Unidad de aprendizaje: Álgebra Lineal

Reporte Técnico

Proyecto 3: Manipulación de Imágenes con Transformaciones Lineales

Autores:

García Juanillo Alan
García Juárez Dayan
Luna Campos Liam
Reyes León Angel Eduardo

Contenido

Introducción

El presente proyecto tiene como objetivo explorar y aplicar conceptos fundamentales del álgebra lineal en el ámbito del procesamiento digital de imágenes. Utilizando transformaciones lineales, se desarrolló un software capaz de realizar operaciones de rotación, escalado, reflexión y traslación sobre imágenes.

Estas operaciones están fundamentadas en la representación matricial de transformaciones geométricas en el plano, lo cual permite manipular las imágenes de manera eficiente y precisa. En este reporte, se aborda detalladamente el trasfondo matemático de las transformaciones, su implementación computacional y los resultados obtenidos.

Objetivo

Desarrollar un programa que manipule imágenes digitales mediante transformaciones lineales, aplicando rotaciones, escalados, reflexiones y traslaciones.

Marco Teórico

Transformaciones Lineales

En álgebra lineal, una transformación lineal es una función entre dos espacios vectoriales que preserva las operaciones de suma de vectores y multiplicación por escalar. Matemáticamente, una transformación lineal se define como:

Donde es una matriz que describe la transformación, y es un vector en el espacio original. Las transformaciones lineales son fundamentales en aplicaciones como gráficos computacionales, donde los puntos de una imagen se representan como vectores.

En términos prácticos, las coordenadas de cada píxel de una imagen se transforman mediante matrices que definen las operaciones deseadas, como rotación, escalado, reflexión y traslación.

Matrices de Transformación y Fundamentos Matemáticos

Rotación

La rotación implica girar un punto alrededor del origen. Este concepto se fundamenta en la trigonometría, utilizando las funciones seno y coseno para calcular las nuevas posiciones. La matriz que define esta transformación es:

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

El cálculo de las nuevas coordenadas (x', y') de un punto (x, y) se realiza mediante la multiplicación matricial:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = R(\theta) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

En la implementación del programa, el ajuste necesario surge porque las bibliotecas como OpenCV utilizan un sistema de coordenadas que considera los ángulos en sentido horario, mientras que las fórmulas matemáticas clásicas los calculan en sentido antihorario. Para resolverlo, se cambió el signo del seno en la matriz de rotación.

Escalado

El escalado modifica el tamaño de los objetos en el plano, ampliándolos o reduciéndolos de acuerdo con los factores s_x y s_y. Su matriz correspondiente es:

$$S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La transformación se realiza mediante:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = S \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

En el software, los factores de escalado son parámetros proporcionados por el usuario. Además, se asegura que el escalado se centre ajustando las coordenadas de la imagen antes y después de aplicar la transformación.

Reflexión

La reflexión invierte las coordenadas respecto a un eje específico. Para el eje horizontal y el eje vertical, las matrices son:

Reflexión horizontal:

$$F_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Reflexión vertical:

$$F_y = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Al aplicar estas matrices, las coordenadas se invierten en la dirección correspondiente:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = F \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

En el código, los ajustes incluyen la selección de la matriz adecuada en función del eje proporcionado por el usuario. Se garantiza la correcta inversión cambiando los signos de las coordenadas afectadas.

Traslación

La traslación desplaza un punto en el plano mediante los valores t_x y $t_y$lo cual se representa con la matriz:

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

El cálculo es directo:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = T \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

A diferencia de las otras transformaciones, esta operación no requiere ajustes adicionales para centrarse, ya que simplemente desplaza los píxeles en el espacio.

Desarrollo del Sistema

Herramientas Utilizadas

El programa fue desarrollado en Python utilizando:

NumPy: Para operaciones matriciales y algebraicas.

OpenCV: Para cargar, manipular y visualizar imágenes.

Adaptación Matemática al Código

Rotación

En el código, la rotación se implementa utilizando la función rotar, que calcula la matriz de rotación. Esto incluye convertir el ángulo dado en grados a radianes, ya que las funciones trigonométricas en Python (como cos y sin) utilizan radianes:

```
radianes = np.radians(angulo)

cos_theta, sin_theta = np.cos(radianes), np.sin(radianes)

matriz_rotacion = np.array([

    [cos_theta, sin_theta, 0],

    [-sin_theta, cos_theta, 0],

    [0, 0, 1]

])
```

np.radians(angulo): Convierte el ángulo de grados a radianes, ya que las funciones trigonométricas de NumPy esperan radianes como entrada.

np.cos y np.sin: Calculan los valores del coseno y el seno del ángulo dado, necesarios para llenar la matriz de rotación.

Para garantizar que la rotación ocurra alrededor del centro de la imagen en lugar del origen (0,0), se aplican traslaciones antes y después de la rotación:

```
matriz_traslado_centro = np.array([

    [1, 0, -centro_x],

    [0, 1, -centro_y],

    [0, 0, 1]

])
```

```
matriz_traslado_origen = np.array([
    [1, 0, centro_x],
    [0, 1, centro_y],
    [0, 0, 1]
])
matriz_final = np.dot(np.dot(matriz_traslado_origen, matriz_rotacion), matriz_traslado_centro)
```

El ajuste del signo de sin_theta en la matriz de rotación es crucial debido a la convención de OpenCV, que mide los ángulos en sentido horario, en contraste con el sentido antihorario de las fórmulas matemáticas clásicas. Por ello, se establece:

```
matriz_rotacion = np.array([
    [cos_theta, sin_theta, 0],
    [-sin_theta, cos_theta, 0],
    [0, 0, 1]
])
```

Este cambio asegura que la rotación visual sea consistente con las expectativas del usuario.

Escalado

El escalado utiliza la matriz:

```
matriz_escalado = np.array([
    [factor_x, 0, 0],
    [0, factor_y, 0],
    [0, 0, 1]
])
```

Los factores factor_x y factor_y son parámetros proporcionados por el usuario para ajustar el tamaño de la imagen. Se aplican traslaciones al centro de manera similar a la rotación para garantizar que el escalado ocurra respecto al centro de la imagen y no desde el origen (0,0):

```
matriz_traslado_centro = np.array([
```

```
    [1, 0, -centro_x],
    [0, 1, -centro_y],
    [0, 0, 1]
])
matriz_traslado_origen = np.array([
    [1, 0, centro_x],
    [0, 1, centro_y],
    [0, 0, 1]
])
matriz_final = np.dot(np.dot(matriz_traslado_origen, matriz_escalado), matriz_traslado_centro)
```

Reflexión

La función reflejar selecciona la matriz de reflexión en función del eje especificado por el usuario. Estas matrices son:

Reflexión horizontal:

```
matriz_reflexion = np.array([
    [1, 0, 0],
    [0, -1, 0],
    [0, 0, 1]
])
```

Reflexión vertical:

```
matriz_reflexion = np.array([
    [-1, 0, 0],
    [0, 1, 0],
    [0, 0, 1]
])
```

El sistema de coordenadas de OpenCV tiene el origen en la esquina superior izquierda, lo que significa que el eje aumenta hacia abajo. Este diseño implica que las inversiones deben respetar estas convenciones para obtener el reflejo correcto. Como en la rotación y el escalado, se aplican traslaciones para centrar la reflexión:

```
matriz_traslado_centro = np.array([
    [1, 0, -centro_x],
    [0, 1, -centro_y],
    [0, 0, 1]
])
matriz_traslado_origen = np.array([
    [1, 0, centro_x],
    [0, 1, centro_y],
    [0, 0, 1]
])
matriz_final = np.dot(np.dot(matriz_traslado_origen, matriz_reflexion), matriz_traslado_centro)
```

Traslación

En el programa, la traslación se implementa utilizando la matriz:

```
matriz_traslacion = np.array([
    [1, 0, dx],
    [0, 1, dy],
    [0, 0, 1]
])
```

La traslación simplemente desplaza cada punto de la imagen en las direcciones y según los valores y . A diferencia de otras transformaciones, no requiere ajustes adicionales al centro, ya que solo implica un desplazamiento lineal.

Para garantizar que los nuevos píxeles se mantengan dentro de los límites de la imagen, se validan las posiciones calculadas antes de asignar los valores a la nueva matriz de píxeles:

```
if 0 <= x_nuevo < columnas and 0 <= y_nuevo < filas:
```

```
    imagen_transformada[y_nuevo, x_nuevo] = imagen[y_original, x_original]
```

Este enfoque asegura que no se produzcan errores al intentar acceder a píxeles fuera de los límites.

Procesamiento de Imágenes y Estructura del Código

Carga y Procesamiento de Imágenes

Carga de Imágenes

El programa utiliza la función cv2.imread de OpenCV para cargar las imágenes seleccionadas por el usuario. Estas imágenes se representan como matrices tridimensionales, donde las dimensiones corresponden a:

Alto: Número de filas (píxeles) en la imagen.

Ancho: Número de columnas (píxeles) en la imagen.

Canales: Valores de color (normalmente BGR: Azul, Verde, Rojo).

Por ejemplo, una imagen de 800x600 píxeles tendrá una representación como una matriz NumPy de dimensiones (600, 800, 3).

Conversión de Formato

Como OpenCV utiliza el formato de color BGR por defecto, mientras que otras bibliotecas como Matplotlib utilizan RGB, es necesario realizar una conversión antes de mostrar las imágenes. Esto se logra mediante la función

`cv2.cvtColor:`

`python`

```
imagen_rgb = cv2.cvtColor(imagen_bgr, cv2.COLOR_BGR2RGB)
```

Este paso garantiza que los colores se muestren correctamente al visualizar las imágenes transformadas y originales lado a lado.

Validación de Coordenadas

Una parte crítica del programa es garantizar que las nuevas coordenadas generadas después de aplicar una transformación permanezcan dentro de los límites de la imagen. Esto evita errores de indexación al intentar acceder a posiciones fuera del rango permitido.

El programa realiza esta validación en la función aplicar_transformacion:

python

```
if 0 <= x_nuevo < columnas and 0 <= y_nuevo < filas:
    imagen_transformada[y_nuevo, x_nuevo] = imagen[y_original, x_original]
```

Esta verificación asegura que solo los píxeles válidos se mapeen en la nueva imagen transformada.

Organización de Resultados

- Para mantener un orden claro y eficiente en los resultados, el programa implementa la creación de carpetas específicas para cada tipo de transformación:
- Carpeta Principal: processed es la carpeta base donde se almacenan todos los resultados generados.
- Subcarpetas por Transformación: Dentro de processed, se crean carpetas como rotar, escalar, reflejar y trasladar. Cada carpeta contiene las imágenes procesadas correspondientes a esa transformación.
- Además, los nombres originales de las imágenes se conservan, facilitando su identificación después de ser procesadas.

Interfaz Gráfica (GUI)

Herramientas Utilizadas

Se utilizó la biblioteca Tkinter para diseñar la interfaz gráfica, lo que proporciona una forma sencilla y eficiente de interactuar con el programa. Los componentes principales incluyen:

Botones (Button): Permiten cargar imágenes y seleccionar el tipo de transformación a aplicar.

Entradas de Texto (Entry): Facilitan el ingreso de parámetros específicos, como ángulos, factores de escala o desplazamientos.

Mensajes (messagebox): Notifican al usuario sobre el éxito de las operaciones, errores o instrucciones.

Organización Visual

La ventana principal de la interfaz incluye botones para cada transformación, dispuestos de manera lógica para facilitar su uso. Además, se utilizan ventanas emergentes (Toplevel) para solicitar parámetros específicos para cada operación.

Flujo de Interacción

Selección de Imágenes: El usuario selecciona las imágenes desde su computadora a través de un cuadro de diálogo proporcionado por filedialog.askopenfilenames. Este método permite cargar múltiples imágenes a la vez.

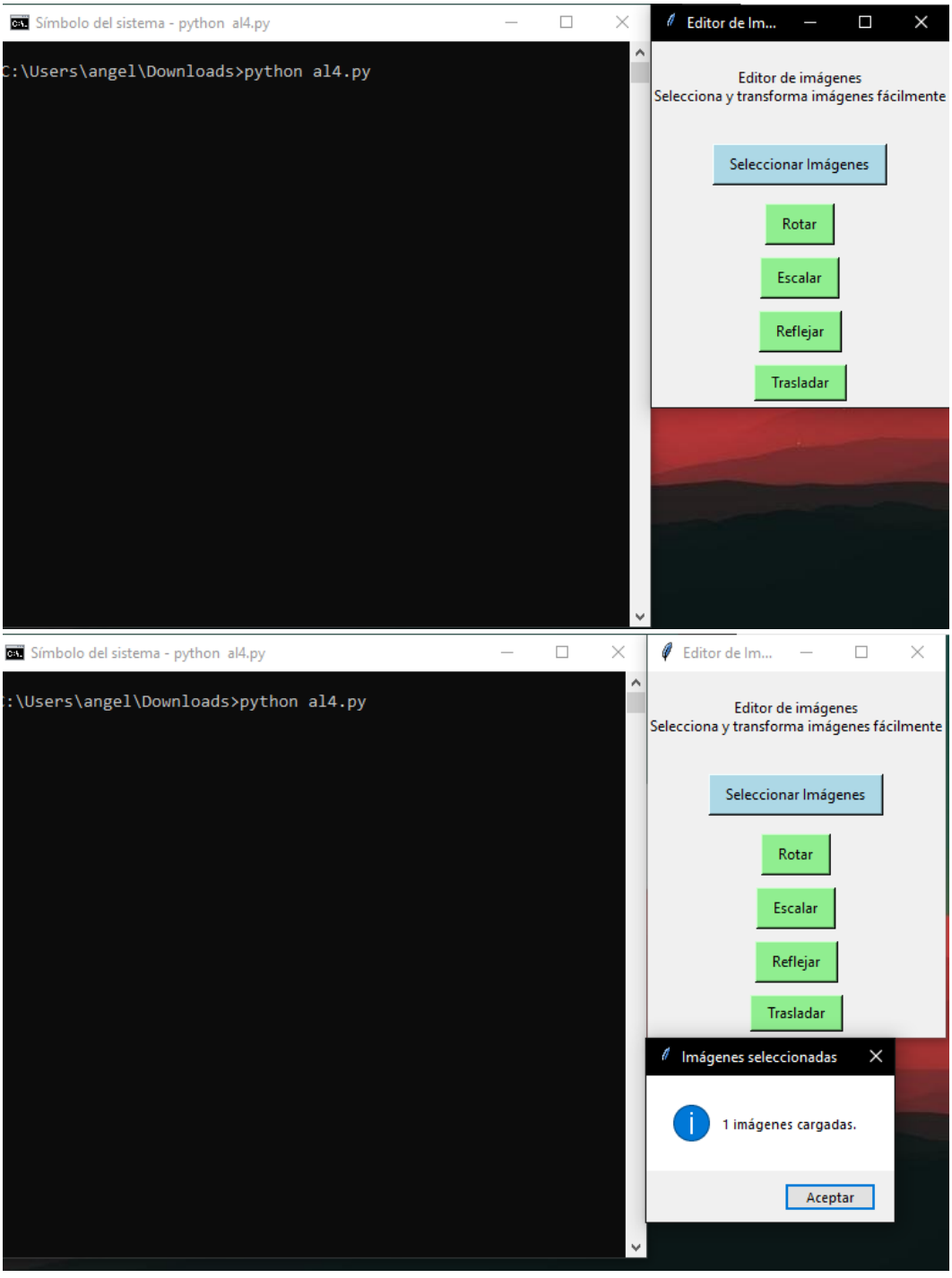
Selección de Transformación: Una vez cargadas las imágenes, el usuario elige una transformación (rotar, escalar, reflejar o trasladar) mediante botones dedicados. Dependiendo de la transformación seleccionada, se solicitan los parámetros correspondientes.

Visualización de Resultados: Los resultados de las transformaciones se muestran utilizando Matplotlib. Las imágenes originales y transformadas se presentan lado a lado para facilitar la comparación.

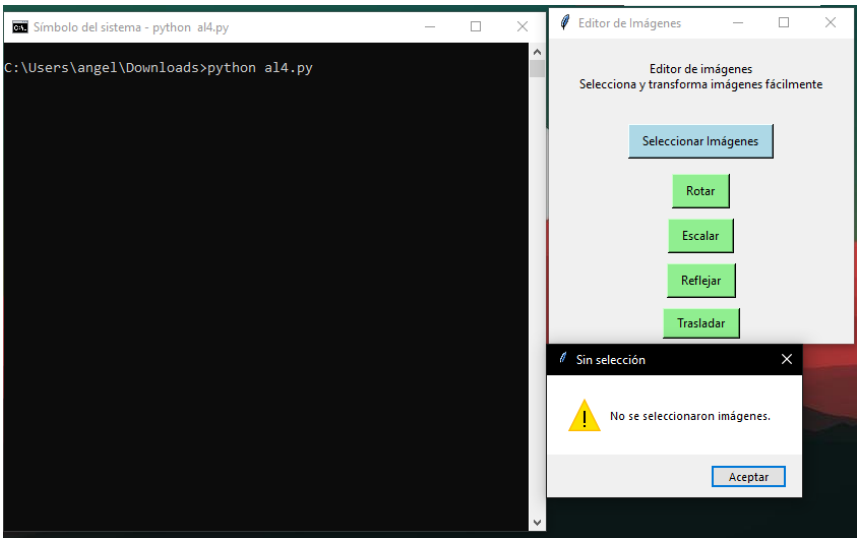
Guardado Automático: Las imágenes procesadas se guardan automáticamente en carpetas organizadas por tipo de transformación, manteniendo su nombre original.

Pruebas Realizadas

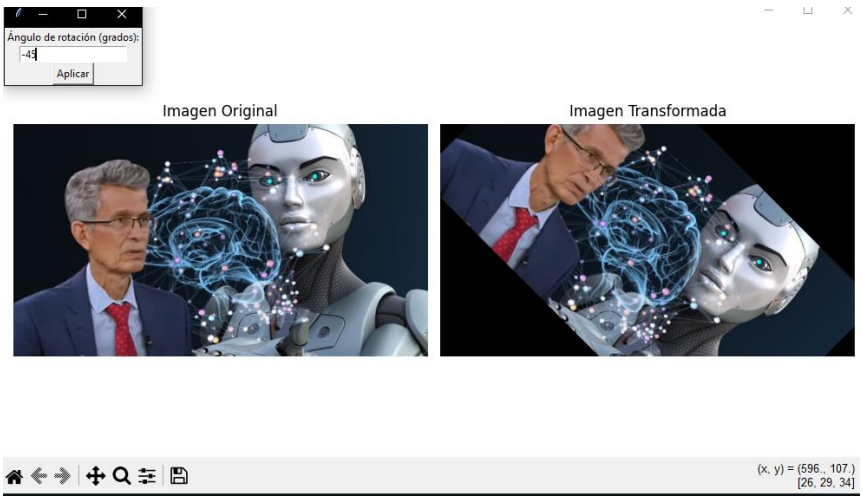
Apertura de programa y carga exitosa de imagen



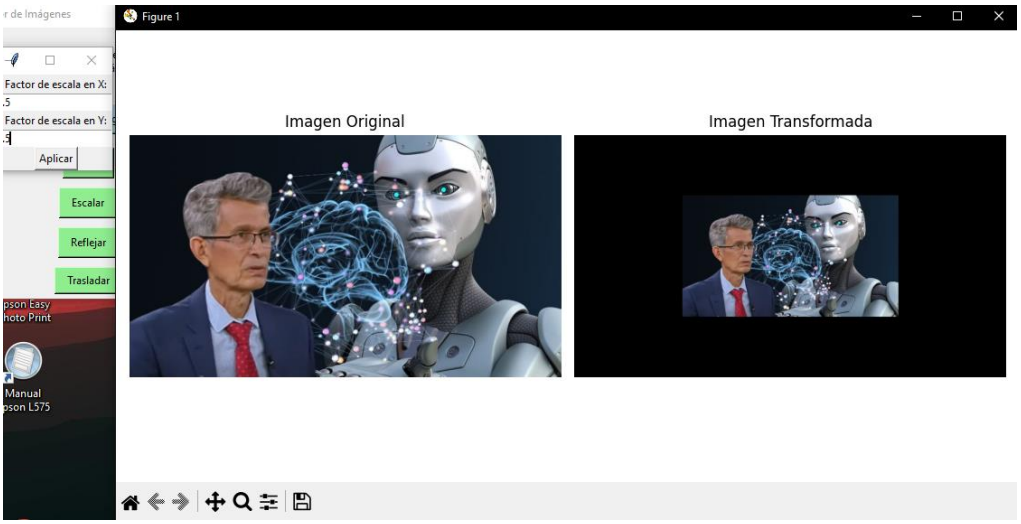
Mensaje de carga fallida de imagen



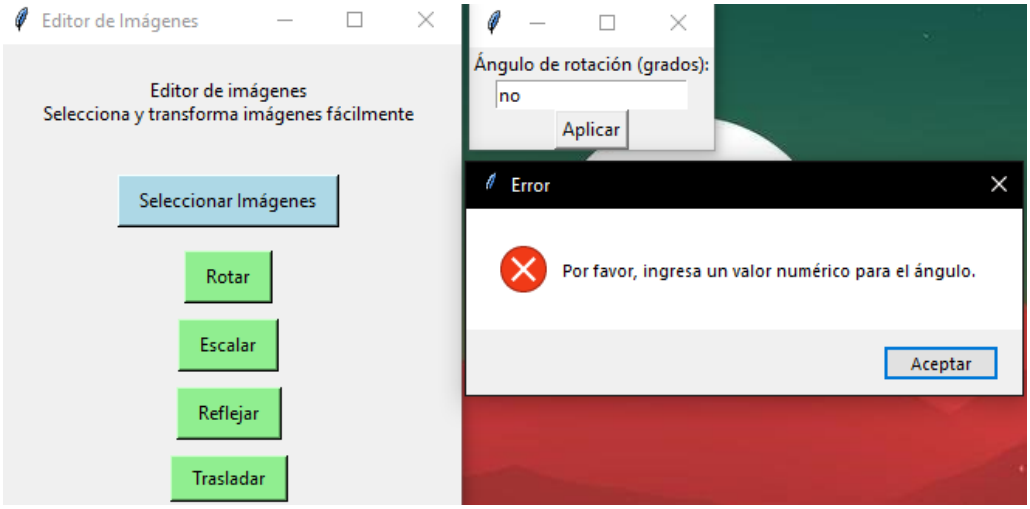
Rotación positiva



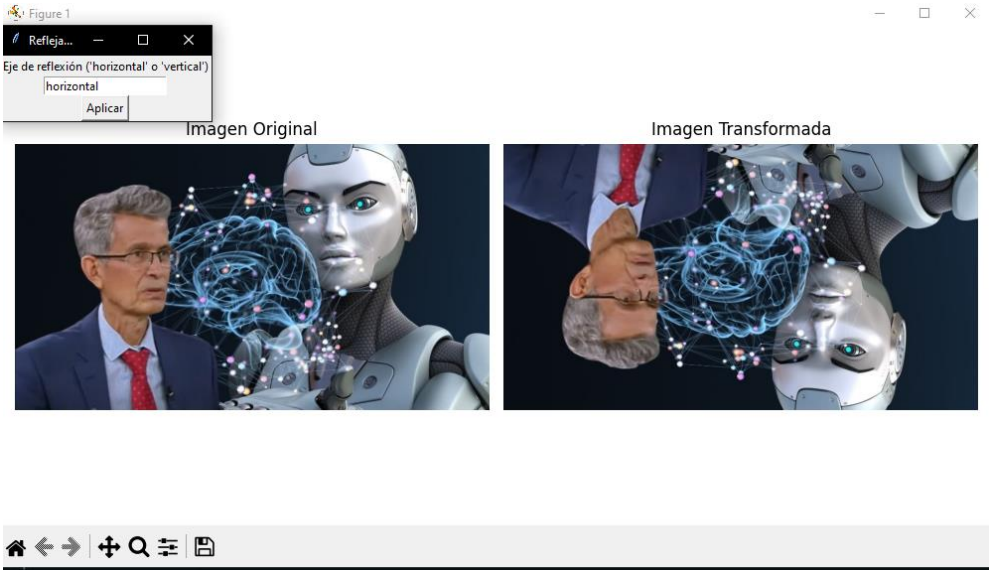
Rotación negativa



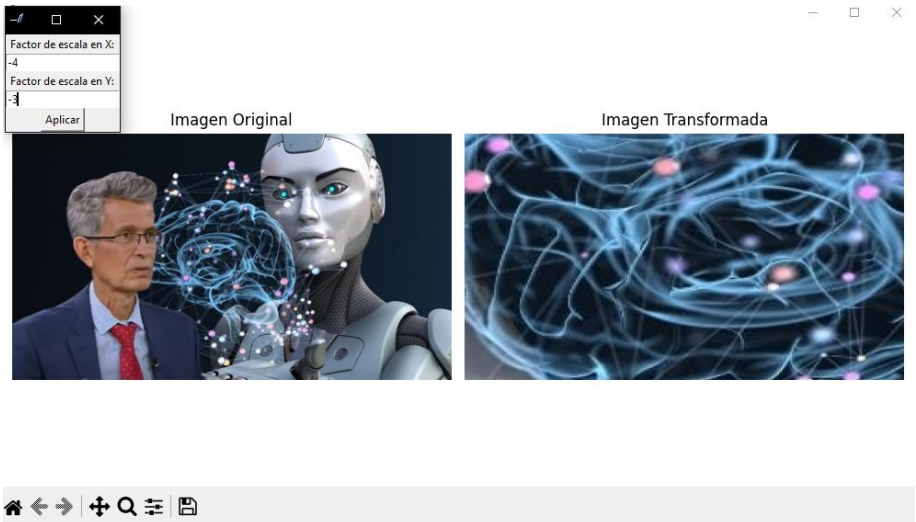
Mensaje valor inválido



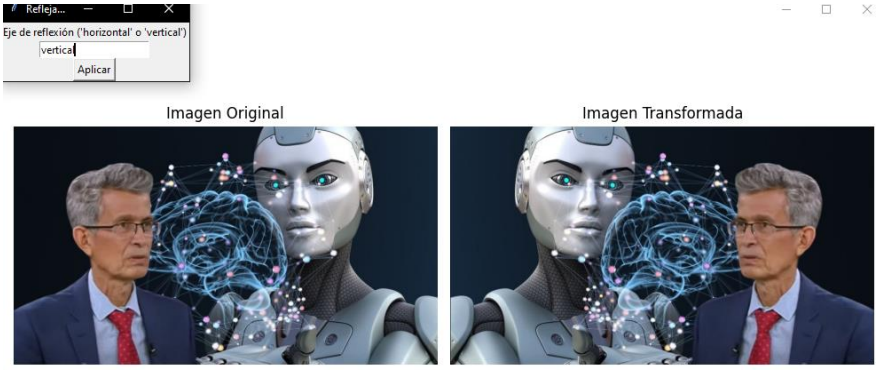
Escalado positivo



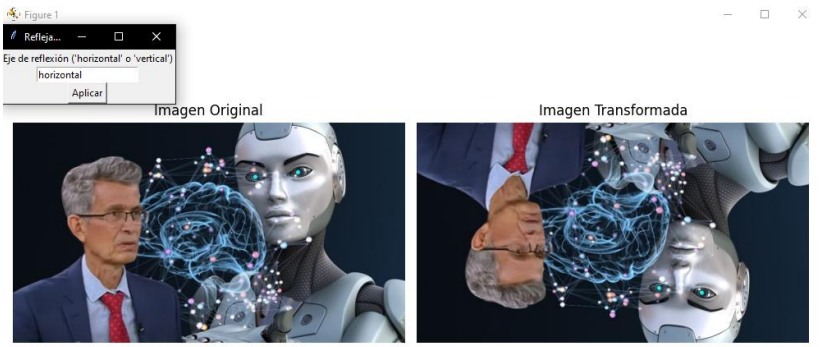
Escalado Negativo (invertido y aumentado)



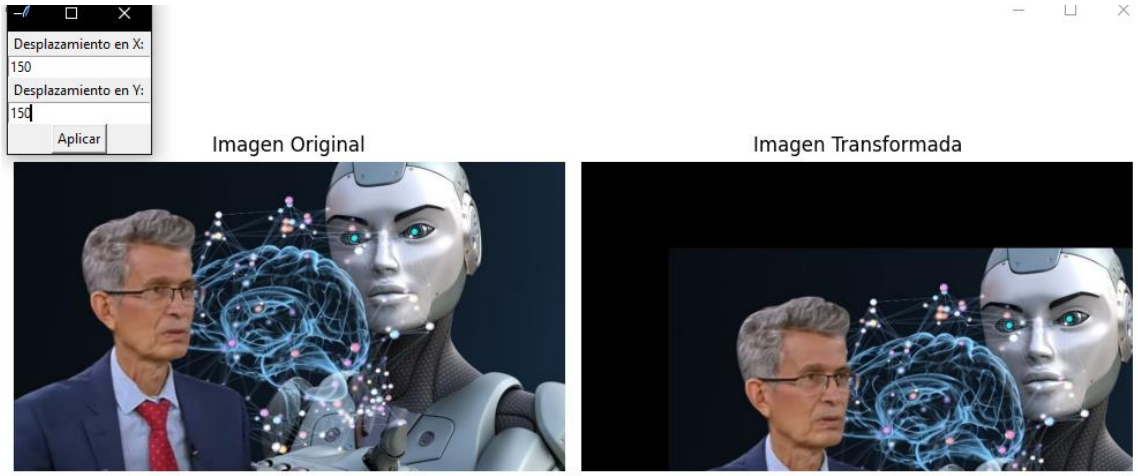
Reflexión Vertical



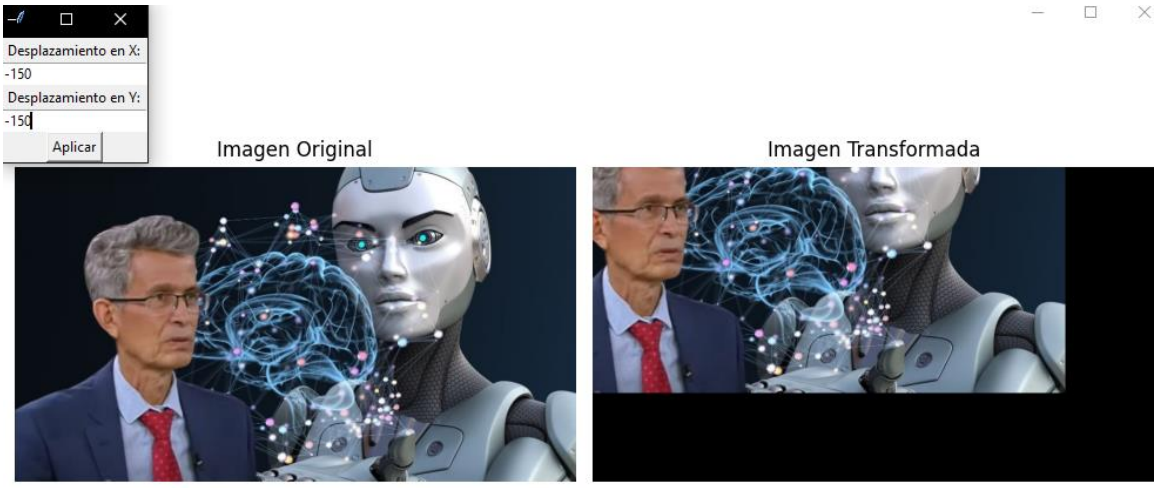
Reflexión Horizontal



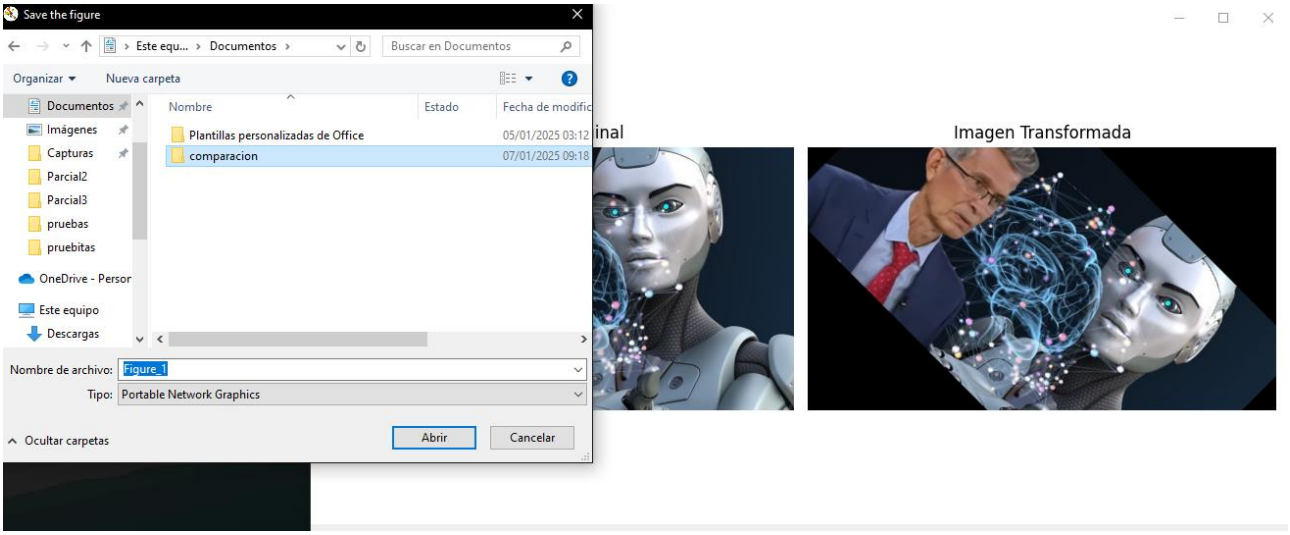
Traslación positiva

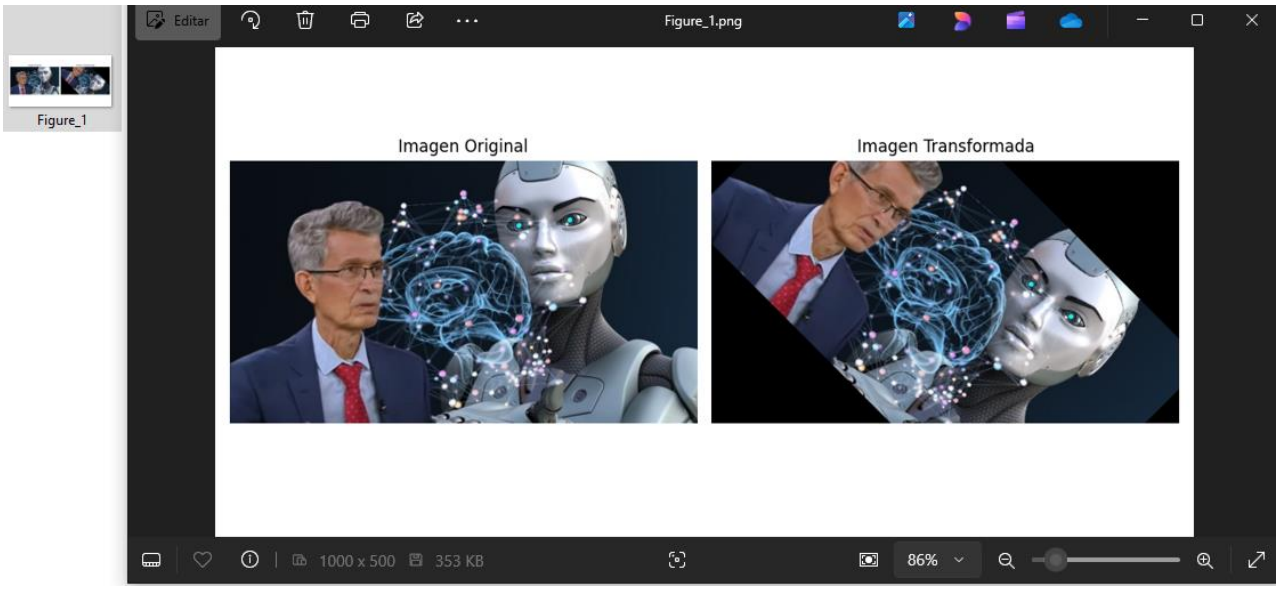


Traslación Negativa



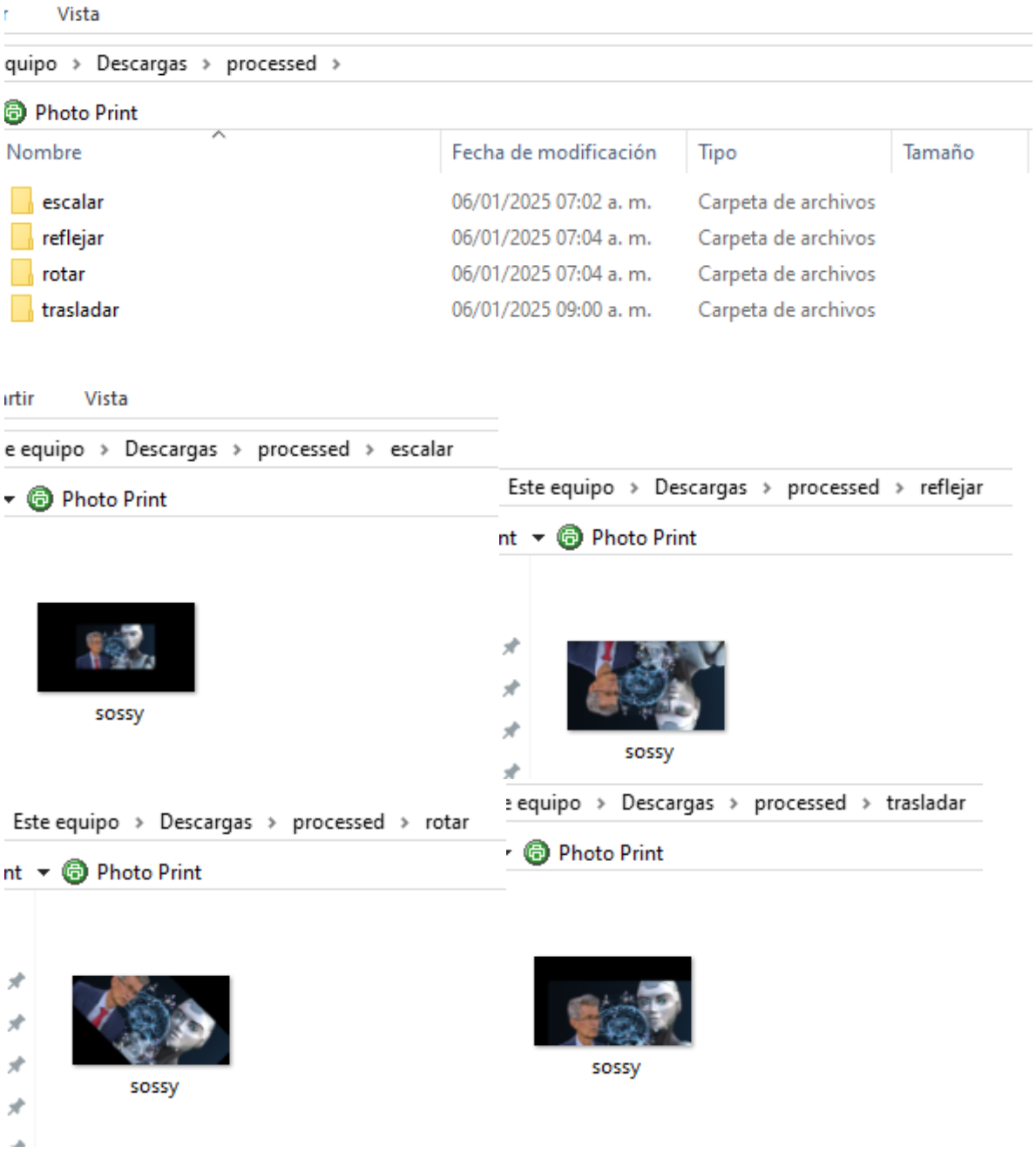
Guardado imagen comparativa





Guardado automático de transformaciones

```
PS C:\Users\angel\Downloads\Geoesp> & C:/Users/angel/Downloads/Geoesp/env/Scripts/python.exe c:/Users/angel/Downloads/al4.py
Procesada y guardada en: C:\Users\angel\Downloads\Geoesp\processed\escalar\sossy.jpg
Procesada y guardada en: C:\Users\angel\Downloads\Geoesp\processed\escalar\sossy.jpg
Procesada y guardada en: C:\Users\angel\Downloads\Geoesp\processed\escalar\sossy.jpg
Procesada y guardada en: C:\Users\angel\Downloads\Geoesp\processed\reflejar\sossy.jpg
Procesada y guardada en: C:\Users\angel\Downloads\Geoesp\processed\reflejar\sossy.jpg
Procesada y guardada en: C:\Users\angel\Downloads\Geoesp\processed\trasladar\sossy.jpg
Procesada y guardada en: C:\Users\angel\Downloads\Geoesp\processed\trasladar\sossy.jpg
```



Conclusiones

Aplicación del Álgebra Lineal en Procesamiento de Imágenes: Este proyecto demuestra cómo conceptos fundamentales del álgebra lineal, como las transformaciones lineales, se pueden aplicar de manera práctica para manipular imágenes digitales. Operaciones como rotación, escalado, reflexión y traslación se fundamentan en representaciones matriciales, lo que refuerza la conexión entre la teoría matemática y su uso en aplicaciones computacionales.

Importancia de las Adaptaciones Computacionales: Durante el desarrollo, se evidenció la necesidad de ajustar las fórmulas matemáticas para que fueran compatibles con las convenciones y restricciones de las bibliotecas utilizadas, como OpenCV. Esto incluyó cambios en el manejo de ángulos y ajustes al sistema de coordenadas.

Interacción Intuitiva con la Interfaz Gráfica: La integración de una GUI amigable mediante Tkinter permitió a los usuarios realizar transformaciones sin requerir conocimientos avanzados de programación. Esto resalta la importancia de combinar cálculos complejos con interfaces accesibles para garantizar un uso eficiente del software.

Flexibilidad y Modularidad del Sistema: La implementación de funciones específicas para cada transformación y el uso de matrices homogéneas hizo que el sistema fuera modular y fácil de expandir. Esto permite incorporar nuevas transformaciones o características en el futuro.

Validación y Gestión de Errores: El sistema incluyó validaciones robustas para garantizar que las coordenadas generadas permanecieran dentro de los límites de las imágenes, evitando errores de indexación. Esto demuestra la importancia de anticipar y manejar errores en el diseño de software.

Reforzamiento de los Conceptos Teóricos: La implementación de transformaciones geométricas permitió a los desarrolladores profundizar en los conceptos de álgebra lineal y entender mejor su utilidad en problemas reales, lo que enriquece el aprendizaje académico.

Organización Clara de Resultados: La creación de carpetas específicas para cada tipo de transformación facilitó la gestión de los resultados generados, ofreciendo un flujo de trabajo estructurado y eficiente.

En conclusión, este proyecto no solo logró cumplir sus objetivos técnicos, sino que también sirvió como una herramienta pedagógica para explorar y entender el álgebra lineal en un contexto práctico. Esto resalta el valor del aprendizaje aplicado en la formación académica y profesional.

Referencias:

Gonzalez, R. C., & Woods, R. E. (2018). **Digital Image Processing** (4th ed.). Pearson.
Páginas 127-129: Transformaciones geométricas básicas.

Hartley, R., & Zisserman, A. (2004). **Multiple View Geometry in Computer Vision** (2nd ed.). Cambridge University Press.
Páginas 27-30: Uso de matrices homogéneas para transformaciones geométricas.

OpenCV. (n.d.). Geometric Transformations of Images. OpenCV Documentation. Recuperado de https://docs.opencv.org
Sección: WarpAffine y WarpPerspective.

Szeliski, R. (2010). **Computer Vision: Algorithms and Applications**. Springer.
Páginas 54-56: Representación de transformaciones en imágenes digitales.

Foley, J. D., van Dam, A., Feiner, S. K., & Hughes, J. F. (1995). **Computer Graphics: Principles and Practice** (2nd ed.). Addison-Wesley.
Páginas 206-211: Transformaciones geométricas en gráficos por computadora.