PONTIFICIA UNIVERSIDAD JAVERIANA

PROYECTO DE PROCESAMIENTO DE IMÁGENES Y VIDEO

SOFTWARE DE ESCÁNER DE DOCUMENTOS

Daniel Ricardo Vásquez Maldonado

Luis Daniel Beltrán Rodríguez

Sebastián Rincón Reyes

2020

I. Introducción:

Se desarrolla y se implementa por medio de la concepción de técnicas de procesamiento de imágenes y video vistas, un software que tenga la funcionalidad de escanear documentos desde diferentes posiciones captadas por la cámara y se ejecuten las respectivas operaciones morfológicas, filtrado, encontrar contornos, extracción de contornos y transformación de perspectiva, todo esto dentro de un resultado lo más nítido posible, con la particularidad de la resolución de cámara utilizada, para este proyecto se utiliza la biblioteca Open CV en Python.

II. Objetivos:

- Aplicar las técnicas de procesamiento de imágenes y video aprendidas la asignatura para la implementación de un proyecto funcional.
- Utilizar la librería Open CV y el entorno de programación como herramientas en el desarrollo del proyecto
- Analizar cada etapa del proyecto mediante conceptos y diagrama de flujo para un entendimiento óptimo tanto para nosotros como para el lector.

III. Conceptos y pasos claves:

- **Open CV:**

Esta biblioteca se utiliza para el desarrollo en visión por computadora en tiempo real, es de código abierto, gratis para uso comercial, además es multiplataforma, es decir compatible con entornos como Linux, MacOS, Windows, iOS y Android; una característica muy importante es que está debidamente documentada y muy bien explicada, lo cual la hace mucho más accesible a un aprendizaje más rápido y práctico para diferentes aplicaciones. [1].

Desenfoque:

El desenfoque se utiliza principalmente para quitar ruido de la captura o imagen, es decir lo que contienen las altas frecuencias, para este fin se pueden utilizar ya sea **filtro Gaussiano** que se hace con la convolución de cada uno de los puntos de la matriz de entrada con el kernel de Gauss, los cuales suma para obtener finalmente una matriz en la salida.

El bilateral que considera los píxeles vecinos con pesos determinados al igual que en el de Gauss, los pesos tienen dos componentes, uno que es la misma ponderación utilizada por el filtro de Gauss, en el otro se da la diferencia de intensidad entre los píxeles vecinos y la evaluada, en resumen suaviza los bordes; o **el de mediana**, éste se ejecuta en cada elemento de la imagen, donde se reemplaza cada uno de los píxeles por la mediana de sus píxeles vecinos, alrededor del píxel analizado en una zona cuadrada. [2]

Umbralización:

Este proceso es el método para segmentar las imágenes, el cual básicamente es que, a partir de la captura pasada a escala de grises, se pueden usar umbrales para obtener imágenes binarias, todo con el fin de poder diferenciar los tonos de las intensidades de los píxeles.

En Open CV generalmente se utiliza el **umbral adaptativo**, donde se calcula el umbral de la imagen en pequeñas regiones, consiguiendo umbrales en las diferentes zonas.

También el **umbral simple** donde si el píxel tiene un valor mayor que el del umbral, se le asigna un valor y si no se cumple esta condición se le asigna otro, por ejemplo, la asignación es blanco o negro.

"Después de la aplicación de la técnica para detección de bordes llamada supresión no máxima, los píxeles de borde restantes proporcionan una representación más precisa de los bordes reales de una imagen. Sin embargo, quedan algunos píxeles de borde causados por el ruido y la variación de color. Para tener en cuenta estas respuestas falsas, es esencial filtrar los píxeles de borde con un valor de gradiente débil y conservar los píxeles de borde con un valor de gradiente alto. Esto se logra seleccionando valores de umbral alto y bajo. Si el valor de degradado de un píxel de borde es mayor que el valor de umbral alto, se marca como un píxel de borde fuerte. Si el valor de degradado de un píxel de borde es menor que el valor de umbral alto y mayor que el valor de umbral bajo, se marca como un píxel de borde débil. Si el valor de degradado de un píxel de borde es menor que el valor de umbral bajo, se suprimirá. Los dos valores umbral se determinan empíricamente y su definición dependerá del contenido de una imagen de entrada dada." [3].

- **Extracción de contornos, detección de bordes:**

En ese mismo orden de ideas en el proceso, el paso siguiente es encontrar los contornos, es decir el cuadro delimitador más grande para recortar la imagen de interés.
Utilizamos ***canny edge detection*** a continuación de la extracción del mayor contorno (4 puntos de la imagen).
Se debe enviar una imagen con el menor ruido posible para que se detecten sólo los bordes importantes.
Entonces, ya encontrados los bordes se pasa la imagen por la función de ***findContours***, donde se unen los puntos continuos a lo largo de dichos bordes, para en última instancia tener los contornos.

Utilizando las coordenadas (x,y) del contorno "más grande" se hace la transformación de cuatro puntos usando estas coordenadas, se recorta el contorno con ***warpPerspective*** , teniendo el resultado relevante respecto a la entrada.

IV. Desarrollo:

El procesamiento de imagen que se da para el resultado deseado y propuesto contiene las siguientes etapas:

a. Tomar imagen original.
b. Pasarla a escala de grises.
c. Desenfoque.
d. Binarización.
e. Detectar bordes.
f. Detectar contornos presentes en la imagen, obteniendo el contorno más grande la imagen de interés.
g. Encontrar los puntos de las esquinas del contorno más grande para tener la imagen deseada.
h. Umbral adaptativo para el escaneado de la imagen.
i. Poder guardar el resultado en una carpeta con presionar una tecla.
j. Además, se incluyen barras de seguimiento para "jugar" con los valores de threshold para tener un resultado óptimo según nuestro criterio.

Siguiendo todos los pasos descritos anteriormente se procede a ejecutar y planear el algoritmo del diseño en cuestión, con las funcionalidades de guardar la imagen escaneada y también poder observar en una misma ventana los resultados tanto de proceso como lo son el paso a escala de grises, filtrado, umbralización, extracción de contornos y transformación de perspectiva.

A continuación, se observa el diagrama de flujo correspondiente al diseño:

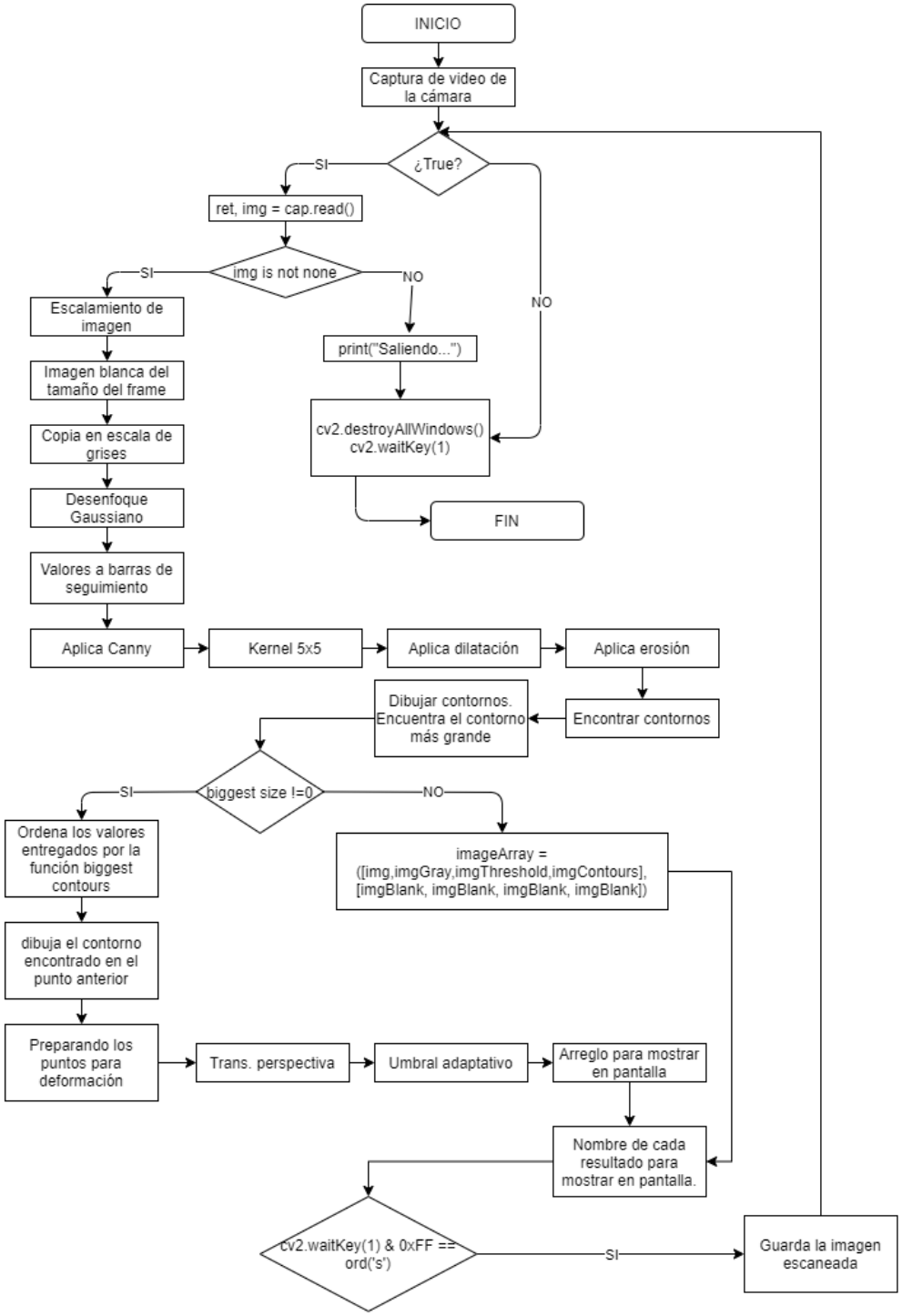

Fig.1 Diagrama de flujo principal.

Se evidencia el paso a paso del tratamiento que se le hace al documento para obtener finalmente el resultado del proceso de escaneado.

Para que el programa principal se ejecute satisfactoriamente, se necesita de determinadas funciones explicadas también en los siguientes diagramas de flujo.

FUNCIONES:

- **BiggestContour:**

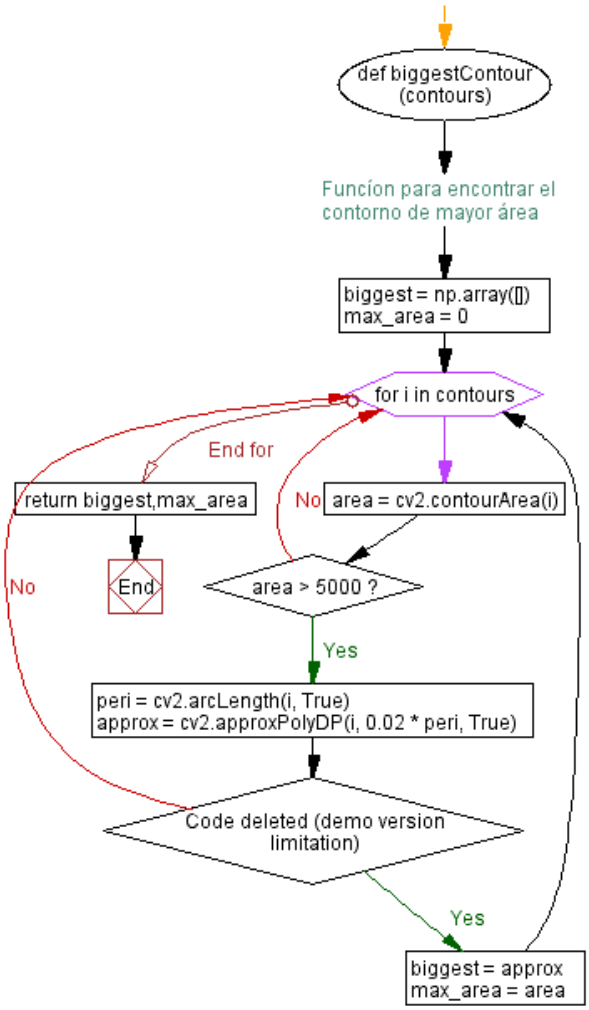

Fig.2 Función-Mayor área

- **Reorder:**

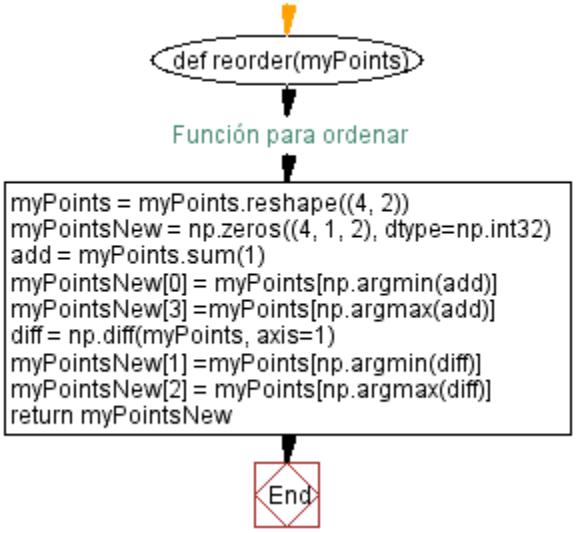

Fig.3 Función Reordenar

- **DrawRectangle:**

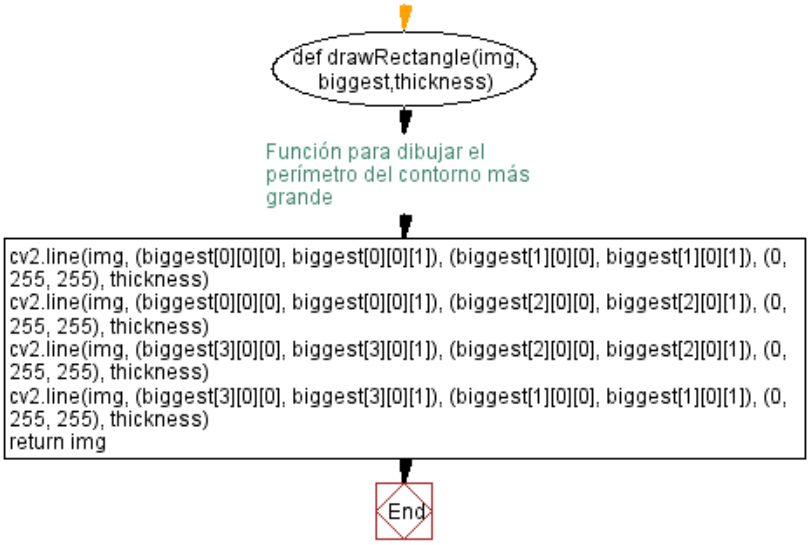

Fig.4 Función drawRectangle

- **InitializeTrackBars:**

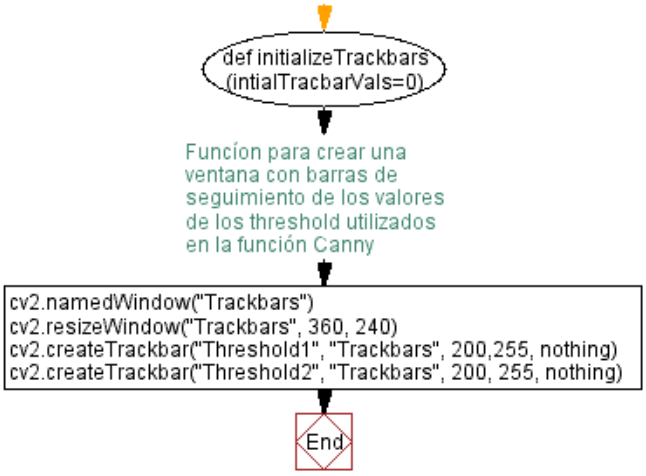

Fig.5 Crear barras de seguimiento

- **ValTrackbars:**

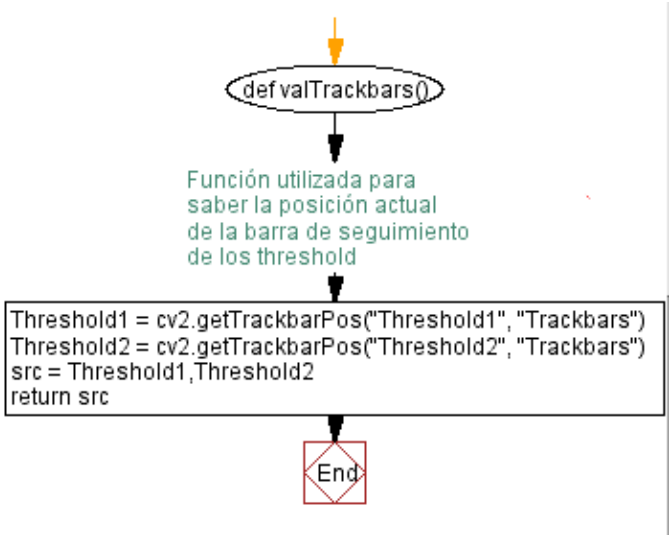

Fig.6 Posición barra de seguimiento

- **Stackimages:**

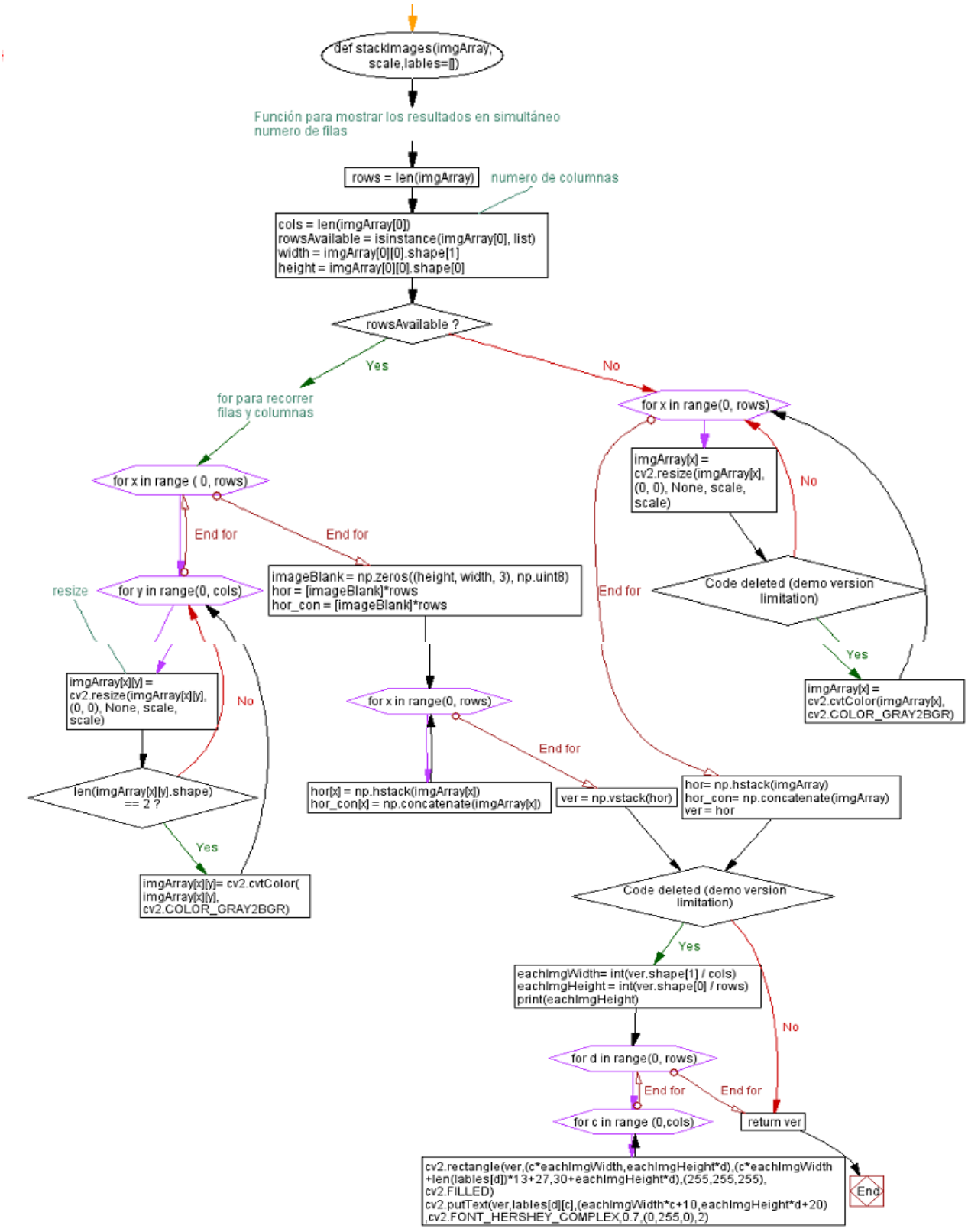

Fig.7 Función para mostrar resultados unificados

V. Resultados:

Fig.8 Momento en el cual se escanea y se guarda la imagen en el equipo.

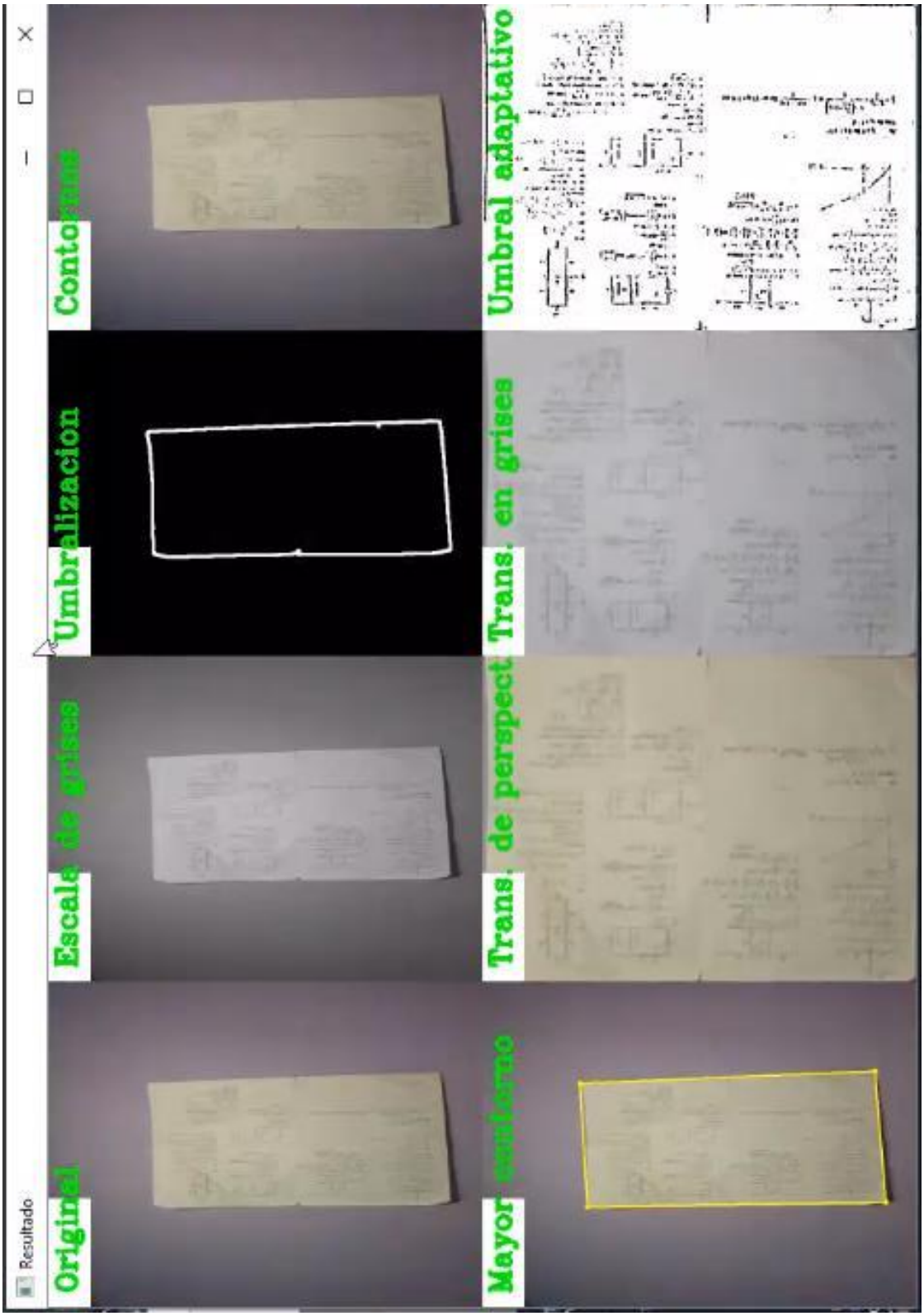

Fig.9 Diferentes resultados, siendo el de mayor contorno y transformación de perspectiva los de mayor relevancia para el fin de este proyecto.

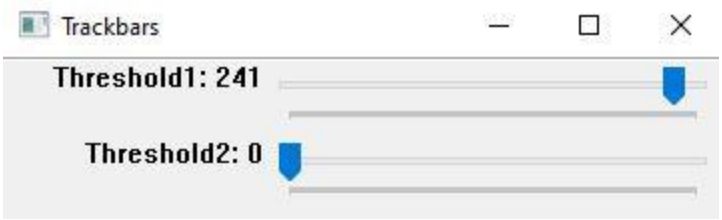

Fig.10 Barras de seguimiento de los umbrales

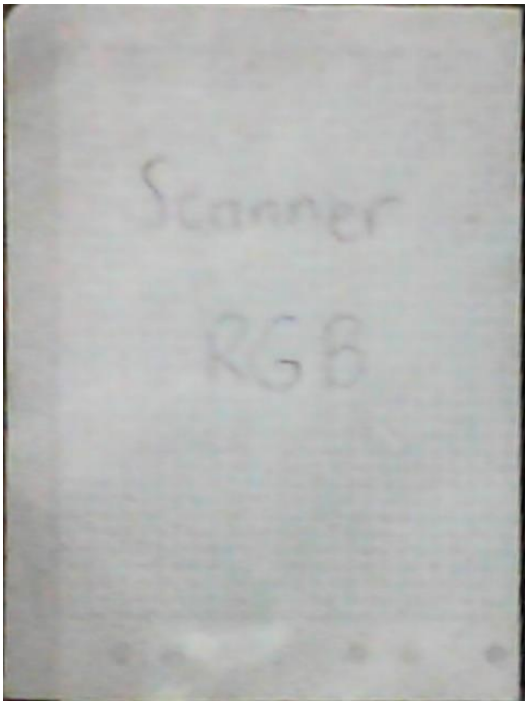

Fig.11 Resultado de la captura hecha en Fig.8

VI. Conclusiones:

A partir de los resultados obtenidos con la implementación de los algoritmos mostrados tanto en los diagramas de flujo como el código fuente se puede contrastar lo ejecutado con el correcto tratamiento que debe tener una imagen y/o video, es decir la adquisición, digitalización y dentro de los respectivos pasos de pre procesamiento y posterior procesamiento en lo que respecta a la mejora de la imagen, utilizando técnicas de remoción de ruido, suavizado y operadores morfológicos para posteriormente aplicar técnicas de procesamiento como lo son la detección de bordes, extracción de contornos y transformación de perspectiva, con el objetivo final de estar lo más cerca posible de un escáner convencional de documentos.

Según los resultados vistos cabe resaltar que la resolución de la cámara es importante e influyente en la adquisición y digitalización del documento que va a ser escaneado. En este caso se hizo uso de la cámara por defecto del computador personal, la cual no tiene una resolución deseada para este tipo de aplicaciones.

La implementación hecha tuvo en cuenta cada uno de los pasos mencionados anteriormente, utilizando las herramientas con mejor funcionalidad para estos fines.

VII. Referencias:

Para los diagramas de flujo de las funciones se usó el software en su versión de prueba Visustin v8.07.
Para el diagrama de flujo principal se utilizó app.diagrams.net.
El código fuente usa las librerías de OpenCV: https://docs.opencv.org/4.4.0/
[1] https://es.wikipedia.org/wiki/OpenCV
[2] https://sites.google.com/site/cg05procesamientodeimagenes/home/smoothing-images
[3] http://en.wikipedia.org/wiki/Canny_edge_detector