautista

22310278

Explicación de Operaciones Aritméticas en Imágenes con OpenCV

En el procesamiento de imágenes, las operaciones aritméticas permiten modificar los valores de los píxeles para ajustar brillo, contraste, rotación, traslación, entre otros. OpenCV permite realizar estas operaciones de manera eficiente manipulando matrices de píxeles. A continuación, se explican diferentes operaciones y sus implementaciones en Python.

1. Suma (+) - Aumento de Brillo

La suma incrementa el valor de cada píxel, haciendo que la imagen se vea más clara. Si el valor supera 255, se mantiene en 255 para evitar saturaciones.

```
import cv2

import numpy as np

img = cv2.imread('watch.jpg')  # Cargar imagen a color

brighter_img = cv2.add(img, np.array([50.0]))  # Aumentar brillo en 50 niveles

cv2.imshow('Imagen con mayor brillo', brighter_img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

2. Resta (-) - Reducción de Brillo

La resta disminuye la intensidad de los píxeles, oscureciendo la imagen. Si el resultado es menor a 0, se mantiene en 0 para evitar valores negativos.

```
import cv2

import numpy as np

img = cv2.imread('watch.jpg')

darker_img = cv2.subtract(img, np.array([50.0]))  # Reducir brillo en 50 niveles

cv2.imshow('Imagen con menor brillo', darker_img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

3. Multiplicación (*) - Ajuste de Contraste

Multiplicar cada píxel por un valor mayor a 1 aumenta el contraste, mientras que un valor entre 0 y 1 lo reduce.

```
import cv2

import numpy as np

img = cv2.imread('watch.jpg')

higher_contrast = cv2.multiply(img, np.array([1.5]))  # Aumentar contraste 1.5x

cv2.imshow('Imagen con mayor contraste', higher_contrast)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

4. División (/) - Reducción de Intensidad

La división reduce la intensidad de los píxeles, creando una imagen más apagada o normalizada.

```
import cv2

import numpy as np

img = cv2.imread('watch.jpg')

low_intensity = cv2.divide(img, np.array([2.0]))  # Reducir la intensidad a la mitad

cv2.imshow('Imagen con menor intensidad', low_intensity)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

5. Negación - Inversión de Colores**

La inversión de colores se logra restando cada píxel a 255, generando un negativo de la imagen.

```
import cv2

img = cv2.imread('watch.jpg')

negative_img = 255 - img  # Inversión de colores

cv2.imshow('Imagen negativa', negative_img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

6. Transpuesta - Intercambio de Filas y Columnas

Transponer una imagen intercambia las filas por columnas, similar a una rotación de 90 grados.

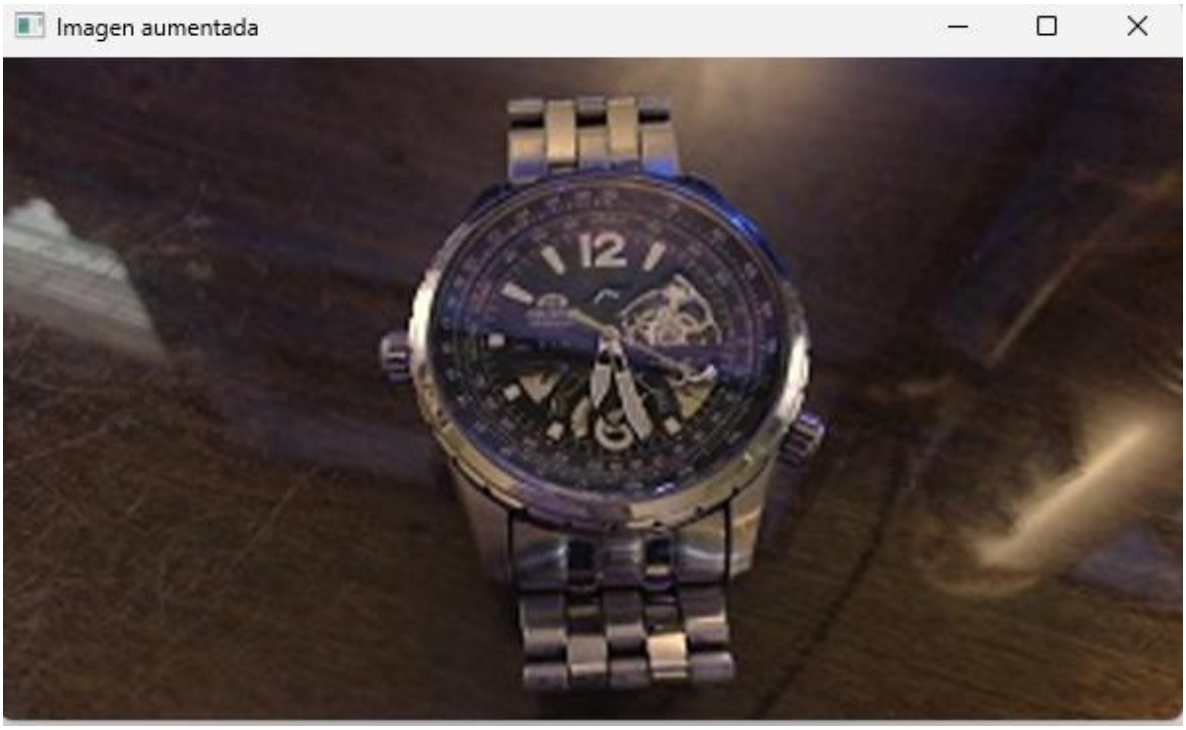
```
import cv2

img = cv2.imread('watch.jpg')

transposed_img = cv2.transpose(img)  # Aplicar transposición

cv2.imshow('Imagen transpuesta', transposed_img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

7. Escalado - Aumento o Reducción de Tamaño

Redimensionar una imagen permite ajustar su tamaño sin distorsionar sus proporciones.

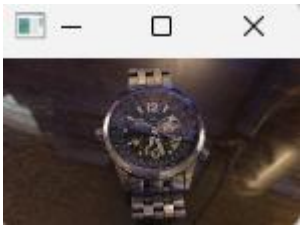
Aumentar Tamaño:

```
import cv2

img = cv2.imread('watch.jpg')

bigger_img = cv2.resize(img, None, fx=2, fy=2, interpolation=cv2.INTER_CUBIC)  # Doble de tamaño

cv2.imshow('Imagen aumentada', bigger_img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```



Reducir Tamaño:

```
import cv2

img = cv2.imread('watch.jpg')

smaller_img = cv2.resize(img, None, fx=0.5, fy=0.5, interpolation=cv2.INTER_AREA)  # Mitad de tamaño

cv2.imshow('Imagen reducida', smaller_img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```



8. Rotación de la Imagen

Explicación: Se usa una matriz de transformación afín para rotar la imagen en un ángulo determinado.

```
import cv2

import numpy as np

img = cv2.imread('watch.jpg')

(h, w) = img.shape[:2]

center = (w // 2, h // 2)

matrix = cv2.getRotationMatrix2D(center, 45, 1.0)  # Rotación de 45 grados

rotated_img = cv2.warpAffine(img, matrix, (w, h))

cv2.imshow('Imagen rotada', rotated_img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

9. Traslación - Movimiento de la Imagen

Se crea una matriz de transformación que mueve la imagen en X e Y.

```
import cv2

import numpy as np

img = cv2.imread('watch.jpg')

(h, w) = img.shape[:2]

matrix = np.float32([[1, 0, 50], [0, 1, 100]])  # Mover 50px en X, 100px en Y

translated_img = cv2.warpAffine(img, matrix, (w, h))

cv2.imshow('Imagen trasladada', translated_img)

cv2.waitKey(0)

cv2.destroyAllWindows
```