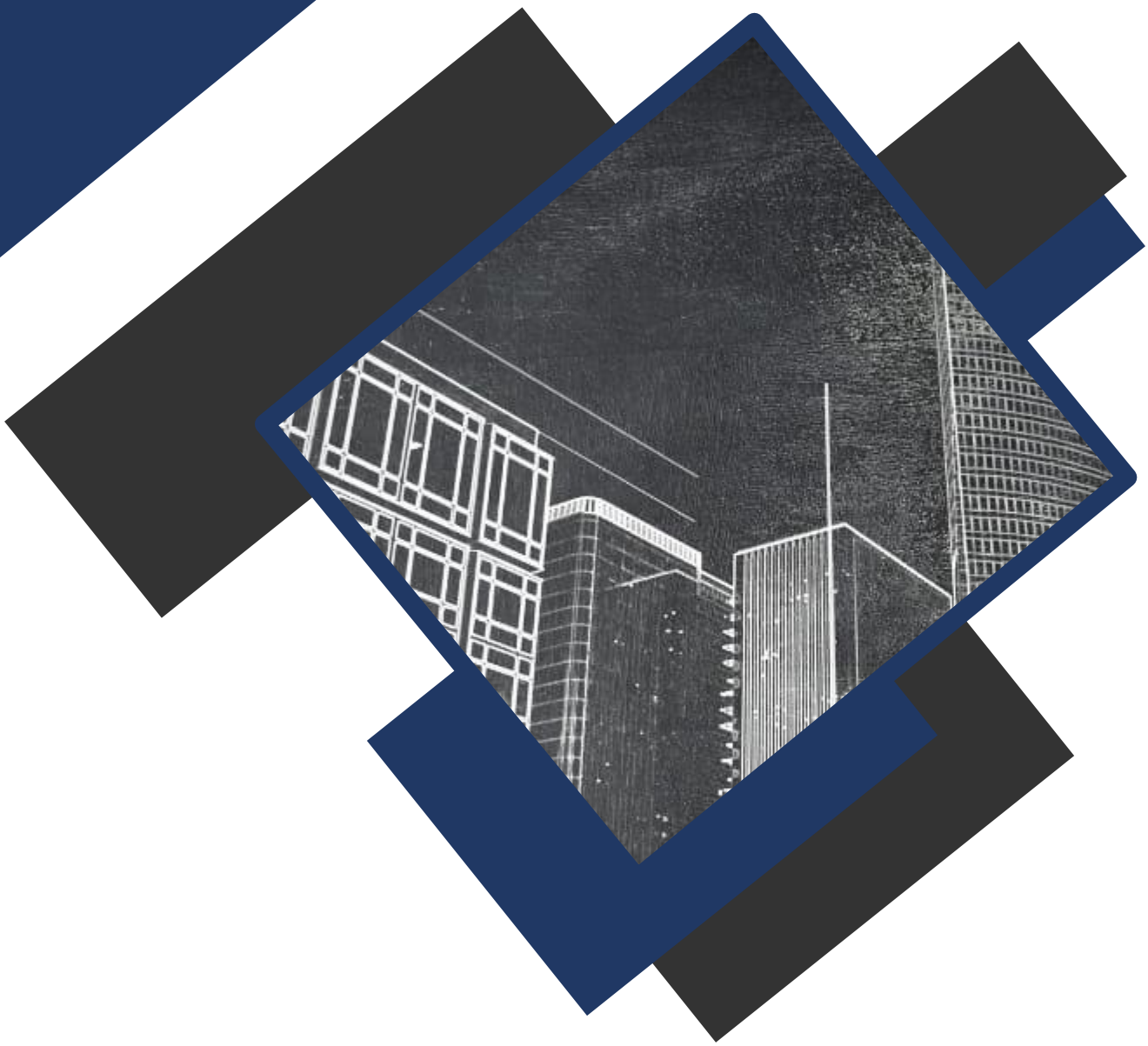


Reporte técnico

Manipulación de Imágenes con Transformaciones Lineales

Integrantes
Herrera Monroy Abraham Andre
Pérez Méndez Nancy Esmeralda
Martínez Berumen Juan Luis
Romero Rosales Lino Ehecatl

Objetivo del proyecto

Desarrollar un programa que manipule imágenes digitales mediante transformaciones lineales, aplicando rotaciones, escalados, reflexiones y traslaciones. Los estudiantes deberán utilizar matrices de transformación para modificar la imagen y visualizar los resultados de manera gráfica**.**

Descripción del Proyecto

El proyecto consiste en crear una herramienta que permita al usuario cargar una imagen, aplicar diversas transformaciones geométricas y observar los efectos resultantes. Los estudiantes deberán:

1. Cargar imágenes digitales:
    - Leer imágenes en formatos comunes (por ejemplo, JPEG o PNG) y representarlas como matrices de píxeles.

2. Aplicar transformaciones geométricas:
    - Rotación: Girar la imagen un número de grados especificado.
    - Escalado: Modificar el tamaño de la imagen, manteniendo o no las proporciones.
    - Reflexión: Invertir la imagen respecto a un eje.
    - Traslación: Mover la imagen en el plano sin alterar su orientación.

3. Visualizar resultados:
    - Mostrar la imagen original junto con la imagen transformada.

4. Exportar resultados (opcional):
    - Guardar la imagen transformada en un nuevo archivo.

Introducción teórica

Lenguaje de programación usado: Python última versión a la fecha enero 2025
Python es una buena opción para el procesamiento y manipulación de imágenes debido a su sintaxis sencilla y a la disponibilidad de numerosas bibliotecas.
Las bibliotecas que se utilizaron en el código fueron:

- **NumPy**: fundamental para trabajar con matrices y operaciones numéricas, que ayudaran para el procesamiento de imágenes.
- **OpenCV**: visión por computadora, ayuda para el procesamiento de las imágenes, como la lectura, escritura, manipulación, detección de objetos, etc.
- **Tkinter**: Esta biblioteca es para crear una interfaz de la aplicación.
- **Pillow**: proporciona herramientas con imágenes, como guarda, modificar, y mostrar imágenes.

Conceptos clave:
Las imágenes digitales se representan como matrices numéricas. Cada elemento de la matriz corresponde a un píxel y su valor representa la intensidad del color en ese punto.

Las matrices de transformación, como la rotación, se pueden representar como matrices de transformación. Al multiplicar esta matriz por la de las coordenades de un punto de la imagen, obtenemos las nuevas coordenadas del punto después de la transformación.

Cos (θ) -Sen (θ)
Sen (θ) Cos (θ)

- **cos(θ):** Representa la proyección de un vector unitario en el eje x después de una rotación de θ grados.
- **-sin(θ):** Representa la proyección de un vector unitario en el eje y después de una rotación de θ grados, pero con signo negativo debido a la dirección de la rotación.
- **sin(θ):** Representa la proyección de un vector unitario en el eje x después de una rotación de θ grados y 90 grados adicionales.
- **cos(θ):** Representa la proyección de un vector unitario en el eje y después de una rotación de θ grados y 90 grados adicionales.

Al multiplicar esta matriz por las coordenadas de un punto, estamos esencialmente rotando ese punto en un ángulo θ alrededor del origen.

Usos

- **Rotación:** La matriz de rotación gira una imagen un ángulo θ alrededor de un punto fijo
- **Escalado:** Cambia de tamaño la matriz de una imagen a lo largo de los ejes **X** e **Y.**
- **Reflexión:** Invierte la imagen a lo largo de un eje.
- **Traslación:** Desplaza la imagen una cierta distancia a lo largo de los ejes **X** e **Y.**

Desarrollo del sistema

Con ayuda de las librerías previamente mencionadas, nos ayudara a realizar el código para llevar a cabo cada una de las funciones que se solicitan.

Iniciamos con la función base del programa que es para cargar imágenes:

```
def cargar_imagen(ruta):
    """Carga una imagen desde la ruta especificada y la convierte a escala de grises."""
    imagen = cv2.imread(ruta, cv2.IMREAD_GRAYSCALE)
    if imagen is None:
        raise FileNotFoundError("No se pudo cargar la imagen. Verifica la ruta.")
    return imagen
```

La función utiliza la función **imread** que es para poder leer imágenes y cargarlas, parte de la librería OpenCv, además de pasarla completamente a una escala de grises, además de desplegar un mensaje cuando no se encontró o no se cargo una imagen.

Matriz de rotación:

```
def rotar_imagen(imagen, angulo):
    """Aplica una rotación a la imagen."""
    filas, columnas = imagen.shape
    matriz_rotacion = cv2.getRotationMatrix2D((columnas / 2, filas / 2), angulo, 1)
    return cv2.warpAffine(imagen, matriz_rotacion, (columnas, filas))
```

Se crea una matriz de rotación con la función **getRotationMatrix2D**, que es una función parte de la librería OpenCv, esta función solo nos pide el punto medio de la imagen, ángulo y su escala para crear su matriz de rotación.

Matriz de escalado:

```
def escalar_imagen(imagen, factor_x, factor_y):
    """Escala la imagen por los factores especificados."""
    if factor_x > 10 or factor_y > 10:
        raise ValueError("Los factores de escala deben ser menores o iguales a 10.")
    return cv2.resize(imagen, None, fx=factor_x, fy=factor_y, interpolation=cv2.INTER_LINEAR)
```

No se crea explícitamente una matriz, pues la función **resize** utiliza internamente una matriz de escala para poder calcular y transformar la imagen original a una nueva imagen y Re muestrear los pixeles.

Matriz de traslación

```
def trasladar_imagen(imagen, desplazamiento_x, desplazamiento_y):
    """Aplica una traslación a la imagen."""
    filas, columnas = imagen.shape
    matriz_traslacion = np.float32([[1, 0, desplazamiento_x], [0, 1, desplazamiento_y]])
    return cv2.warpAffine(imagen, matriz_traslacion, (columnas, filas))
```

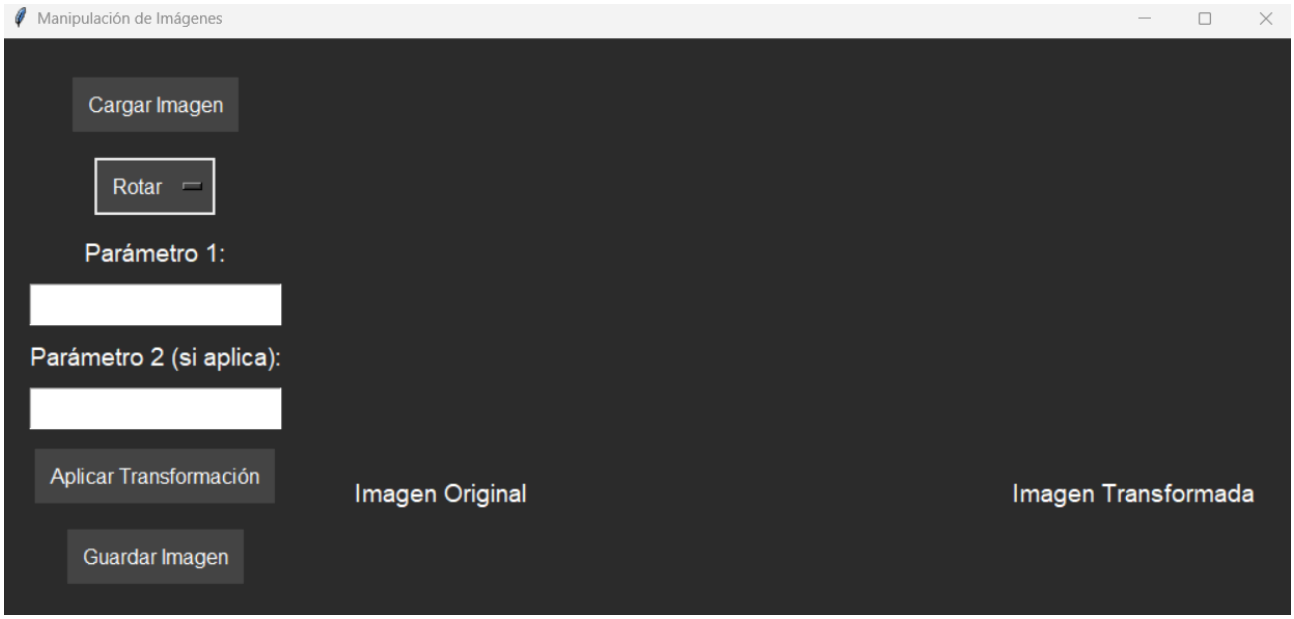
Creamos una matriz manualmente con ayuda de la librería NumPy y utilizando la función **warpAffine** que nos pide e ingresar los parámetros para trasladar la imagen, los elementos que pide son la imagen la matriz y los bordes de la imagen.

Matriz de reflexión

```python
def reflejar_imagen(imagen, eje):
    """Refleja la imagen sobre el eje especificado."""
    if eje == 'horizontal':
        return cv2.flip(imagen, 0)
    elif eje == 'vertical':
        return cv2.flip(imagen, 1)
    else:
        raise ValueError("El eje debe ser 'horizontal' o 'vertical'.")
```

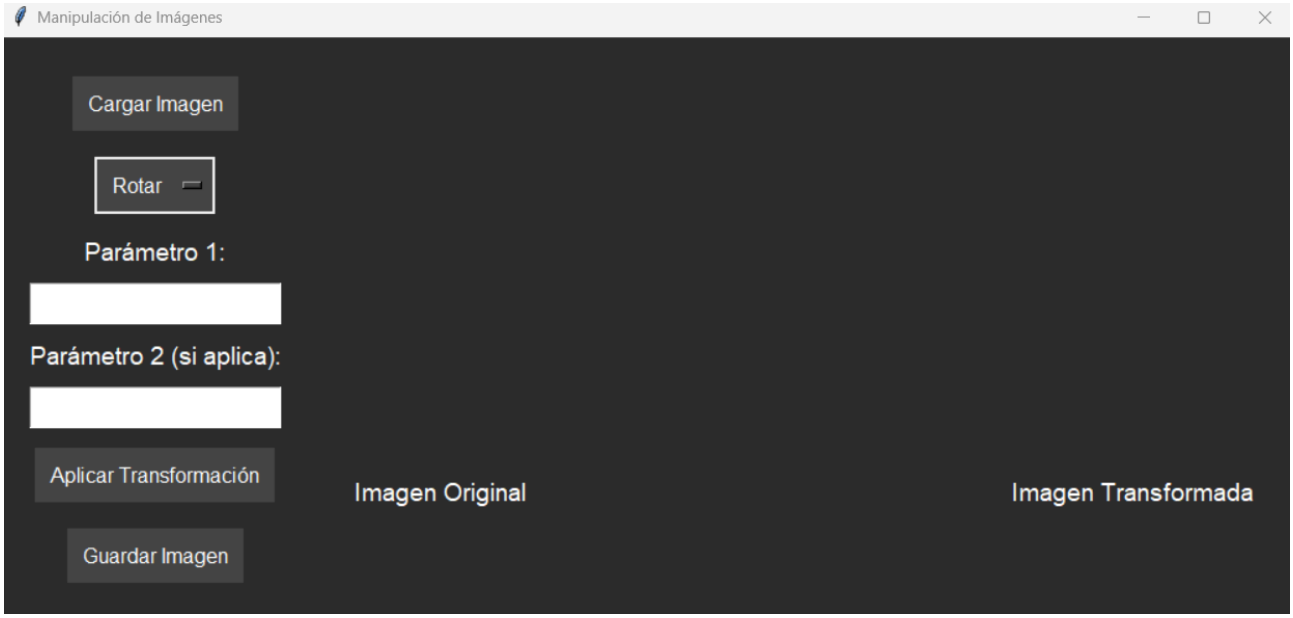
Esta funciona utiliza la función del OpenCv **flip** que hace una rotación de 180 grados o lo refleja, dentro del código se especifica si se refleja ya sea en el eje horizontal o vertical, si algún otro dato es añadido imprime el error y solicita que se haga correctamente.

Esas son las funciones principales para el propósito del proyecto, además de que el código incluye funciones para poder guardar la imagen, mostrar la interfaz de usuario donde se dan las instrucciones.
Contamos con dos programas, el programa principal donde se muestra todo lo ya anteriormente mencionada y un segundo programa que muestra un panel de control para facilitar su manipulación.

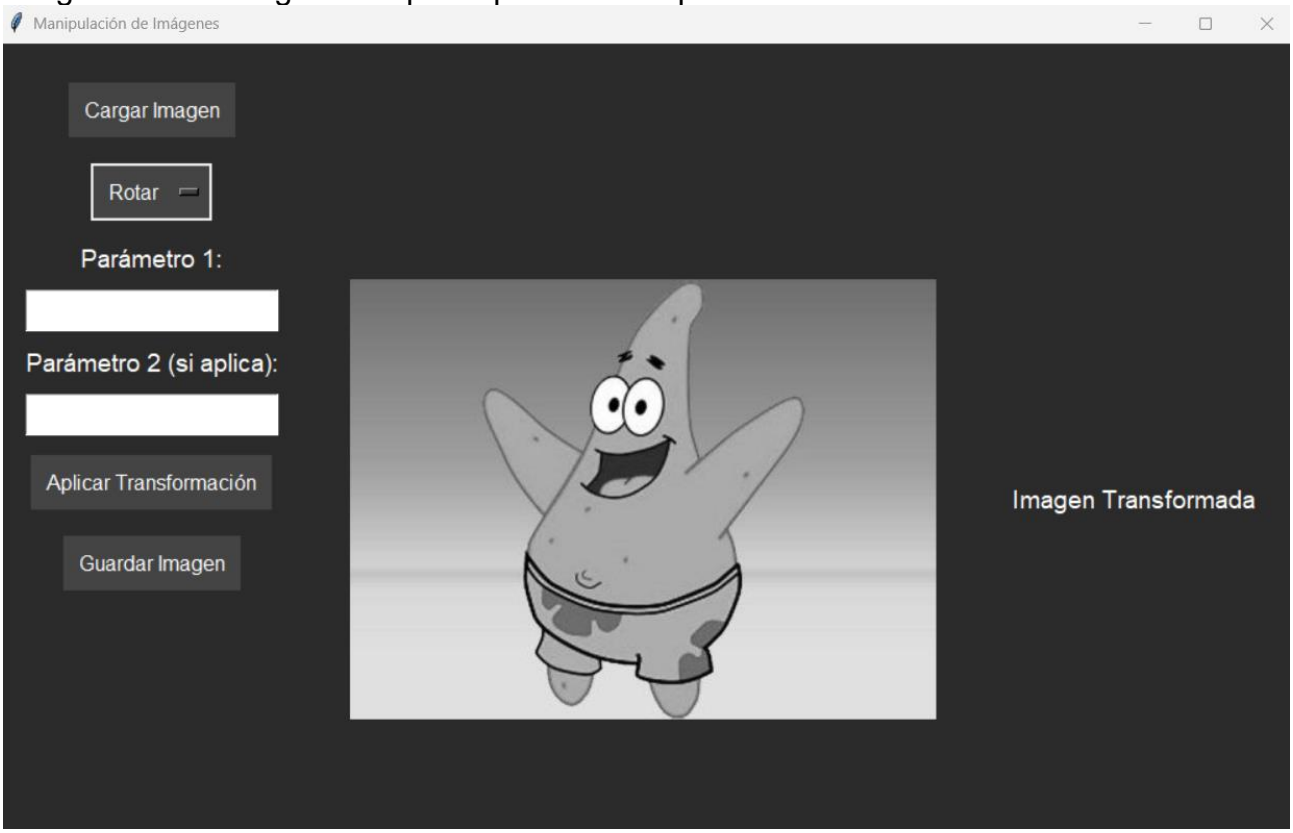


Pruebas realizadas

Inicializamos el programa para el panel de control

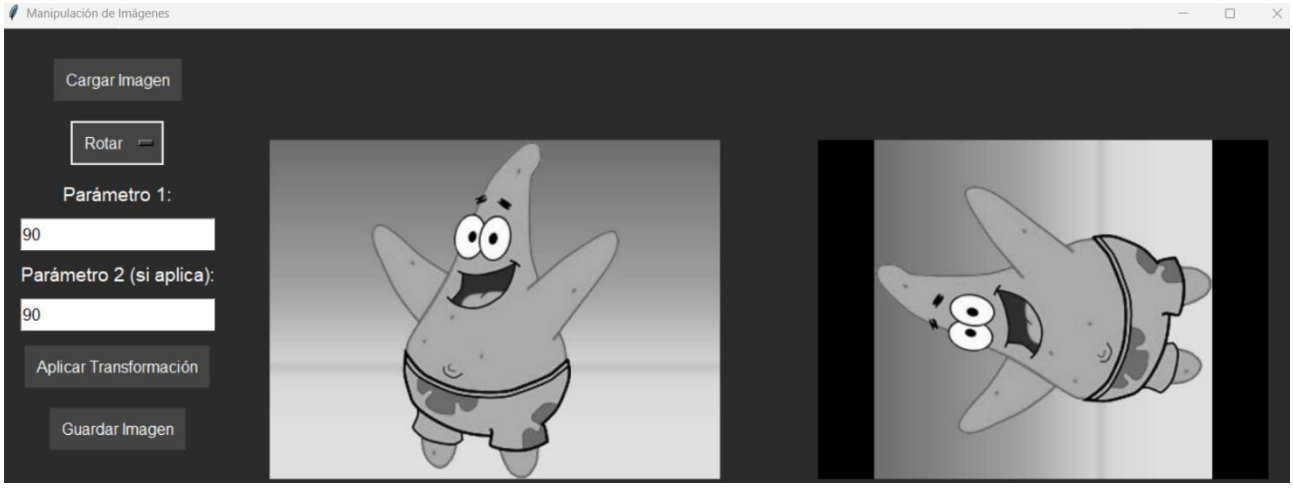


Cargamos una imagen cualquiera para su manipulación.

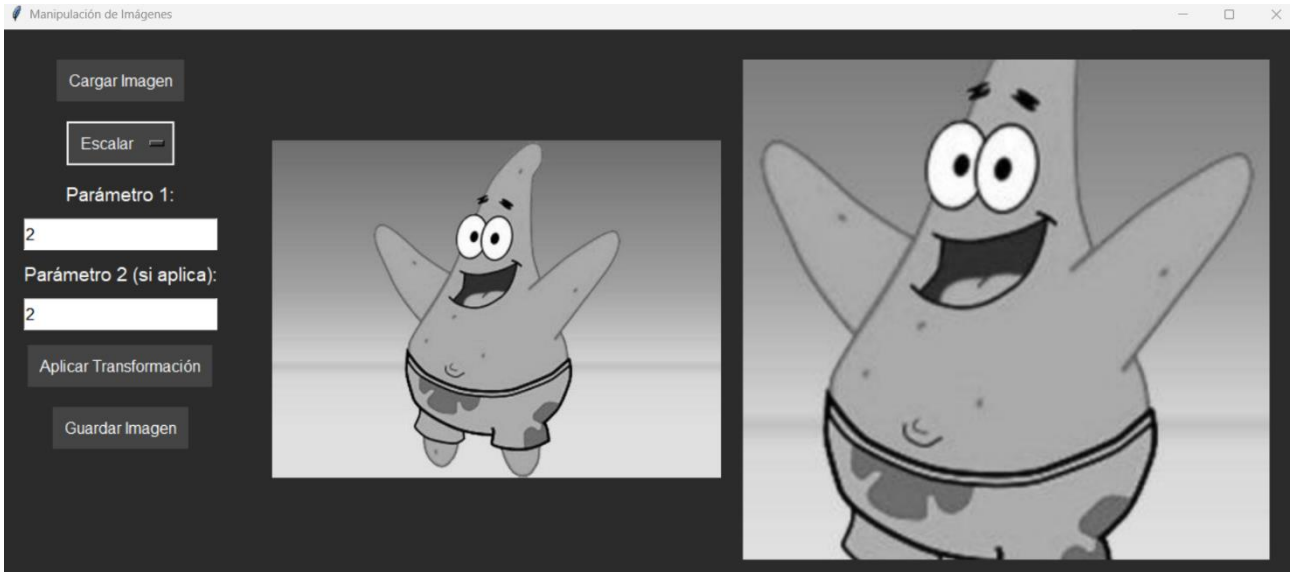


Utilizamos las funciones para su demostración.

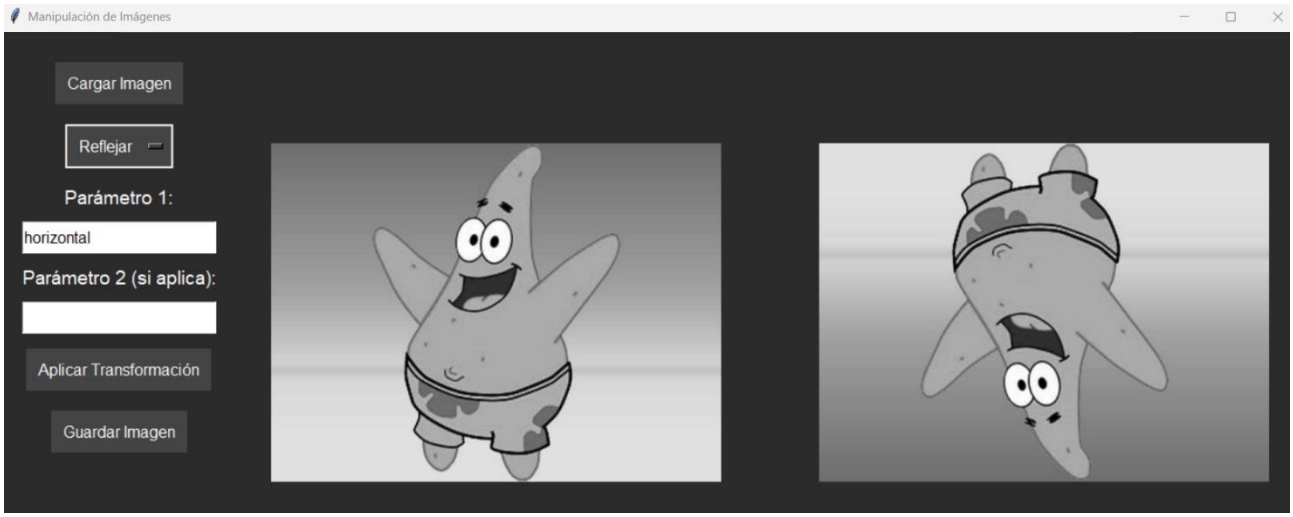
Rotación



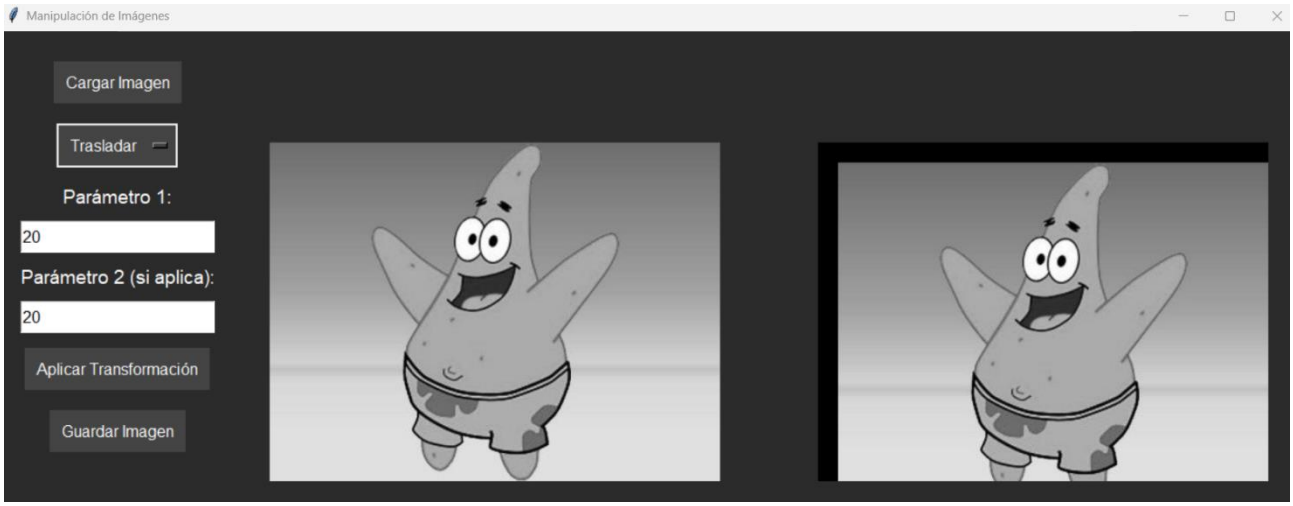
Escalar



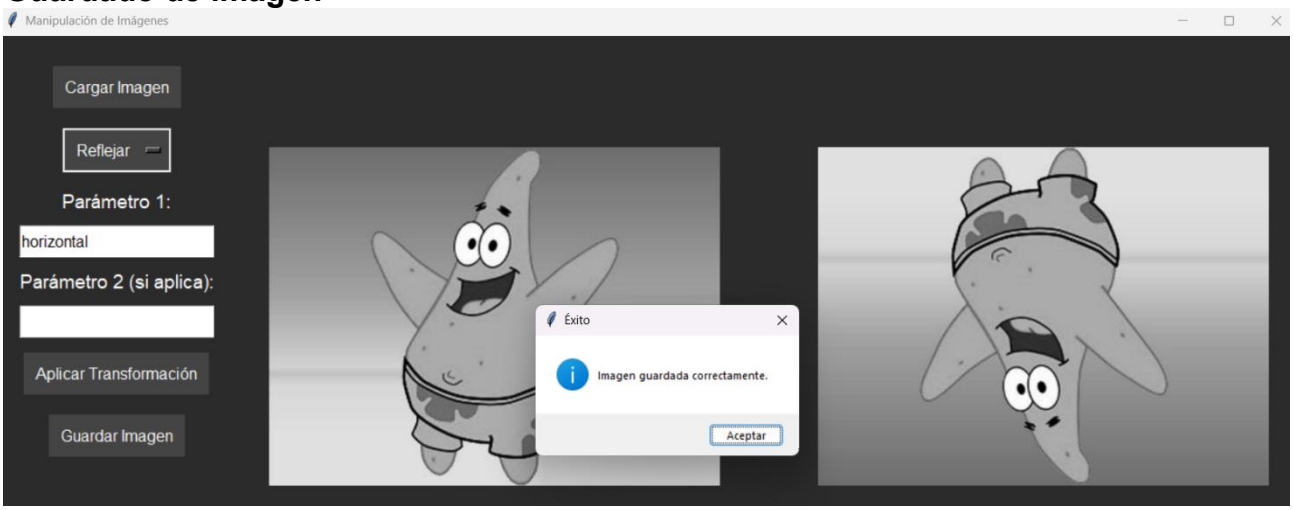
Reflexión



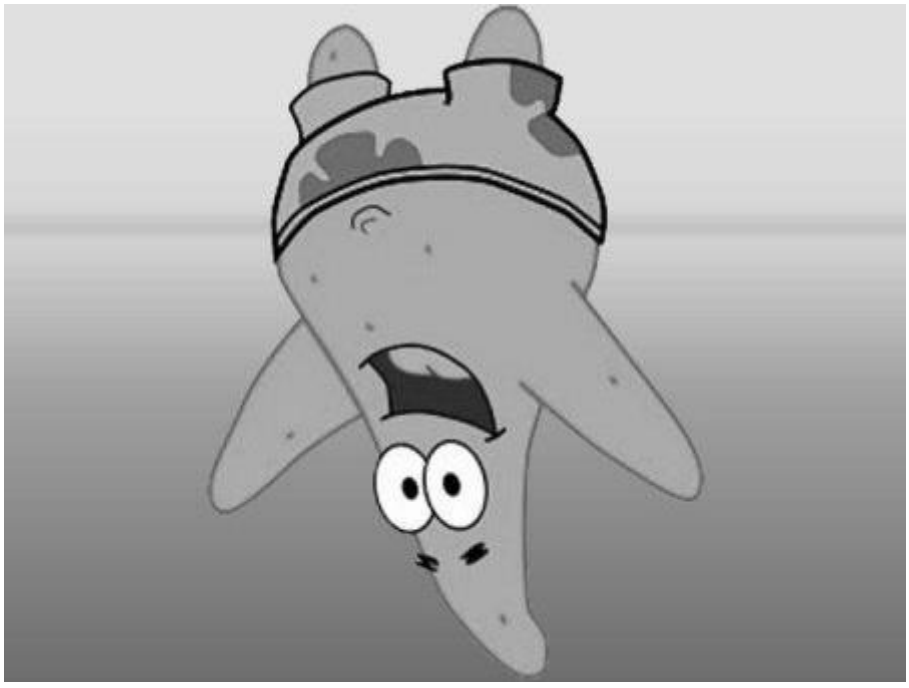
Trasladar



Guardado de imagen



Verificamos si es correcto y si ha guardado la imagen correctamente, después de su edición.



Conclusiones

La algebra lineal es fundamental para la transformación de imágenes, mas enfocado en el uso de matrices, las transformaciones geométricas, ambas nos permitieron manipular imágenes y poder realizar diferentes ediciones a ellas, como la rotación, reflexión, escalarlas y trasladarlas.
Además de que no se hubiera podido llevar a cabo de no ser por el uso de Python y sus librerías especializadas en el uso de matrices y la manipulación de imágenes.
El código demuestra cómo utilizar las librerías para cargar la imagen y aplicar diferentes ediciones y mostrar los resultados.
Además de las ediciones anteriormente vistas que ha se podrían considerar como las más básicas dentro de la edición o manipulación de imágenes, podríamos investigar si es posible hacer manipulaciones mas complejas.