INFORME PROYECTO FINAL PROCESAMIENTO DE IMÁGENES

Juan Camilo Sarmiento Peñuela

Camilo Andres Trujillo Muñoz

Sebastian Medardo Diaz Paez.

INTRODUCCIÓN:

Anteriormente, para llevar a cabo el escaner de una fotocopia o algún otro documento que se requería escanear, se procedía a buscar una impresora para llevar a cabo el procedimiento. Hoy en día es común realizar dicho proceso con un elemento de nuestro uso diario y cotidiano. El teléfono celular. Aplicaciones como CamScanner tiene millones de usuario que necesitan de dicha aplicación para llevar a cabo sus rutinas diarias. En el presente proyecto, se va a trabajar en como realizar el diseño y la implementación de una herramienta que nos permita escanear documentos con los conocimientos adquiridos de procesamiento de imágenes. Explicando el diseño desde lo básico hasta la implementación en Python, enseñando los resultados del escaner aplicando filtros y umbralizaciones a la imagen para encontrar el resultado óptimo.

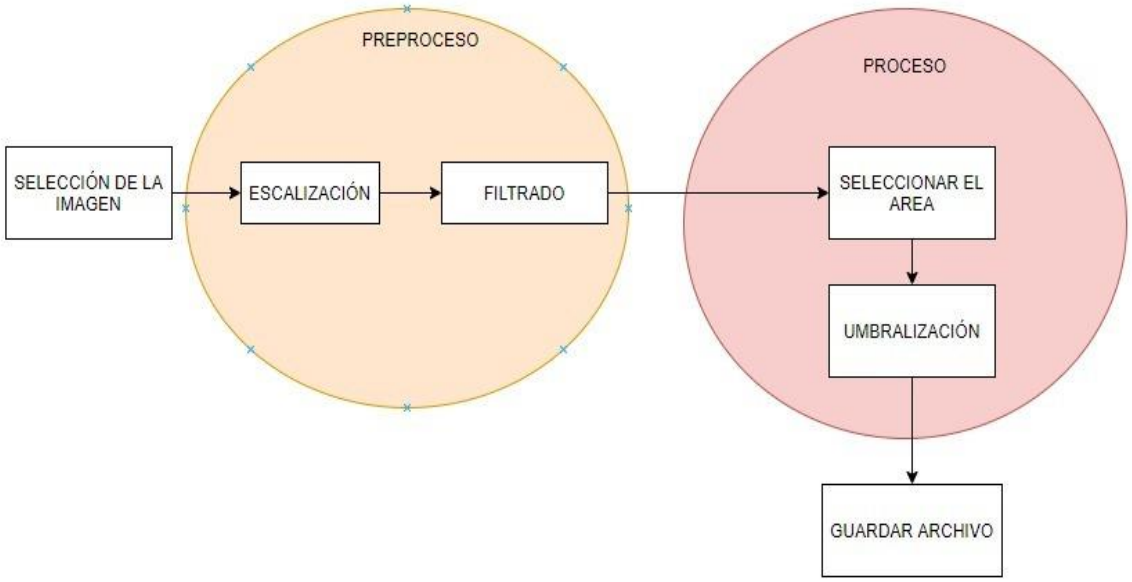
OBJETIVOS:

Objetivo General: Diseñar e implementar una herramienta en Python para recortar y escanear imágenes.

Objetivos Específicos:

- Crear mediante codificación un contorno que permita delimitar manualmente la imagen a escanear.
- Generar mediante codificación un filtro y una umbralización automática aplicable a la imagen.
- Realizar la transformación de perspectiva a partir de la imagen generada después de delimitar su contorno.

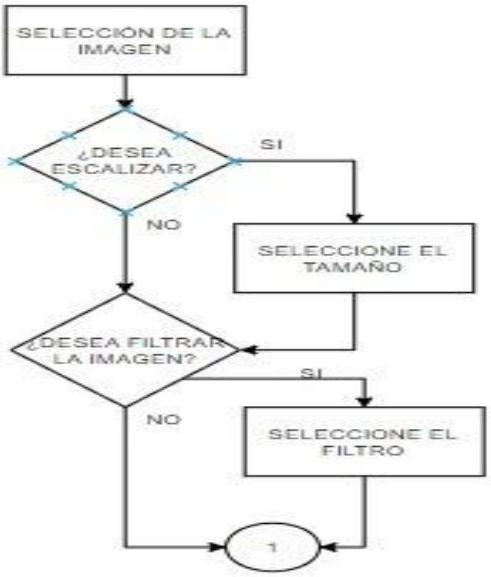
DIAGRAMA DE BLOQUES DEL SISTEMA:



Descripción de los bloques:

- **SELECCIÓN DE LA IMAGEN:** la primera parte que se realiza es hacer la selección de la imagen. Dicha imagen es una previamente creada y guardada en una carpeta de destino. Con ayuda de las configuraciones que *simple gui* nos brinda se realiza la siguiente interfaz para hacer la selección. Se escoge inicialmente la carpeta, se selecciona la imagen y para imágenes de formato .png y .gif se puede ver una visualización previa.
- **ESCALIZACION:** debido a que la rapidez de la herramienta puede variar al procesar imágenes de tamaños muy grandes. Evitando estos contratiempos se ha diseñado una opción para poder escalizar la imagen de un tamaño nxm inicial a un tamaño n2xm2 definido por el usuario. En la pestaña de previsualización el usuario se puede hacer una idea del tamaño de la imagen para hacer la selección de la escalización. Existe un camino directo en el diagrama de flujo, esto es porque el usuario tiene la facultad para decidir si escaliza o no.
- **FILTRADO:** con el fin de eliminar la mayoría del ruido posible y de las imperfecciones en la imagen, se ha optado por que el usuario tenga en su disponibilidad el uso de varios filtros a su conveniencia. Con lo cual puede seleccionar entre varios filtros y escoger el que mejor respuesta de. Los filtros aplicables en esta pestaña son los filtros promedio, mediana, filtro Gaussiano y un filtro bilateral.
- **SELECCIONAR EL AREA:** en la selección del área se procede a escoger la parte de la imagen a escanear. Aparece un recuadro sobre la imagen para que el usuario pueda seleccionar de la imagen previamente escalizada, cual es la sección de interés para posteriormente realizar el escáner. El recuadro que le aparece a la persona permite variar los límites de la imagen tanto de forma vertical como de forma horizontal.

Para el Desarrollo del sistema se tiene el siguiente diagrama de flujo:



DESARROLLO

Empleando las instrucciones del simpleGUI se crea inicialmente un menú que permita escoger la carpeta raíz donde se encuentra alojada la imagen, las instrucciones más relevantes en este paso son FolderBrowse() para buscar la carpeta y ListBox() para desplegar las imágenes que serán procesadas (imagen 1).

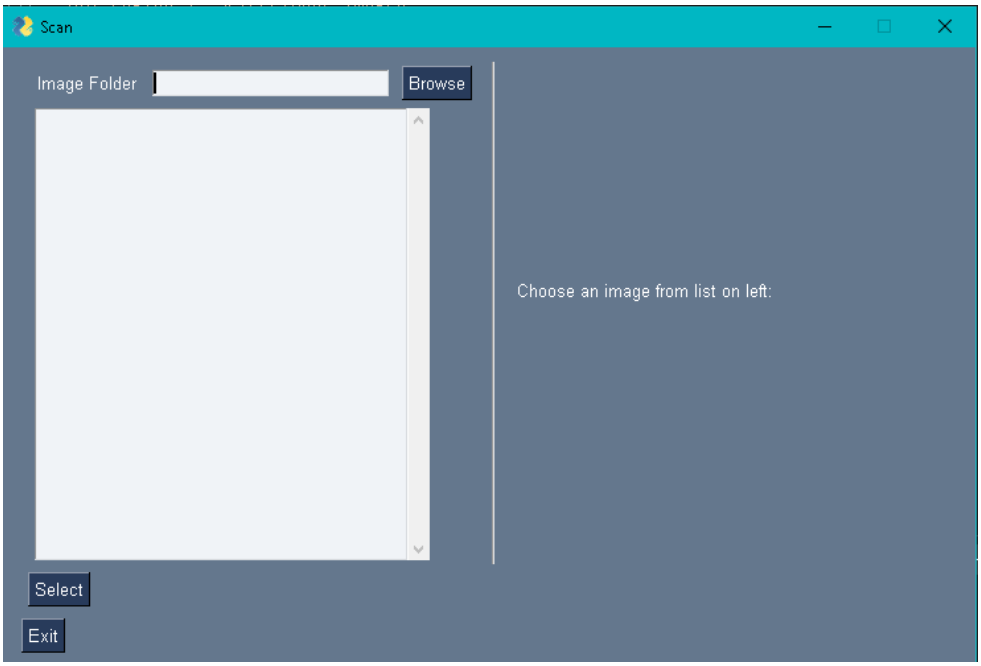


Imagen 1. Selección de archivo para procesarse.

Empleando botones de selección se escoge la imagen de la lista del lado izquierdo y su previsualización se muestra en el lado derecho de la imagen 1. Una vez seleccionada la imagen se le permite al usuario escoger la escalizacion de la imagen empleando un scroll para ampliarla, las funciones relevantes en este paso son resize donde se emplea una interpolación de área (imagen 2 e imagen 3).

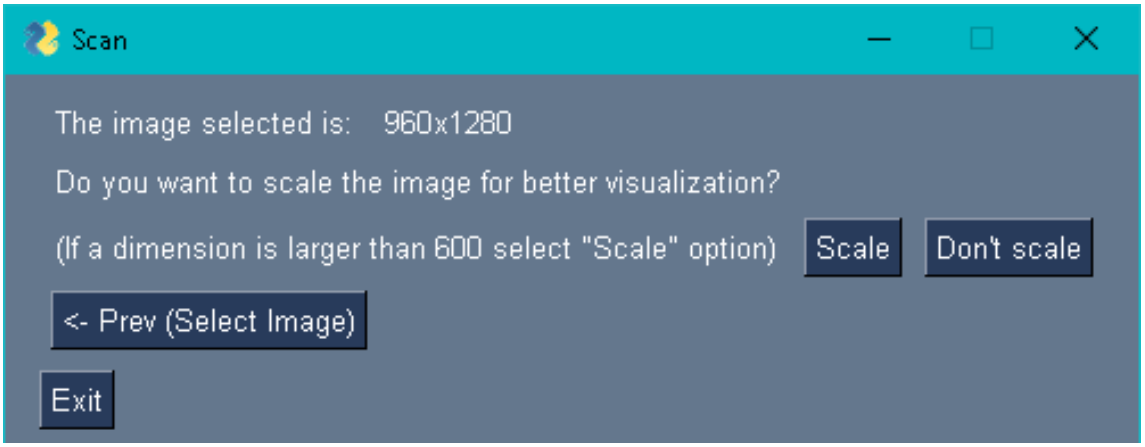


Imagen 2. Mediante botones se pregunta si se desea o no escalizar la imagen o devolverse a seleccionar otra imagen.

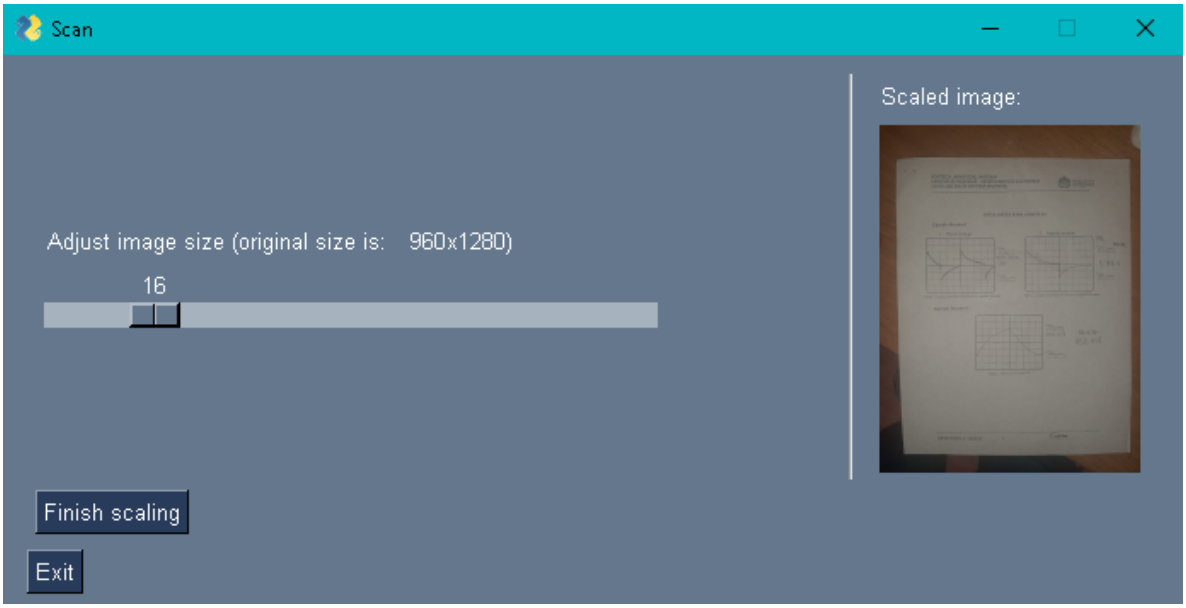


Imagen 3. Mediante el scroll se elige el tamaño al cual se desea reajustar el tamaño de la imagen, el valor indicado es el porcentaje con respecto a la imagen original.

Una vez escalizada la imagen se le permite al usuario aplicar filtros de tipo averaging, median, Gaussian y bilateral, en cada una se da la posibilidad de escoger el tamaño del kernel y el diámetro y sigma para el caso del filtro bilaterial (imagen 4). Las funciones mas relevantes implementadas en este paso son BilateralFilter(), Blur(), MedianBlur(),

GaussianBlur(), las cuales reciben de parámetros la imagen escalizada y los valores seleccionados en el scroll correspondiente.

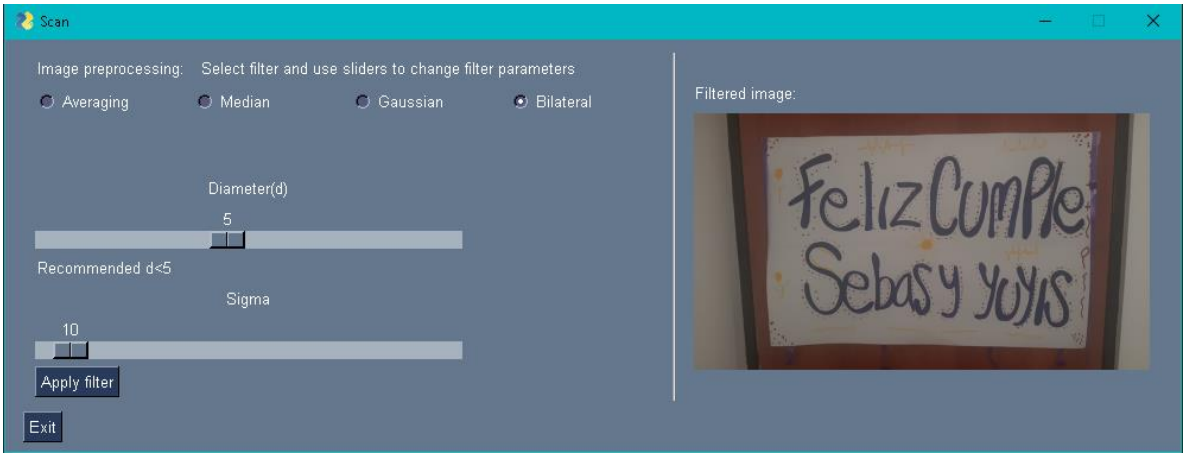


Imagen 4. Aplicación de filtros a la imagen escalizada.

El siguiente paso es aplicar una ventana trapezoide y desplegar los 4 vértices sobre la imagen para delimitar el área a escanear (imagen 5 e imagen 6). En la codificación se inicia creando una función que recibe los colores del borde del trapezoide, color de fondo, color de los vertices, tamaño de la imagen y una constante que permite tener un gap con respecto al borde de la imagen para que sea cómodo seleccionar los vértices. Para crear el contorno se usa la función np.array() la cual genera un borde basándose en los limites de la imagen y aplica un gap de 5 pixeles para tener un margen. Se define la función get_border_index() la cual retorna un borde según la coordenada que yo fije manualmente, este borde y la coordenada se muestra en la pantalla mediante la función set_border que reajusta la imagen con el borde y las coordenadas de los vértices que manualmente se escojan.

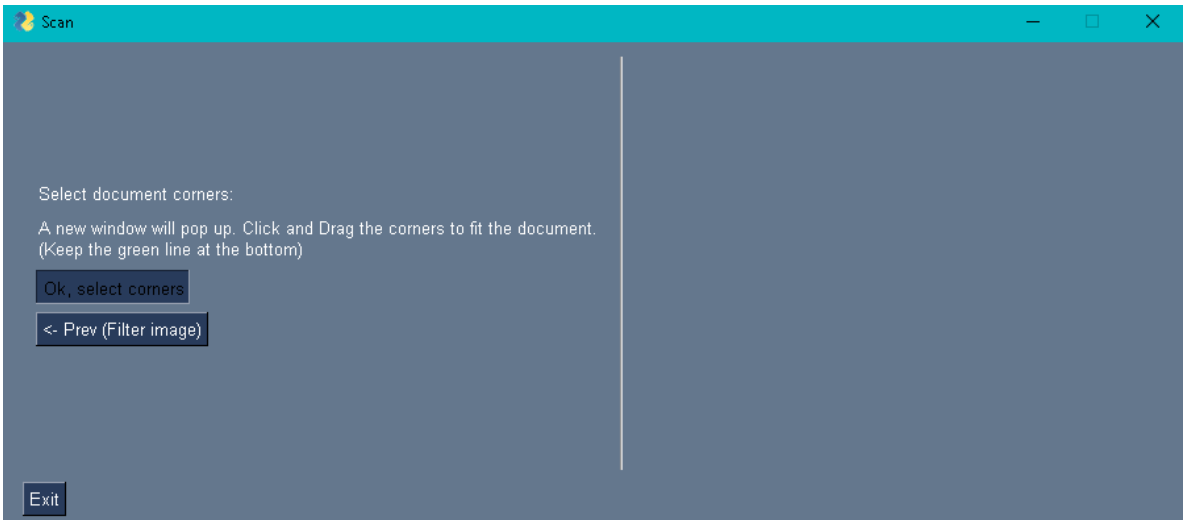


Imagen 5. Menú de selección para seleccionar las 4 esquinas o retornar al menú de aplicación de filtro.

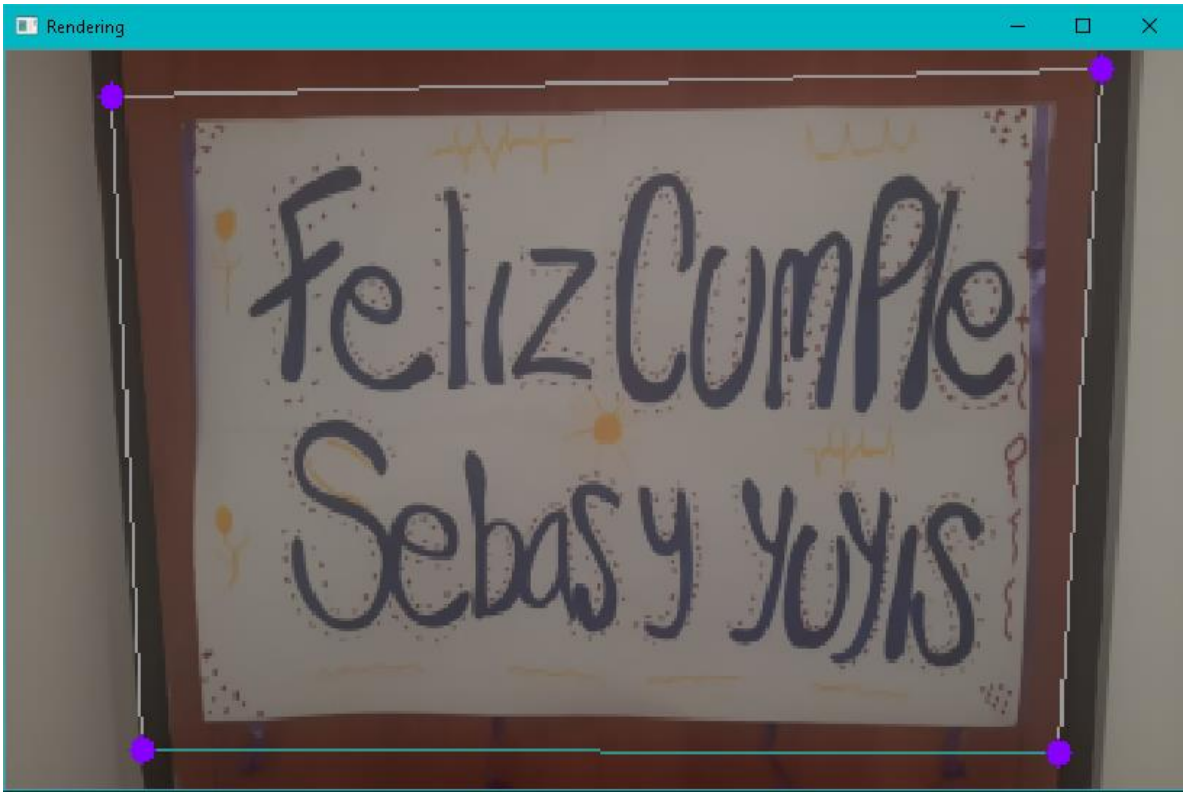


Imagen 6. Selección de ventana de interés empleando 4 vértices.

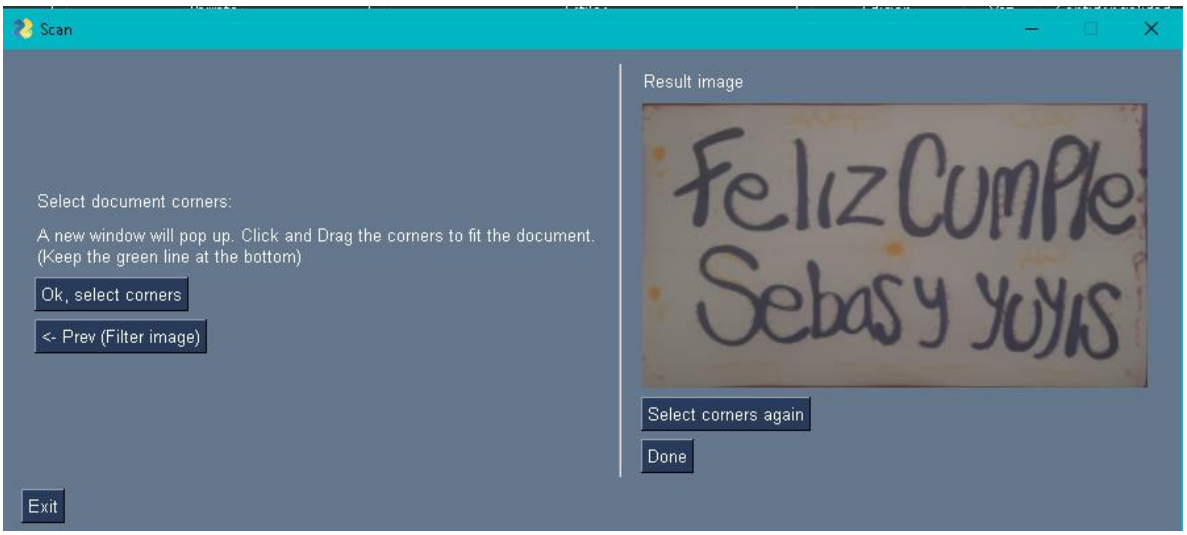


Imagen 7. Resultado de la imagen delimitada.

Una vez escogido los limites se procede a escoger la umbralizacion para depurar el texto y realzar los contornos de las letras, para este propósito se procesa la imagen con la función

adaptiveThreshold(), dentro de esta función se pasa la imagen a escala de grises con la función BGR2GRAY, se aplica la umbralizacion con la función adaptive_thresh_mean_c() el cual toma como parámetro el valor C fijado en el scroll y thresh_binary() el cual pasa la imagen de escala de grises a binaria. El valor de umbralizacion y tamaño del bloque son fijados por el scroll en dado caso que se escoja una umbralizacion simple o se parametriza con el tamaño del bloque y el valor de la constante C para umbralizaciones adaptativas.

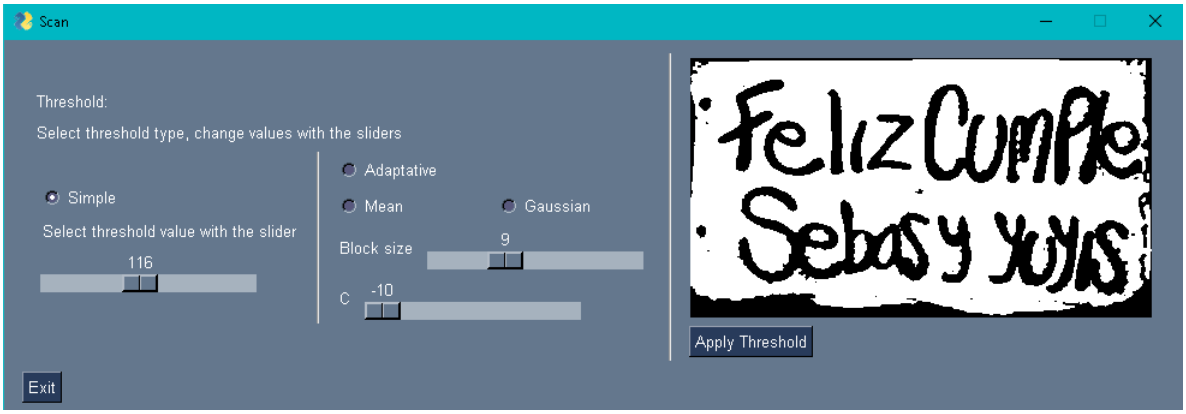


Imagen 8. Aplicación de umbralizacion sobre la imagen.

Una vez aplicada la imagen, se procede a guardar el resultado en distintos formatos, entre ellos están png, jpeg,bmp y pdf, para esto se hace uso de las funciones save para el formato pdf y la función imwrite para los demás formatos.

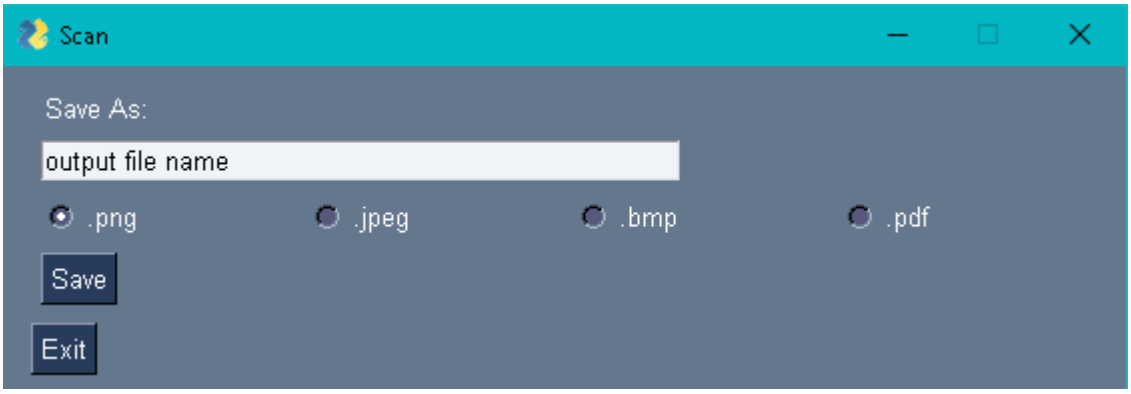
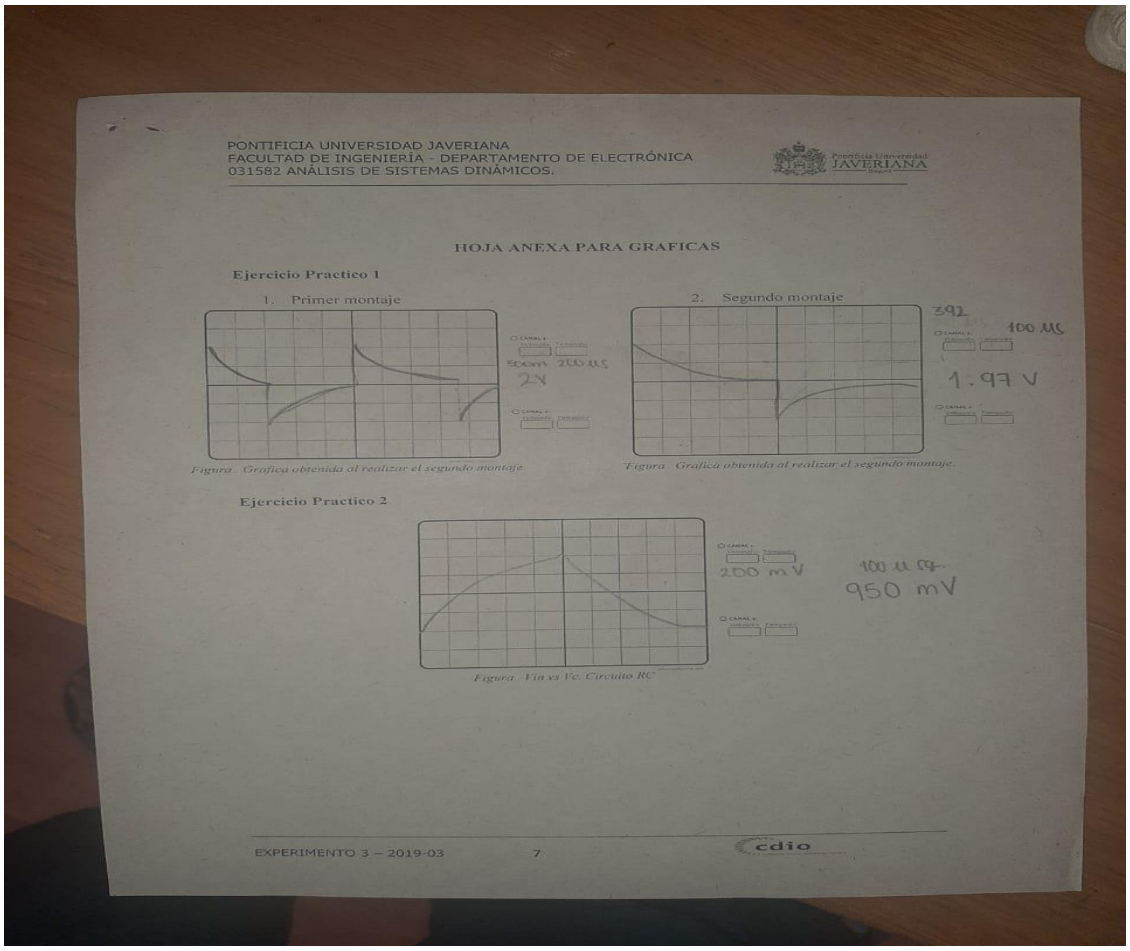


Imagen 9. Pop up para seleccionar el formato de salida.

Resultados

Para el protocolo de pruebas se toman varias imágenes de entrada y se analizan sus resultados en cada etapa:



PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA - DEPARTAMENTO DE ELECTRÓNICA
031582 ANÁLISIS DE SISTEMAS DINÁMICOS.

HOJA ANEXA PARA GRAFICAS

Ejercicio Practico 1

1. Primer montaje

2. Segundo montaje

Figura . Grafica obtenida al realizar el segundo montaje.

Figura . Grafica obtenida al realizar el segundo montaje.

Ejercicio Practico 2

Figura . Vin vs Vc. Circuito RC

EXPERIMENTO 3 – 2019-03 7

Imagen 9. Fotografía original del documento a escanear.

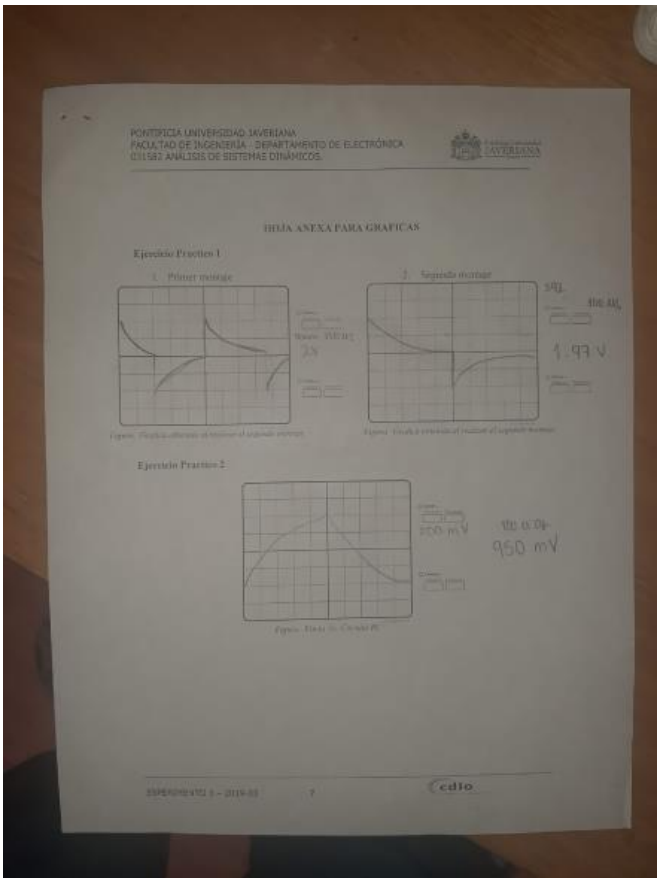


Imagen 10. Imagen escalizada y con filtro bilateral de diámetro 7 y sigma de 49

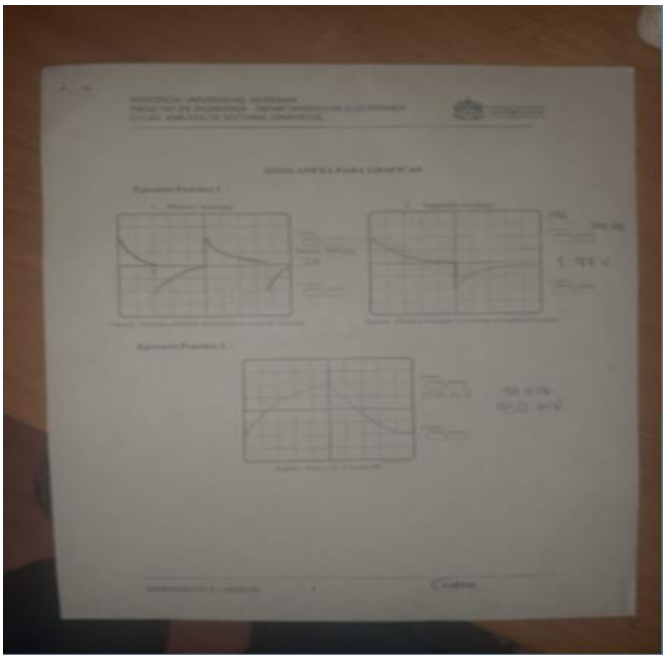


Imagen 11. Imagen escalizada y filtro gaussiano con un kernel de 5

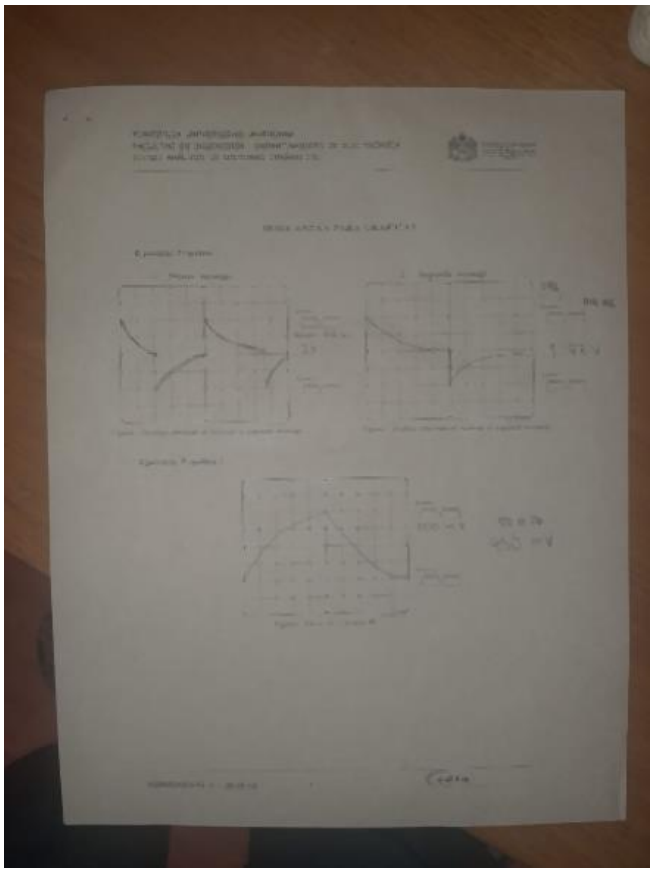


Imagen 12. Aplicación Filtro mediana y kernel de 3

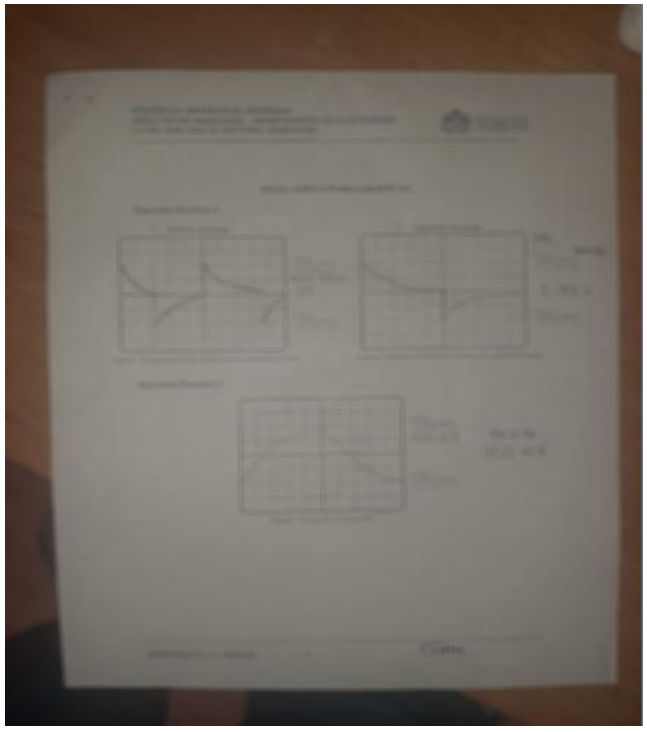


Imagen 13. Filtro promedio y kernel de 5

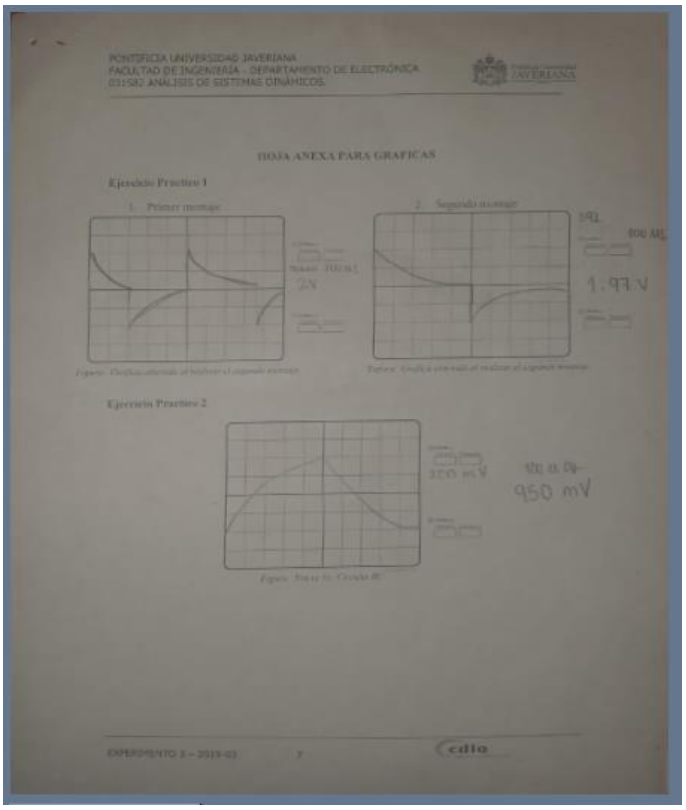


Imagen 14. Resultado de la imagen delimitada en sus bordes

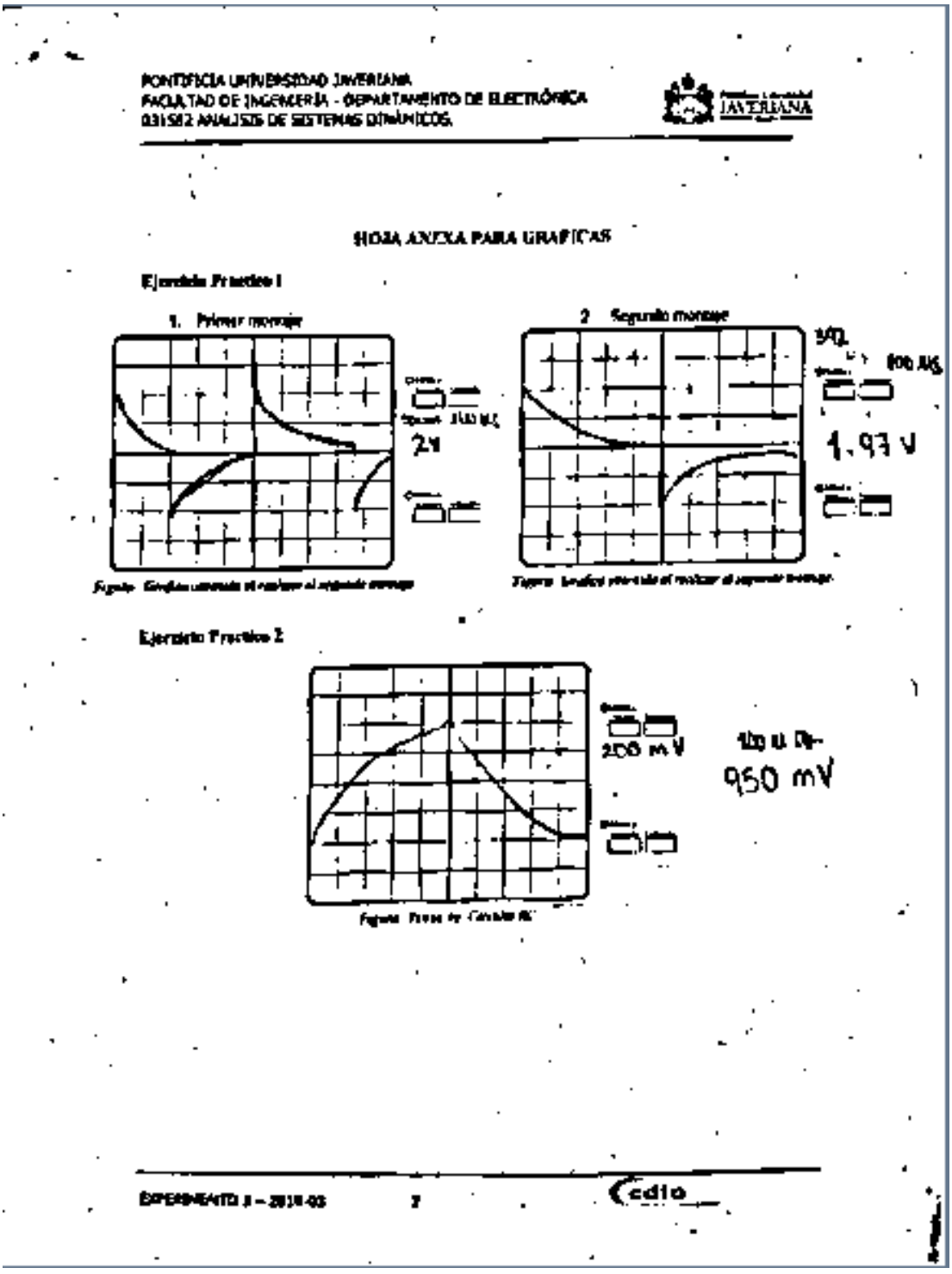


Imagen 15. Resultado final después de umbralizacion.

Imagen 16. Filtro bilateral diámetro 3 y sigma de 157.

Imagen 17. Filtro gaussiano con kernel de 3.

Imagen 18.Filtro mediana y kernel de 6

Imagen 19. Filtro promedio y kernel de 3.

Imagen 20. Imagen delimitada.

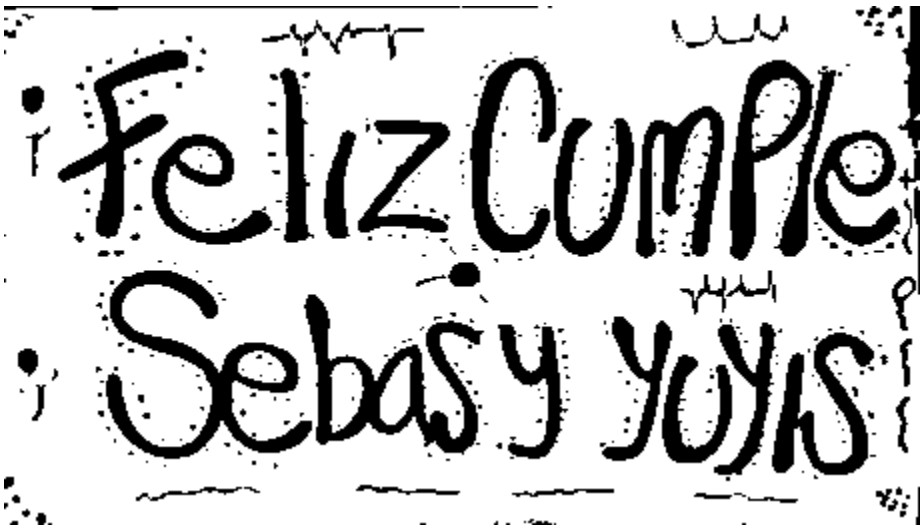


Imagen 21. Imagen de salida después de umbralizacion.

Conclusiones

- De acuerdo a los resultados obtenidos, el filtro que mejor permite la legibilidad es el filtro bilateral debido a que permite hacer filtrados mas suaves y eliminación de ruido, esto se debe a que el filtro bilateral permite diferenciar con mayor claridad los bordes del ruido manteniendo los bordes de los textos mas legibles (en su formula se le asigna un peso distinto para el espacio y el rango donde trabaja a diferencia de los demás filtros que procesan la imagen aplicando un único valor de filtrado sobre todo el espacio)
- En documentos de letra pequeña se obtuvo mejores resultados con la umbralizacion adaptativa y empleando valores de la constante C positivos mayores a 5 y tamaños de bloque de 21. Esto se debe a que se consigue un mayor equilibrio en escenarios de baja o alta iluminación, su valor esta dado por los alrededores del pixel donde esta trabajando a

diferencia de la umbralizacion simple donde toma valores binarios de 0 o 1 para clasificar cada pixel.

- Debido a trabajos previos de la materia de procesamiento de imágenes y visión, la interpolación por área fue escogida entre diferentes tipos de interpolación en estudio. Dicha interpolación tiene la característica de que permite escalizar a diferentes de valores enteros. Es decir, permite hacer escalizaciones de valores como 0.5 o 1.5 etc. Esta característica hace quesea útil para la decimación y para las ampliaciones de tamaño, en estas últimas se comporta de manera similar al método nearest.

Referencias:

- https://pysimplegui.readthedocs.io/en/latest/cookbook/#multiple-columns
- https://pysimplegui.readthedocs.io/en/latest/call%20reference/#graph-element
- https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html
- https://www.geeksforgeeks.org/python-bilateral-filtering/
- https://stackoverflow.com/questions/37203970/opencv-grayscale-mode-vs-gray-color-conversion