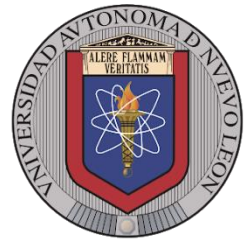


UANL

UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA Y ELECTRICA

FIME

Nombre: Roberto Erick Aguilar Morales

Matricula: 1871004

Carrera: Ingeniero en Tecnologías del Software

2.– Operadores Básicos

Materia: VISION COMPUTACIONAL LABORATORIO

Docente: RAYMUNDO SAID ZAMORA PEQUEÑO

Hora: N1-N2 **Días: Miércoles**

Fecha: 06/10/24

Objetivo

Utilizar diferentes operadores para facilitar el procesamiento complejo de imágenes

Marco teórico

Visión Computacional

La visión computacional es un campo de la inteligencia artificial y las ciencias de la computación que busca replicar las capacidades visuales humanas mediante el análisis y procesamiento de imágenes digitales. En este campo, los algoritmos interpretan, analizan y procesan imágenes para extraer información útil, reconocer patrones y realizar operaciones automáticas.

Tratamiento de Imágenes

El tratamiento de imágenes implica aplicar diversas transformaciones a imágenes digitales para mejorar su calidad, extraer características, o prepararlas para una etapa posterior de análisis. Este proceso puede incluir operaciones como:

- **Ajustes de brillo y contraste**
- **Filtros de suavizado o nitidez**

Las herramientas para el tratamiento de imágenes son fundamentales en áreas como el reconocimiento de objetos, el seguimiento de movimientos y la mejora de imágenes para visualización.

Operadores Básicos en Tratamiento de Imágenes

Existen diversos operadores básicos utilizados en la manipulación de imágenes, que pueden aplicarse a cada píxel de la imagen para lograr distintos efectos:

- **Suma o resta de imágenes**: Se suman o restan valores de píxeles de dos imágenes, útil para mezclar imágenes o corregir iluminación.
- **Multiplicación o división**: Se utilizan para ajustar el brillo o el contraste de las imágenes.
- **AND, OR, XOR**: Operaciones lógicas que se pueden aplicar entre imágenes binarias.

Estas operaciones permiten manipular imágenes de manera sencilla, obteniendo resultados como imágenes combinadas o modificadas.

Método de Shifting

El "shifting" es una operación geométrica que consiste en desplazar la luz de una imagen en una dirección particular (horizontal o vertical) sin cambiar su forma. Esta técnica se utiliza comúnmente para crear efectos de sombras, corregir aproximaciones, o deducir que se interpone entre el objeto y la luz.

En OpenCV, esta operación se realiza aplicando una **transformación de traslación** mediante una matriz de transformación afín. Este tipo de transformaciones son comunes en la manipulación de imágenes y son parte fundamental de procesos como la visión por computadora en robots o vehículos autónomos, donde es importante corregir la perspectiva o la posición de objetos dentro de una imagen.

Librería OpenCV

OpenCV (Open Source Computer Vision Library) es una biblioteca de código abierto que proporciona herramientas para el procesamiento de imágenes y visión computacional en tiempo real. OpenCV soporta tanto imágenes como videos y ofrece una amplia variedad de funciones para el procesamiento, tales como:

- **Cargar y guardar imágenes**
- **Manipulación de color** (conversión entre espacios de color)
- **Aplicación de filtros** para suavizar o mejorar detalles
- **Transformaciones geométricas** (rotación, escalado, traslación)

OpenCV es ampliamente utilizado en la investigación y el desarrollo de soluciones de visión computacional en áreas como la robótica, el análisis de imágenes médicas, y la conducción autónoma.

Introducción

Este programa aplica operadores básicos en el tratamiento de imágenes utilizando la librería OpenCV. Las operaciones incluyen el ajuste de brillo y contraste, la creación de una copia y el negativo de la imagen, y la aplicación de shifting en distintas direcciones.

Importar librerías

Se utilizarán 2 librerías esenciales para la creación de este programa

```
import cv2
import numpy as np
```

- **cv2**: Es la librería principal de OpenCV, que ofrece funciones para el procesamiento de imágenes.
- **numpy (np):** Se utiliza para manipular matrices, que en este caso representan las imágenes.

Cargar la Imagen

La imagen se carga desde un archivo utilizando la función cv2.imread. Si la imagen no se encuentra en el directorio correcto o no se puede cargar, se imprime un mensaje de error.

```
# Cargar la imagen
imagen_a_color = cv2.imread("Incendio.png")

if imagen_a_color is None:
    print("Error: No se pudo cargar la imagen. Asegúrate de que el archivo está en el directorio correcto.")
```

Conversión a Escala de Grises

Se convierte la imagen a escala de grises después de ser cargada para facilitar las operaciones de procesamiento:

```
    # Convertir la imagen en blanco y negro
    imagen = cv2.cvtColor(imagen_a_color, cv2.COLOR_BGR2GRAY)

    # NOTA: se convierte en blanco y negro despues de leerse porque sino pierde pixeles
```

La conversión a escala de grises permite aplicar operaciones como la manipulación del brillo y contraste de manera más eficiente.

Aclarar u Oscurecer la Imagen

Se utiliza una función para modificar el brillo de la imagen. La función cv2.convertScaleAbs ajusta los valores de los píxeles según un valor de brillo especificado por el usuario:

```
    # Función para aclarar u oscurecer la imagen
    def aclarar_oscurecer(imagen, valor):
        return cv2.convertScaleAbs(imagen, alpha=1, beta=valor)
```

El parámetro beta ajusta el brillo, donde valores positivos aclaran la imagen y valores negativos la oscurecen.

Copiar la Imagen

Esta función realiza una copia de la imagen, guarda la matriz de la imagen en un archivo CSV y almacena una nueva imagen en formato PNG:

```
    # Función para copiar la imagen
    def copiar_imagen(imagen):
        Imagen_copiada = imagen.copy()
        np.savetxt('imagen_copiada.csv', Imagen_copiada, delimiter=',',
fmt='%d')
        print("La matriz de la imagen copiada se ha guardado en
'imagen_copiada.csv'.")
        cv2.imwrite("Imagen copiada.png", Imagen_copiada)
        return Imagen_copiada
```

Invertir la Imagen (Negativo)

Para generar el negativo de la imagen, se utiliza la operación bitwise NOT:

```
    # Función para invertir la imagen (negativo)
    def negativo_imagen(imagen):
        Imagen_Negativa = cv2.bitwise_not(imagen)
        np.savetxt('imagen_negativa.csv', Imagen_Negativa, delimiter=',',
fmt='%d')
        print("La matriz de la imagen negativa se ha guardado en
'imagen_negativa.csv'.")
        cv2.imwrite("Imagen Negativa.png", Imagen_Negativa)
        return Imagen_Negativa
```

Ajustar el Brillo

El ajuste de brillo se logra multiplicando los valores de los píxeles por un factor de brillo. Se asegura que los valores estén dentro del rango 0-255:

```
    # Función para aumentar o disminuir el brillo
    def cambiar_brillo(imagen, factor):
        imagen = imagen.astype(np.float32)  # Convertir a float para evitar
overflow/underflow
        imagen = imagen * factor  # Multiplicar por el factor
        imagen = np.clip(imagen, 0, 255)  # Asegurar que los valores estén
en el rango 0-255
        return imagen.astype(np.uint8)  # Convertir de nuevo a uint8
```

Ajustar el Contraste

El ajuste de contraste se realiza calculando la media de la imagen y escalando los píxeles en función de esa media:

```
    # Función para aumentar o reducir el contraste
    def cambiar_contraste(imagen, factor):
        imagen = imagen.astype(np.float32)  # Convertir a float para evitar
overflow/underflow
        media = np.mean(imagen)  # Calcular la media de los píxeles
        imagen = (1 - factor) * media + factor * imagen  # Ajustar el
contraste
        imagen = np.clip(imagen, 0, 255)  # Asegurar que los valores estén
en el rango 0-255
        return imagen.astype(np.uint8)  # Convertir de nuevo a uint8
```

Aplicación de Shifting

El shifting es una transformación que desplaza la luz de laimagen en varias direcciones. Aquí se muestran varios métodos de shifting:

```
    # Shifting de izquierda a derecha
    def Shifting_1(imagen):
        shifting_izquierda_a_derecha = imagen.copy()

        #Recorrer el arreglo de la imagen en escala de grises
        for i in range(imagen.shape[0]):
            for j in range(imagen.shape[1]):
                if j != 0:
                    shifting_izquierda_a_derecha[i][j] =
shifting_izquierda_a_derecha[i][j-1]/ (imagen[i][j-1]/imagen[i][j])

        np.savetxt('Shifting_izq_a_der.csv', shifting_izquierda_a_derecha,
delimiter=',', fmt='%d')
        print("La matriz de la imagen Shifting izquierda a derecha se ha
guardado en 'Shifting_izq_a_der.csv'.")

        cv2.imwrite("shifting izquierda a derecha.png",
shifting_izquierda_a_derecha)
        return shifting_izquierda_a_derecha
```

```
# Shifting de derecha a izquierda
    def Shifting_2(imagen):
        shifting_dercha_a_izquierda = imagen.copy()

        # Recorrer el arreglo de la imagen en escala de grises de derecha a
izquierda
```

```
        for i in range(imagen.shape[0]):
            for j in range(imagen.shape[1] - 2, -1, -1):  # Inicia desde la
penúltima columna y va hacia la izquierda
                shifting_dercha_a_izquierda[i][j] =
shifting_dercha_a_izquierda[i][j+1] / (imagen[i][j+1] / imagen[i][j])

        np.savetxt('Shifting_der_a_izq.csv', shifting_dercha_a_izquierda,
delimiter=',', fmt='%d')

        np.savetxt('Shifting_der_a_izq.csv', shifting_dercha_a_izquierda,
delimiter=',', fmt='%d')
        print("La matriz de la imagen Shifting derecha a izquierda se ha
guardado en 'Shifting_der_a_izq.csv'.")

        cv2.imwrite("shifting derecha a izquierda.png",
shifting_dercha_a_izquierda)
        return shifting_dercha_a_izquierda
```

Los demás métodos surgen del mismo concepto solamente en direcciones diferentes con sus respectivos comentarios en cada uno de los métodos correspondientes

Valores constantes

Los siguientes valores fueron creados para mostrar los ejemplos de las clases anteriormente mencionados y transformar las mismas imágenes

```
# Aplicar las transformaciones
    valor_aclarado = 150  # Ejemplo de valor para aclarar u oscurecer
    valor_oscurecido = -100   # Ejemplo par oscurecer la imagen
    brillo_aumentado = 1.5      # Ejemplo de factor para aumentar el brillo
    brillo_disminuido = 0.2     # Ejemplo para disminuir el brillo
    Elongacion_contraste = 1.9  # Ejemplo de factor para aumentar el
contraste
    Reduccion_contraste = 0.3   # Ejemplo de factor de reducir contraste
```

Mostrar las imágenes, matrices y guardarlas

Para mostrar los resultados de las imágenes se usa tanto su matriz, imagen original y la imagen resultante, todas estas se muestran y se guardan en la misma carpeta donde se ejecuta el programa

```
# Mostrar las imágenes
    cv2.imshow('Imagen Original', imagen)
```

```
    # Se crea una variable para cada imagen para guardarla en los archivos
pero con la misma funcion para reutilizarla
    Imagen_aclarada = aclarar_oscurecer(imagen, valor_aclarado)
    # Se guarda la matriz resultante en un archivo .csv
    np.savetxt('imagen_aclarada.csv', Imagen_aclarada, delimiter=',',
fmt='%d')
    print("La matriz de la imagen aclarada se ha guardado en
'imagen_aclarada.csv'.")
    # Mostramos la imagen
    cv2.imshow('Imagen Aclarada', Imagen_aclarada)
    # Y por ultimo se guarda
    cv2.imwrite('Imagen Aclarada.png', Imagen_aclarada)
```

Cálculos y Resultados

```
import cv2
import numpy as np

# Cargar la imagen
imagen_a_color = cv2.imread("Incendio.png")

if imagen_a_color is None:
    print("Error: No se pudo cargar la imagen. Asegúrate de que el archivo
está en el directorio correcto.")
else:
    # Convertir la imagen en blanco y negro
    imagen = cv2.cvtColor(imagen_a_color, cv2.COLOR_BGR2GRAY)

    # NOTA: se convierte en blanco y negro despues de leerse porque sino
pierde pixeles

    # Función para aclarar u oscurecer la imagen
    def aclarar_oscurecer(imagen, valor):
        return cv2.convertScaleAbs(imagen, alpha=1, beta=valor)

    # Función para copiar la imagen
    def copiar_imagen(imagen):
        Imagen_copiada = imagen.copy()
        np.savetxt('imagen_copiada.csv', Imagen_copiada, delimiter=',',
fmt='%d')
        print("La matriz de la imagen copiada se ha guardado en
'imagen_copiada.csv'.")
        cv2.imwrite("Imagen copiada.png", Imagen_copiada)
        return Imagen_copiada
```

```
    # Función para invertir la imagen (negativo)
    def negativo_imagen(imagen):
        Imagen_Negativa = cv2.bitwise_not(imagen)
        np.savetxt('imagen_negativa.csv', Imagen_Negativa, delimiter=',',
fmt='%d')
        print("La matriz de la imagen negativa se ha guardado en
'imagen_negativa.csv'.")
        cv2.imwrite("Imagen Negativa.png", Imagen_Negativa)
        return Imagen_Negativa

    # Función para aumentar o disminuir el brillo
    def cambiar_brillo(imagen, factor):
        imagen = imagen.astype(np.float32)  # Convertir a float para evitar
overflow/underflow
        imagen = imagen * factor  # Multiplicar por el factor
        imagen = np.clip(imagen, 0, 255)  # Asegurar que los valores estén
en el rango 0-255
        return imagen.astype(np.uint8)  # Convertir de nuevo a uint8

    # Función para aumentar o reducir el contraste
    def cambiar_contraste(imagen, factor):
        imagen = imagen.astype(np.float32)  # Convertir a float para evitar
overflow/underflow
        media = np.mean(imagen)  # Calcular la media de los píxeles
        imagen = (1 - factor) * media + factor * imagen  # Ajustar el
contraste
        imagen = np.clip(imagen, 0, 255)  # Asegurar que los valores estén
en el rango 0-255
        return imagen.astype(np.uint8)  # Convertir de nuevo a uint8

    # Shifting de izquierda a derecha
    def Shifting_1(imagen):
        shifting_izquierda_a_derecha = imagen.copy()

        #Recorrer el arreglo de la imagen en escala de grises
        for i in range(imagen.shape[0]):
            for j in range(imagen.shape[1]):
                if j != 0:
                    shifting_izquierda_a_derecha[i][j] =
shifting_izquierda_a_derecha[i][j-1]/ (imagen[i][j-1]/imagen[i][j])

        np.savetxt('Shifting_izq_a_der.csv', shifting_izquierda_a_derecha,
delimiter=',', fmt='%d')
        print("La matriz de la imagen Shifting izquierda a derecha se ha
guardado en 'Shifting_izq_a_der.csv'.")
```

```
        cv2.imwrite("shifting izquierda a derecha.png",
shifting_izquierda_a_derecha)
        return shifting_izquierda_a_derecha

    # Shifting de derecha a izquierda
    def Shifting_2(imagen):
        shifting_dercha_a_izquierda = imagen.copy()

        # Recorrer el arreglo de la imagen en escala de grises de derecha a
izquierda
        for i in range(imagen.shape[0]):
            for j in range(imagen.shape[1] - 2, -1, -1):  # Inicia desde la
penúltima columna y va hacia la izquierda
                shifting_dercha_a_izquierda[i][j] =
shifting_dercha_a_izquierda[i][j+1] / (imagen[i][j+1] / imagen[i][j])

        np.savetxt('Shifting_der_a_izq.csv', shifting_dercha_a_izquierda,
delimiter=',', fmt='%d')

        np.savetxt('Shifting_der_a_izq.csv', shifting_dercha_a_izquierda,
delimiter=',', fmt='%d')
        print("La matriz de la imagen Shifting derecha a izquierda se ha
guardado en 'Shifting_der_a_izq.csv'.")

        cv2.imwrite("shifting derecha a izquierda.png",
shifting_dercha_a_izquierda)
        return shifting_dercha_a_izquierda

    # Shifting de arriba a abajo
    def Shifting_3(imagen):
        shifting_arriba_a_abajo = imagen.copy()

        # Recorrer el arreglo de la imagen en escala de grises de arriba a
abajo
        for i in range(1, imagen.shape[0]):  # Inicia desde la segunda fila
hacia abajo
            for j in range(imagen.shape[1]):  # Recorre todas las columnas
                shifting_arriba_a_abajo[i][j] = shifting_arriba_a_abajo[i-
1][j] / (imagen[i-1][j] / imagen[i][j])

        np.savetxt('Shifting_arr_a_aba.csv', shifting_arriba_a_abajo,
delimiter=',', fmt='%d')
        print("La matriz de la imagen Shifting arriba a abajo se ha guardado
en 'Shifting_arr_a_aba.csv'.")
```

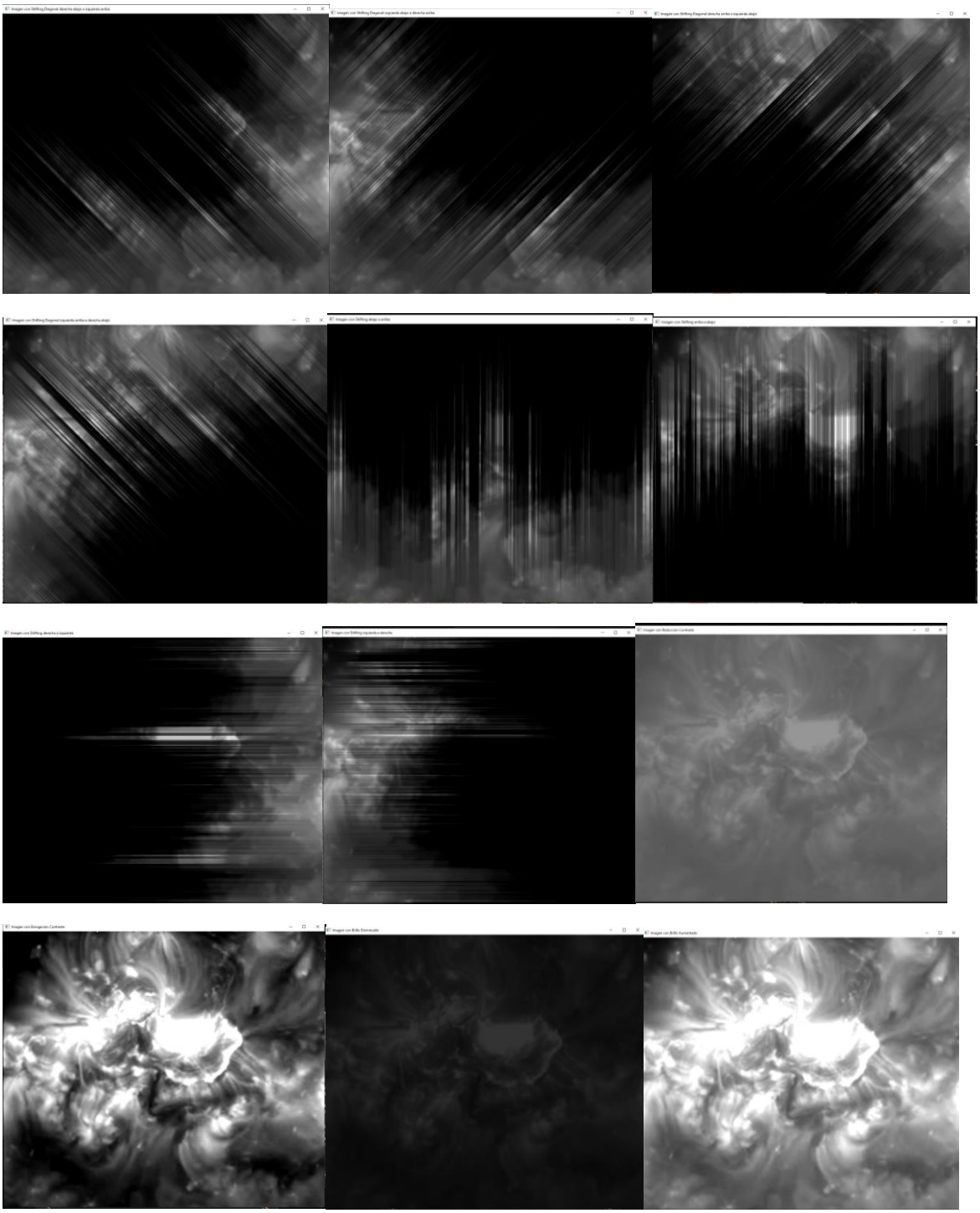
```
        cv2.imwrite("shifting arriba a abajo.png", shifting_arriba_a_abajo)
        return shifting_arriba_a_abajo

    def Shifting_4(imagen):
        shifting_abajo_a_arriba = imagen.copy()

        # Recorrer el arreglo de la imagen en escala de grises de abajo
hacia arriba
        for i in range(imagen.shape[0] - 2, -1, -1):  # Inicia desde la
penúltima fila y va hacia arriba
            for j in range(imagen.shape[1]):  # Recorre todas las columnas
                shifting_abajo_a_arriba[i][j] =
shifting_abajo_a_arriba[i+1][j] / (imagen[i+1][j] / imagen[i][j])

        # Guardar la matriz de la imagen Shifting en un archivo CSV
        np.savetxt('Shifting_aba_a_arr.csv', shifting_abajo_a_arriba,
delimiter=',', fmt='%d')
        print("La matriz de la imagen Shifting abajo a Arriba se ha guardado
en 'Original_Shifting_aba_a_arr.csv'.")

        cv2.imwrite("shifting abajo a arriba.png", shifting_abajo_a_arriba)
        return shifting_abajo_a_arriba

    def Shifting_5(imagen):
        Shifting_diagonal_1 = imagen.copy()

        # Recorrer la imagen en diagonales desde la esquina superior
izquierda a la inferior derecha
        rows, cols = imagen.shape
        for diag in range(rows + cols - 1):  # Esto recorre todas las
diagonales posibles
            # La fila inicial es el mínimo entre la diagonal y el número de
filas - 1
            # La columna inicial es la diferencia entre la diagonal y la
fila
            for i in range(max(0, diag - cols + 1), min(rows, diag + 1)):
                j = diag - i
                if i > 0 and j > 0:  # Evita el borde
                    Shifting_diagonal_1[i][j] = Shifting_diagonal_1[i-1][j-
1] / (imagen[i-1][j-1] / imagen[i][j])

        # Guardar la matriz de la imagen Shifting en un archivo CSV
        np.savetxt('Shifting_diag_izq_arr_a_der_aba.csv',
Shifting_diagonal_1, delimiter=',', fmt='%d')
```

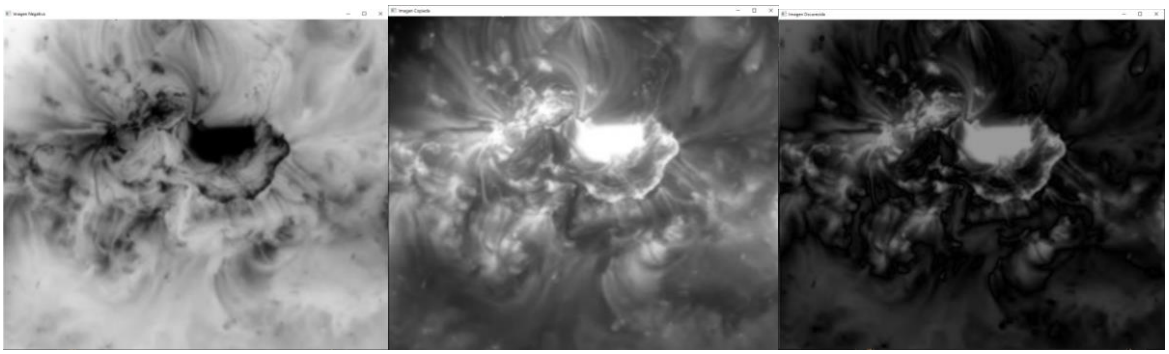
```
        print("La matriz de la imagen Shifting diagonal izquierda arriba a
derecha abajo se ha guardado en 'Shifting_diag_izq_arr_a_der_aba.csv'.")

        cv2.imwrite("shifting diagonal izquierda arriba a derecha
abajo.png", Shifting_diagonal_1)
        return Shifting_diagonal_1

    def Shifting_6(imagen):
        Shifting_diagonal_2 = imagen.copy()

        rows, cols = imagen.shape
        # Recorrer la imagen en diagonales desde la esquina superior derecha
a la inferior izquierda
        for diag in range(-cols + 1, rows):  # Esto recorre todas las
diagonales posibles
            for i in range(max(0, diag), min(rows, cols + diag)):  # Limitar
el recorrido a los bordes de la imagen
                j = cols - 1 - (i - diag)  # Columna que depende de la fila
y la diagonal
                if i > 0 and j < cols - 1:  # Evitar el borde
                    Shifting_diagonal_2[i][j] = Shifting_diagonal_2[i-
1][j+1] / (imagen[i-1][j+1] / imagen[i][j])

        # Guardar la matriz de la imagen Shifting en un archivo CSV
        np.savetxt('Shifting_diag_der_arr_a_izq_aba.csv',
Shifting_diagonal_2, delimiter=',', fmt='%d')
        print("La matriz de la imagen Shifting Diagonal derecha arriba a
izquierda abajo se ha guardado en 'Shifting_diag_der_arr_a_izq_aba.csv'.")

        cv2.imwrite("shifting diagonal derecha arriba a izquierda
abajo.png", Shifting_diagonal_2)
        return Shifting_diagonal_2

    def Shifting_7(imagen):
        Shifting_diagonal_3 = imagen.copy()

        rows, cols = imagen.shape
        # Recorrer la imagen en diagonales desde la esquina inferior
izquierda a la superior derecha
        for diag in range(rows + cols - 1):  # Recorrer todas las diagonales
posibles
            for i in range(min(diag, rows - 1), max(-1, diag - cols), -
1):  # Limitar el recorrido dentro de los bordes
                j = diag - i  # Columna correspondiente a la fila
                if i < rows - 1 and j > 0:  # Evitar los bordes
```

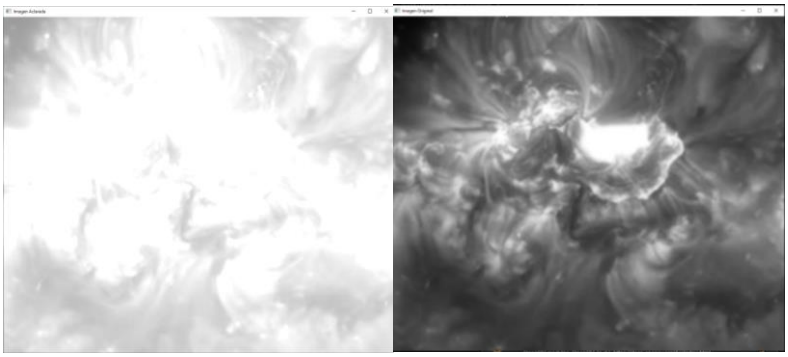
```
                    Shifting_diagonal_3[i][j] = Shifting_diagonal_3[i+1][j-
1] / (imagen[i+1][j-1] / imagen[i][j])

        # Guardar la matriz de la imagen Shifting en un archivo CSV
        np.savetxt('Shifting_diag_izq_aba_a_der_arr.csv',
Shifting_diagonal_3, delimiter=',', fmt='%d')
        print("La matriz de la imagen Shifting Diagonal izquierda abajo a
derecha arriba se ha guardado en 'Shifting_diag_izq_aba_a_der_arr.csv'.")

        cv2.imwrite("shifting diagonal izquierda abajo a derecha
arriba.png", Shifting_diagonal_3)
        return Shifting_diagonal_3

    def Shifting_8(imagen):
        Shifting_diagonal_4 = imagen.copy()

        rows, cols = imagen.shape
        # Recorrer la imagen en diagonales desde la esquina inferior derecha
a la superior izquierda
        for diag in range(rows + cols - 1):  # Recorrer todas las diagonales
posibles
            for i in range(min(rows - 1, diag), max(-1, diag - cols), -
1):  # Limitar el recorrido dentro de los bordes
                j = cols - 1 - (diag - i)  # Columna correspondiente a la
fila
                if i < rows - 1 and j < cols - 1:  # Evitar los bordes
                    Shifting_diagonal_4[i][j] =
Shifting_diagonal_4[i+1][j+1] / (imagen[i+1][j+1] / imagen[i][j])

        # Guardar la matriz de la imagen Shifting en un archivo CSV
        np.savetxt('Shifting_diag_der_aba_a_izq_arr.csv',
Shifting_diagonal_4, delimiter=',', fmt='%d')
        print("La matriz de la imagen Shifting Diagonal derecha abajo a
izquierda arriba se ha guardado en 'Shifting_diag_der_aba_a_izq_arr.csv'.")

        cv2.imwrite("shifting diagonal derecha abajo a izquierda
arriba.png", Shifting_diagonal_4)
        return Shifting_diagonal_4

    # Aplicar las transformaciones
    valor_aclarado = 150  # Ejemplo de valor para aclarar u oscurecer
    valor_oscurecido = -100   # Ejemplo par oscurecer la imagen
    brillo_aumentado = 1.5      # Ejemplo de factor para aumentar el brillo
    brillo_disminuido = 0.2     # Ejemplo para disminuir el brillo
```

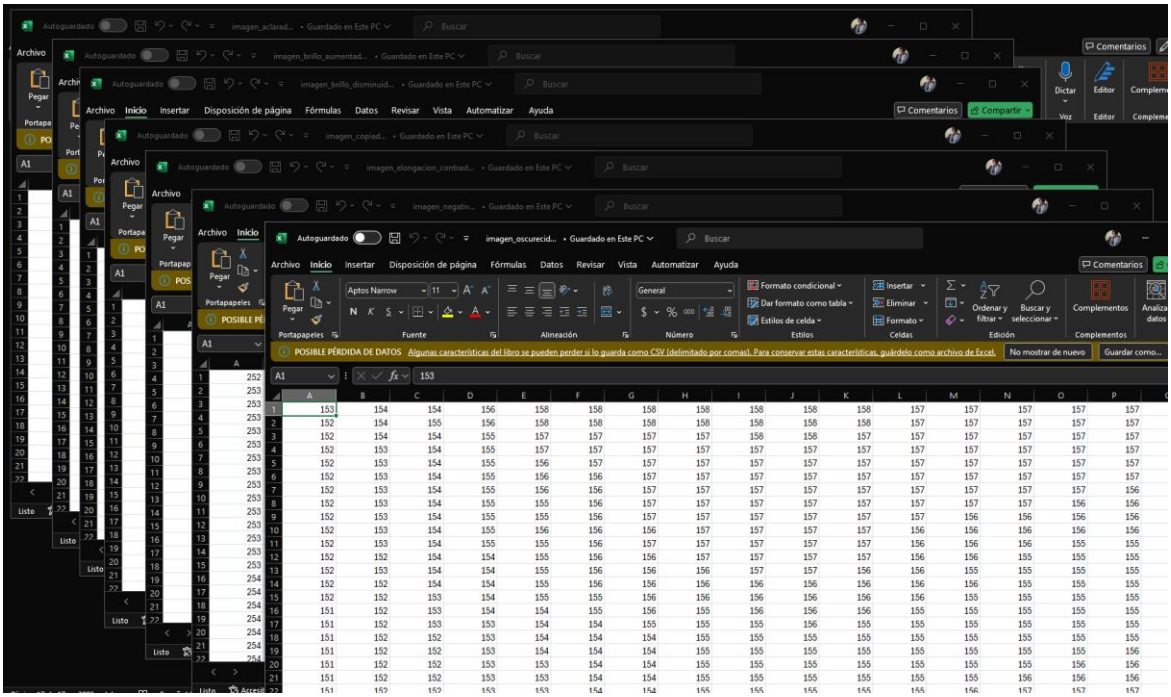
```
    Elongacion_contraste = 1.9  # Ejemplo de factor para aumentar el
contraste
    Reduccion_contraste = 0.3   # Ejemplo de factor de reducir contraste

    # Mostrar las imágenes
    cv2.imshow('Imagen Original', imagen)

    # Se crea una variable para cada imagen para guardarla en los archivos
pero con la misma funcion para reutilizarla
    Imagen_aclarada = aclarar_oscurecer(imagen, valor_aclarado)
    # Se guarda la matriz resultante en un archivo .csv
    np.savetxt('imagen_aclarada.csv', Imagen_aclarada, delimiter=',',
fmt='%d')
    print("La matriz de la imagen aclarada se ha guardado en
'imagen_aclarada.csv'.")
    # Mostramos la imagen
    cv2.imshow('Imagen Aclarada', Imagen_aclarada)
    # Y por ultimo se guarda
    cv2.imwrite('Imagen Aclarada.png', Imagen_aclarada)

    # Seguimos los mismos pasos para las demas imagenes

    Imagen_oscurecida = aclarar_oscurecer(imagen, valor_oscurecido)
    np.savetxt('imagen_oscurecida.csv', Imagen_aclarada, delimiter=',',
fmt='%d')
    print("La matriz de la imagen oscurecida se ha guardado en
'imagen_oscurecida.csv'.")
    cv2.imshow('Imagen Oscurecida', Imagen_oscurecida)
    cv2.imwrite('Imagen Oscurecida.png', Imagen_oscurecida)

    cv2.imshow('Imagen Copiada', copiar_imagen(imagen))

    cv2.imshow('Imagen Negativa', negativo_imagen(imagen))

    Imagen_brillo_aumentado = cambiar_brillo(imagen, brillo_aumentado)
    np.savetxt('imagen_brillo_aumentado.csv', Imagen_brillo_aumentado,
delimiter=',', fmt='%d')
    print("La matriz de la imagen brillo aumentado se ha guardado en
'imagen_brillo_aumentado.csv'.")
    cv2.imshow('Imagen con Brillo Aumentado', Imagen_brillo_aumentado)
    cv2.imwrite('Imagen con Brillo Aumentado.png', Imagen_brillo_aumentado)

    Imagen_brillo_disminuido = cambiar_brillo(imagen, brillo_disminuido)
    np.savetxt('imagen_brillo_disminuido.csv', Imagen_brillo_disminuido,
delimiter=',', fmt='%d')
```

```
    print("La matriz de la imagen brillo disminuido se ha guardado en
'imagen_brillo_disminuido.csv'.")
    cv2.imshow('Imagen con Brillo Disminuido', Imagen_brillo_disminuido)
    cv2.imwrite('Imagen con Brillo Disminuido.png',
Imagen_brillo_disminuido)

    Imagen_elongacion_contraste = cambiar_contraste(imagen,
Elongacion_contraste)
    np.savetxt('imagen_elongacion_contraste.csv',
Imagen_elongacion_contraste, delimiter=',', fmt='%d')
    print("La matriz de la imagen elongacion contraste se ha guardado en
'imagen_elongacion_contraste.csv'.")
    cv2.imshow('Imagen con Elongacion Contraste',
Imagen_elongacion_contraste)
    cv2.imwrite('Imagen con Elongacion Contraste.png',
Imagen_elongacion_contraste)

    Imagen_reduccion_contraste = cambiar_contraste(imagen,
Reduccion_contraste)
    np.savetxt('imagen_reduccion_contraste.csv', Imagen_reduccion_contraste,
delimiter=',', fmt='%d')
    print("La matriz de la imagen reduccion contraste se ha guardado en
'imagen_reduccion_contraste.csv'.")
    cv2.imshow('Imagen con Reduccion Contraste', Imagen_reduccion_contraste)
    cv2.imwrite('Imagen con Reduccion Contraste.png',
Imagen_reduccion_contraste)

    cv2.imshow("Imagen con Shifting izquierda a derecha",Shifting_1(imagen))
    cv2.imshow("Imagen con Shifting derecha a izquierda",Shifting_2(imagen))
    cv2.imshow("Imagen con Shifting arriba a abajo",Shifting_3(imagen))
    cv2.imshow("Imagen con Shifting abajo a arriba",Shifting_4(imagen))
    cv2.imshow("Imagen con Shifting Diagonal izquierda arriba a derecha
abajo",Shifting_5(imagen))
    cv2.imshow("Imagen con Shifting Diagonal derecha arriba a izquierda
abajo",Shifting_6(imagen))
    cv2.imshow("Imagen con Shifting Diagonal izquierda abajo a derecha
arriba",Shifting_7(imagen))
    cv2.imshow("Imagen con Shifting Diagonal derecha abajo a izquierda
arriba",Shifting_8(imagen))

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









Conclusiones

El programa demuestra cómo utilizar funciones de OpenCV para manipular imágenes mediante operadores básicos, como el ajuste de brillo y contraste, la creación de imágenes negativas, y el desplazamiento de la luz en la imagen (shifting). Estos conceptos son esenciales en la visión computacional y tienen

aplicaciones en áreas como el análisis de imágenes, la robótica y el procesamiento de imágenes en tiempo real.

Además, los ejercicios ayudan a fortalecer los conocimientos del tratado de imágenes y su posterior aplicación a diferentes detecciones de objetos con diferentes entornos que no este siendo propicio el ambiente para detectarlos, aplicando estas mismas técnicas para poder realizar su detección apropiada

Bibliografía

OpenCV modules. OpenCV. (n.d.). https://docs.opencv.org/4.x/index.html

NumPy Org. (2024, June 17). NumPy. https://numpy.org/

¿Qué es la visión artificial?. IBM. (2024, January 11). https://www.ibm.com/mx-es/topics/computer-vision