

Instituto Tecnológico de Costa Rica

Ingeniería en Computación

IC1803 - Taller programación

Tarea Programada 1: Filtrar imágenes

Profesor: Saúl Calderón Ramírez

Estudiantes:

Franco Vinicio Rojas Lagos 2022437823

Dayana Xie Li 2022097967

Abel Gutiérrez Martínez 2022437323

Primer semestre 2022

Contents

1	Introducción	3
2	Análisis	3
3	Diseño	5
4	Implementación y pruebas	8
4.1	Librerías utilizadas	8
4.2	Prueba 1 para el filtro con matrices pequeñas	8
4.3	Prueba 2 para el filtro con matrices pequeñas	10
4.4	Pruebas 3 para el filtro con matrices pequeñas	12
4.5	Prueba 4 de filtrado usando la imagen manNoisy.PNG	14
4.6	Prueba 5 de filtrado usando la imagen manNoisy.PNG	16
4.7	Prueba 6 de filtrado usando la imagen <i>mujerRuido.PNG</i>	18
4.8	Comparación del tiempo de ejecución	20
4.9	Manual de usuario	20
5	Conclusiones	36
6	Bibliografía	37

1 Introducción

El ruido de sal y pimienta en imágenes es un problema que tal vez no solemos escuchar o ver muy frecuente debido a que actualmente solemos interactuar con dispositivos de alta calidad diseñados para tener la menor cantidad de fallas posibles, pero sí es un problema que es muy común en las áreas profesionales y sofisticadas cuando se requiere captar y traspasar imágenes. Este tipo de ruido es causado por el mal funcionamiento en los sensores de las cámaras, la mala transmisión de datos o debido a problemas relacionados al hardware del dispositivo. La molestia que genera este tipo de ruido es que crea píxeles en blanco y negro, el píxel de valor cero es al que se conoce como la pimienta, mientras que el píxel de valor 255 es el que se le conoce como sal, siendo estos, valores indeseados por lo que se le catalogan como ruido. El problema a resolver en este trabajo práctico es eliminar completamente o lo más que se pueda, este ruido de las imágenes en escala de grises, por medio de un filtro de medianas que consiste en suplantarlo cada píxel de la imagen con la mediana de una ventana de determinado tamaño, el cual será seleccionado por el usuario. El sistema de filtrado poseerá dos diferentes algoritmos de ordenamiento que ordenarán la ventana, el algoritmo de selección y quick-sort, para así poder seleccionar el valor que deseamos suplantarlo del píxel con ruido con uno sin ruido. Eliminar el ruido será el reto a enfrentar y a lograr de este trabajo práctico. [1]

2 Analisis

En la figura 1 se describe el caso de uso que tiene el sistema a la hora de filtrar imágenes, desde las entradas esperadas de parte del usuario: que son la selección de la ruta, la imagen o la matriz, la ruta en donde desea guardar la imagen filtrada, el tipo de algoritmo de ordenamiento y el tamaño de la ventana; las funcionalidades que debe realizar el sistema internamente con las entradas recibidas y como salida, se espera que se guarde la imagen filtrada con la menor cantidad de sal y pimienta posible en la ruta especificada.

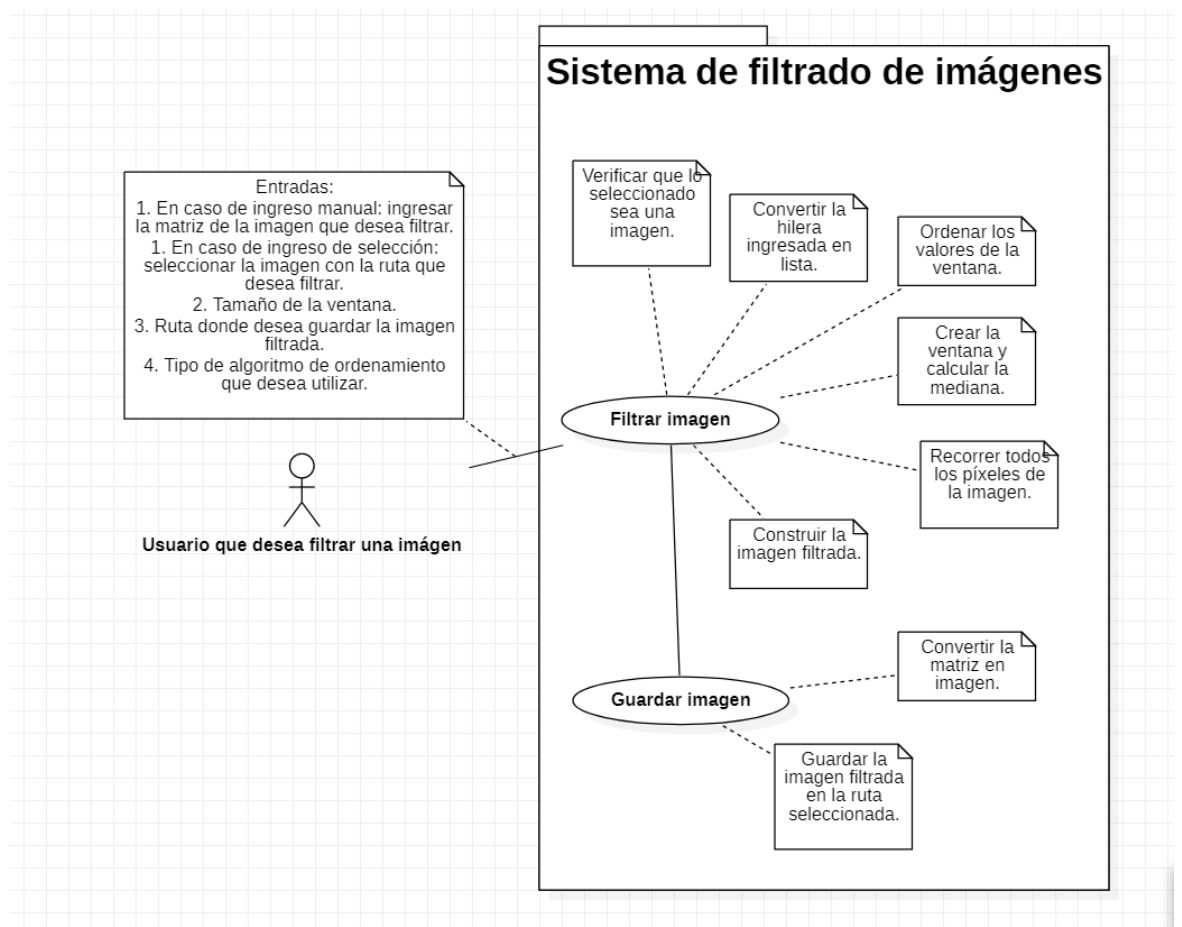


Figure 1: Diagrama de casos de uso

Definición de entradas, salidas y restricciones

- **Entrada:** imagen en escalas grises de cualquier formato y ventana o vecindario a utilizar en el proceso de filtrado.
- **Salidas:** imagen filtrada, en formato imagen.
- **Restricciones:** debe ser una imagen con escala de grises, no puede ser una imagen de color. Solo puede ser datos de tipo imagen.

Subproblemas

1. Recorrer la imagen por cada pixel.

2. Calcular la mediana de una lista ordenada.
3. Guardar la ruta donde se almacenará la imagen filtrada.
4. Solicitar la ruta y nombre del archivo.
5. Solicitar tamaño de la ventana o vecindario al usuario.
6. Ordenar los elementos de una ventana.
7. Construir la imagen filtrada.
8. Crear la ventana o vecindario.
9. Verificar que el archivo seleccionado sea una imagen.
10. Transforma la imagen a una matriz.

3 Diseño

En esta sección, se detallarán todas las librerías utilizadas en el trabajo práctico y el pseudocódigo de las funciones implementadas más complejas. La figura 2 se presenta el diseño del sistema por medio del diagrama de clases del sistema de filtrado de imágenes, en el cual se puede observar que en cada archivo .py está compuesta por diferentes clases que se utiliza en el sistema de filtrado, divididos en cinco archivos importantes, se hace mención al Controlador, Manipulador_Archivo, Ordenador_Seleccion, Ordenador_Quick_Sort y Filtro_Medianas, cada uno con sus respectivos parámetros y entradas esperadas.

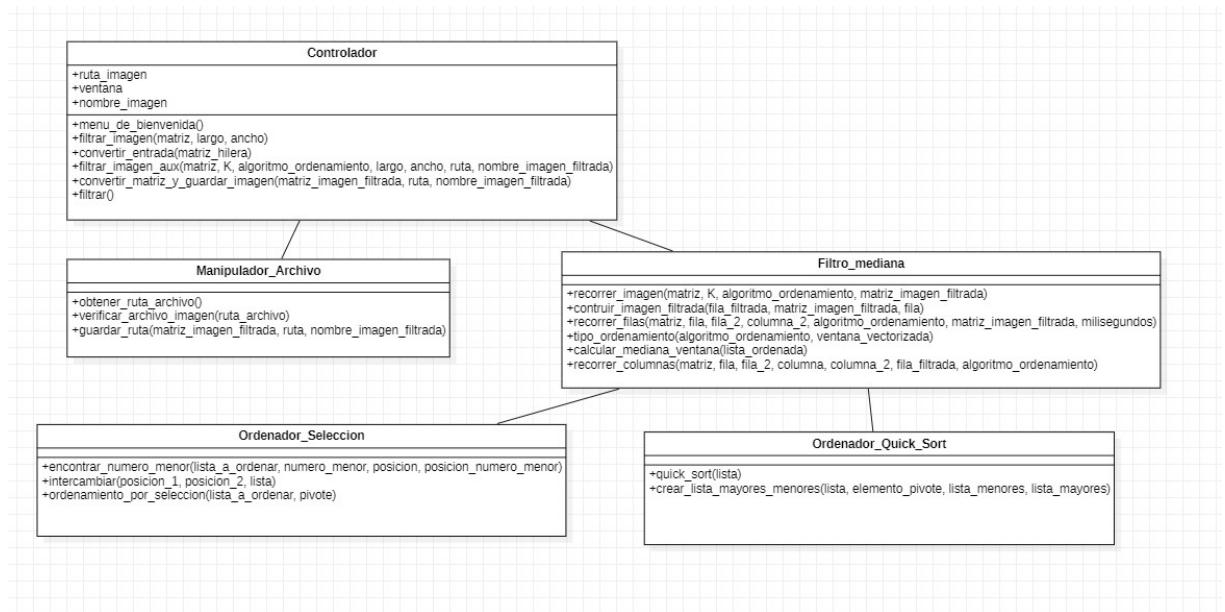


Figure 2: Diagrama de clases

Pseudocódigo de las funciones implementadas más complejas:

Función: recorrer_filas(self, matriz, fila, fila2, columna_2, algoritmo_ordenamiento, matriz_imagen_filtrada, milisegundos)

1. Se verifica si la variable fila es igual al largo de la matriz menos dos (El menos dos es porque no se cuentan los ceros agregados en el padding).
 - (a) Si la fila variable fila es igual al largo entonces se imprime el tiempo de ejecución del filtrado en milisegundos.
 - (b) Se retorna la variable que contiene la matriz de la imagen filtrada.
2. Si la fila no es igual al lado de la matriz menos dos.
 - (a) Se inicializa el valor de la variable columna con cero (La ventana va a iniciar desde el primer valor).
 - (b) Se inicializa la variable fila_filtrada con una lista vacía (En esta lista se irán guardando los pixeles ya filtrados).
 - (c) Se hace llamado a la función recorrer_columnas guardando su retorno en la variable fila_filtrada (El llamado a la funcion devolverá los valores de la fila ya filtrados).

- (d) Se hace llamado a la funcion `construir_imagen_filtrada` y su valor de retorno se guarda en la variable `matriz_imagen_filtrada` (La función llamada se encarga de ir construyendo la matriz de la imagen filtrada).
- (e) Se hace el llamado recursivo, sumando las variables `fila` y `fila_2` del parámetro.

Función: `recorrer_columnas(self, matriz, fila, fila_2, columna, columna_2, fila_filtrada, algoritmo_ordenamiento)`

1. Se verifica si la variable `columna` es igual al largo de la primera lista de la matriz menos dos (Se le resta dos para eliminar los ceros agregados en el padding).
 - (a) Si es igual al largo entonces se retorna `fila_filtrada`.
2. Si no es igual al largo de la primera lista de la matriz menos dos.
3. Se crea la ventana haciendo slicing de la matriz, a su vez haciendo slicing de las filas y las columnas. Manteniendo el tamaño de la ventana elegida por el usuario.
 - (a) Se vectorizan los valores de la ventana.
 - (b) Se hace llamado a la función `tipo_ordenamiento` guardando su valor de retorno en la variable `lista_ordenada` (La función `tipo_ordenamiento` aplica el algoritmo de ordenamiento correspondiente a la ventana vectorizada, devolviendo así los valores de la ventana ordenados de manera ascendente).
 - (c) Se hace llamado a la funcion `calcular_mediana_ventana` guardando su valor de retorno en la variable `mediana` (La función `calcular_mediana_ventana` calcula la mediana de una lista, devolviendo así el valor de la mediana, el del centro).
 - (d) Se le agrega a la variable `fila_filtrada` el valor de la mediana en forma de lista.
 - (e) Se hace el llamado recursivo, sumándole a las variables de las columnas un uno para seguir con la siguiente columna.

Función: `guardar_ruta(self, matriz_imagen_filtrada, ruta, nombre_imagen_filtrada)`

1. Se le agrega variable `nombre_imagen_filtrada` que posee una hilera “.png” (Es el formato de la imagen filtrada).
2. Se le agrega a la dirección de la ruta el slash y el nombre de la imagen filtrada (Para así crear la ruta correspondiente con la imagen).

3. Se convierte la matriz de la imagen filtrada a escala de grises, guardandolo en la variable guardar.
4. Se convierte la matriz en escala de grises a una imagen.
5. Se guarda la imagen filtrada en la ruta correspondiente.
6. Se hace un print para informar al usuario sobre la ruta donde se encuentra la imagen filtrada.

4 Implementación y pruebas

4.1 Librerías utilizadas

- Tkinter: Se utilizó para pedir por medio de una interfaz gráfica la imagen y la ruta de donde se iba a importar y exportar las imágenes.
- Os: Se utilizó como complemento junto a tkinter. Sólo se dio un uso a esta librería y fue para poder obtener el nombre del archivo de la ruta donde se encontraba la imagen.
- PIL: Se utilizó la librería PIL para dos funciones específicas, las cuales son: verificar que el archivo seleccionado fuera una imagen y a su vez convertir el array de la matriz en una imagen.
- NumPy: Librería principal del trabajo práctico, Se utilizó principalmente para poder trabajar con matrices, las funcionalidades utilizadas de esta librería fue: poder convertir listas en el tipo de dato array, crear la matriz de ceros para la imagen filtrada, con la librería se utilizó la función pad para poder bordear la imagen con ceros para así poder acceder en los bordes de la imagen.
- JSON: Se utilizó la librería con el fin de poder convertir las matrices ingresadas manualmente, de tipo string, a una lista verdadera.
- time: Se usó esta librería para poder crear el conteo del filtrado de imágenes en milisegundos.
- sys: se utilizó la función setrecursionlimit para poder aumentar el límite de la pila de llamados.

4.2 Prueba 1 para el filtro con matrices pequeñas

1. El sistema le pregunta al usuario si desea ingresar la imagen manualmente o si desea seleccionar la imagen.
 - (a) El usuario ingresa 1 (Manualmente)

2. El sistema le pide al usuario que ingrese la matriz que desea filtrar.
 - (a) El usuario ingresa: $[[221.0, 222, 250, 251, 223, 249], [223.0, 220, 250, 251, 225, 242], [221.0, 0, 250, 251, 221, 249], [221.0, 222, 255, 251, 0, 249], [220.0, 219, 250, 251, 221, 249], [221.0, 222, 250, 251, 223, 249]]$
3. El sistema le pide seleccionar la ruta donde desea guardar la imagen filtrada:
 - (a) El usuario selecciona la ruta: C:/Users/dynxl/OneDrive/Imágenes.
4. El sistema le pide al usuario un nombre con el que desea guardar la imagen filtrada.
 - (a) El usuario ingresa “imagen pequeña”.
5. El sistema le pide al usuario el tamaño de la ventana.
 - (a) El usuario ingresa: 3
6. El sistema le pregunta al usuario cuál algoritmo de ordenamiento le gustaría usar, desplegando las opciones.
 - (a) El usuario ingresa 1 (Selección)
7. El sistema realiza el proceso de filtrado, ordenamiento (con el algoritmo de selección), cálculo de la mediana y construcción de la imagen filtrada.
8. El sistema muestra el tiempo de ejecución del filtrado en milisegundos, que es 1652654087110.
9. El sistema le indica que se guardó en la ruta: C:/Users/dynxl/OneDrive/Imágenes/imagen pequeña.png

Resultado esperado de la prueba: Se espera ver en el directorio de las imágenes del usuario la imagen filtrada de la matriz pequeña ingresada. En la figura 3 se puede observar el resultado del filtrado de la prueba 1.

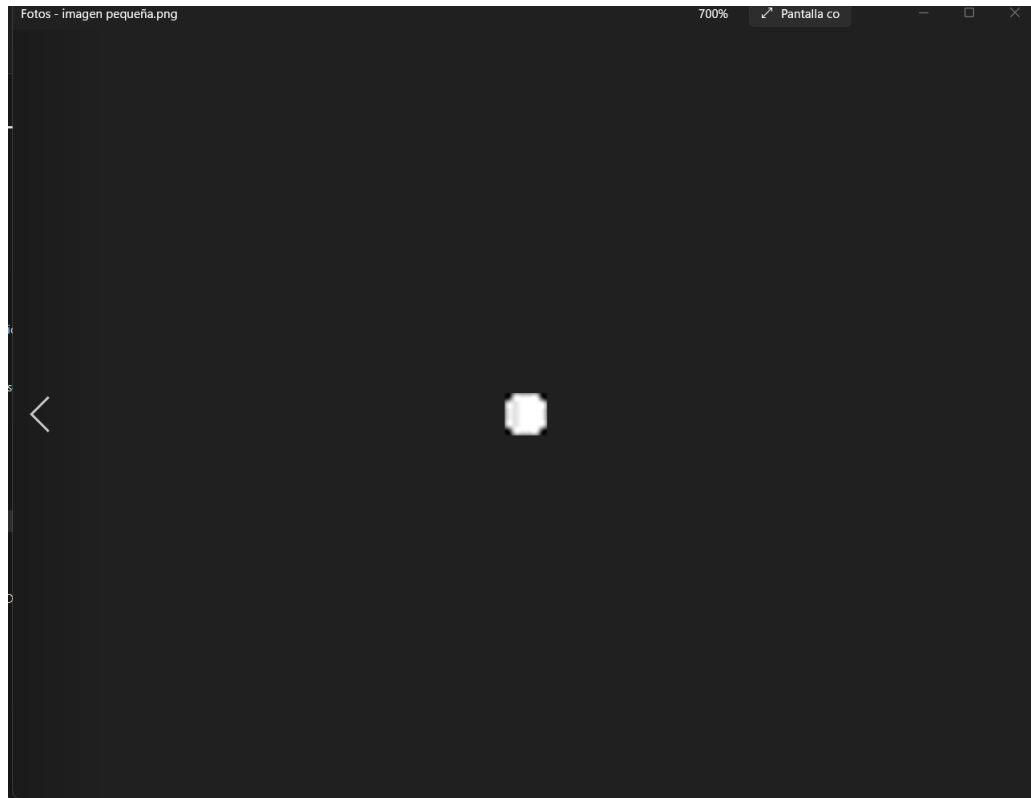


Figura 3: Prueba 1

4.3 Prueba 2 para el filtro con matrices pequeñas

1. El sistema le pregunta al usuario si desea ingresar la imagen manualmente o si desea seleccionar la imagen.
 - (a) El usuario ingresa 1 (Manualmente)
2. El sistema le pide al usuario que ingrese la matriz que desea filtrar.
 - (a) El usuario ingresa: $[[221.0, 222, 250, 251, 223, 249], [223.0, 220, 250, 251, 225, 242], [221.0, 0, 250, 251, 221, 249], [221.0, 222, 255, 251, 0, 249], [220.0, 219, 250, 251, 221, 249], [221.0, 222, 250, 251, 223, 249]]$
3. El sistema le pide seleccionar la ruta donde desea guardar la imagen filtrada:
 - (a) El usuario selecciona la ruta: C:/Users/dynxl/OneDrive/Imágenes.
4. El sistema le pide al usuario un nombre con el que desea guardar la imagen filtrada.

- (a) El usuario ingresa “imagen_pequenia”.
- 5. El sistema le pide al usuario el tamaño de la ventana.
 - (a) El usuario ingresa: 5
- 6. El sistema le pregunta al usuario cuál algoritmo de ordenamiento le gustaría usar, desplegando las opciones.
 - (a) El usuario ingresa 2 (Quick Sort)
- 7. El sistema realiza el proceso de filtrado, ordenamiento (con el algoritmo de quick sort), cálculo de la mediana y construcción de la imagen filtrada.
- 8. El sistema muestra el tiempo de ejecución del filtrado en milisegundos, que es 1652654877847.
- 9. El sistema le indica que se guardó en la ruta: C:/Users/dynxl/OneDrive/Imágenes/imagen_pequenia.png

Resultado esperado de la prueba: Se espera ver en el directorio de las imágenes del usuario la imagen filtrada de la matriz pequeña ingresada. En la figura 4 se puede observar el resultado del filtrado de la prueba 2.

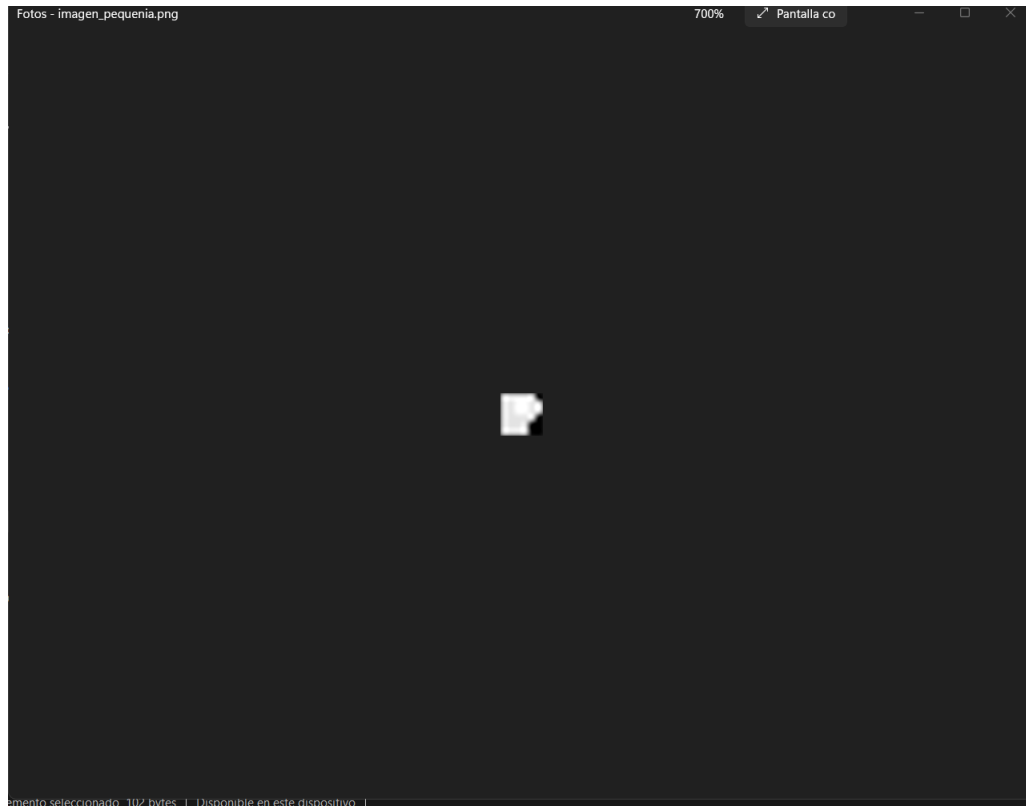


Figura 4: Prueba 2

4.4 Pruebas 3 para el filtro con matrices pequeñas

1. El sistema le pregunta al usuario si desea ingresar la imagen manualmente o si desea seleccionar la imagen.
 - (a) El usuario ingresa 1 (Manualmente)
2. El sistema le pide al usuario que ingrese la matriz que desea filtrar.
 - (a) El usuario ingresa: $\begin{bmatrix} 221.0, & 222, & 250, & 251, & 223, & 249 \\ 223.0, & 220, & 250, & 251, & 225, & 242 \\ 221.0, & 0, & 250, & 251, & 221, & 249 \\ 221.0, & 222, & 255, & 251, & 0, & 249 \\ 220.0, & 219, & 250, & 251, & 221, & 249 \\ 221.0, & 222, & 250, & 251, & 223, & 249 \end{bmatrix}$
3. El sistema le pide seleccionar la ruta donde desea guardar la imagen filtrada:
 - (a) El usuario selecciona la ruta: C:/Users/dynxl/OneDrive/Imágenes.
4. El sistema le pide al usuario un nombre con el que desea guardar la imagen filtrada.

- (a) El usuario ingresa “imagen filtrada”.
- 5. El sistema le pide al usuario el tamaño de la ventana.
 - (a) El usuario ingresa: 3
- 6. El sistema le pregunta al usuario cuál algoritmo de ordenamiento le gustaría usar, desplegando las opciones.
 - (a) El usuario ingresa 2 (Quick Sort)
- 7. El sistema realiza el proceso de filtrado, ordenamiento (con el algoritmo de quick sort), cálculo de la mediana y construcción de la imagen filtrada.
- 8. El sistema muestra el tiempo de ejecución del filtrado en milisegundos, que es 1652655044482.
- 9. El sistema le indica que se guardó en la ruta: C:/Users/dynxl/OneDrive/Imágenes/imagen filtrada.png

Resultado esperado de la prueba: Se espera ver en el directorio de las imágenes del usuario la imagen filtrada de la matriz pequeña ingresada. En la figura 5 se puede observar el resultado del filtrado de la prueba 3.

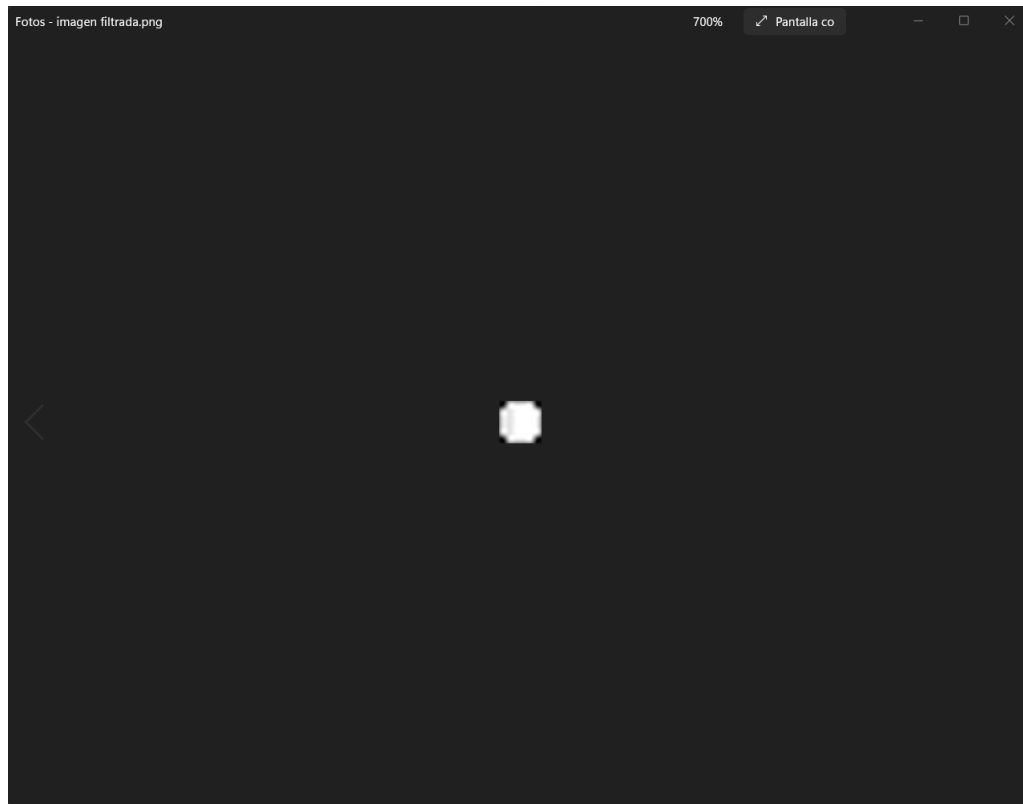


Figura 5: Prueba 3

4.5 Prueba 4 de filtrado usando la imagen manNoisy.PNG

1. El sistema le pregunta al usuario si desea ingresar la imagen manualmente o si desea seleccionar la imagen.
 - (a) El usuario ingresa 2 (Seleccionando la imagen)
2. El sistema le pide seleccionar la imagen y a la vez la ruta de la imagen que desea filtrar en la ventana de tkinter.
 - (a) El usuario selecciona la imagen manNoisy.PNG
3. El sistema le notifica que el nombre de la imagen seleccionada es: manNoisy.PNG
 - (a) El sistema le informa que la imagen se cargó exitosamente.
4. El sistema le informa que la imagen se cargó exitosamente.
5. El sistema le pide seleccionar la ruta donde desea guardar la imagen filtrada:

- (a) El usuario selecciona la ruta: C:/Users/dynxl/OneDrive/Imágenes.
- 6. El sistema le pide al usuario un nombre con el que desea guardar la imagen filtrada.
 - (a) El usuario ingresa “man noisy filtrada”.
- 7. El sistema le pide al usuario el tamaño de la ventana.
 - (a) El usuario ingresa: 3
- 8. El sistema le pregunta al usuario cuál algoritmo de ordenamiento le gustaría usar, desplegando las opciones.
 - (a) El usuario ingresa 1 (Selección)
- 9. El sistema realiza el proceso de filtrado, ordenamiento (con el algoritmo de selección), cálculo de la mediana y construcción de la imagen filtrada.
- 10. El sistema muestra el tiempo de ejecución del filtrado en milisegundos, que es 1652656247577.
- 11. El sistema le indica que se guardó en la ruta: C:/Users/dynxl/OneDrive/Imágenes/man noisy filtrada.png

Resultado esperado de la prueba: Se espera ver en el directorio de las imágenes del usuario la imagen filtrada de manNoisy.PNG con menor ruido. En la figura 6 se puede observar el resultado del filtrado de la prueba 4.



Figura 6: Prueba 4

4.6 Prueba 5 de filtrado usando la imagen manNoisy.PNG

1. El sistema le pregunta al usuario si desea ingresar la imagen manualmente o si desea seleccionar la imagen.
 - (a) El usuario ingresa 2 (Seleccionando la imagen)
2. El sistema le pide seleccionar la imagen y a la vez la ruta de la imagen que desea filtrar en la ventana de tkinter.
 - (a) El usuario selecciona la imagen manNoisy.PNG
3. El sistema le notifica que el nombre de la imagen seleccionada es: manNoisy.PNG
4. El sistema le informa que la imagen se cargó exitosamente.
5. El sistema le pide seleccionar la ruta donde desea guardar la imagen filtrada:
 - (a) El usuario selecciona la ruta: C:/Users/dynxl/OneDrive/Imágenes.

6. El sistema le pide al usuario un nombre con el que desea guardar la imagen filtrada.
 - (a) El usuario ingresa “man noisy quick sort”.
7. El sistema le pide al usuario el tamaño de la ventana.
 - (a) El usuario ingresa: 3
8. El sistema le pregunta al usuario cuál algoritmo de ordenamiento le gustaría usar, desplegando las opciones.
 - (a) El usuario ingresa 2 (Quick Sort)
9. El sistema realiza el proceso de filtrado, ordenamiento (con el algoritmo de quick sort), cálculo de la mediana y construcción de la imagen filtrada.
10. El sistema muestra el tiempo de ejecución del filtrado en milisegundos, que es 1652656736192.
11. El sistema le indica que se guardó en la ruta: C:/Users/dynxl/OneDrive/Imágenes/man noisy quick sort.png

Resultado esperado de la prueba: Se espera ver en el directorio de las imágenes del usuario la imagen filtrada de manNoisy.PNG con menor ruido. En la figura 7 se puede observar el resultado del filtrado de la prueba 5.



Figura 7: Prueba 5

4.7 Prueba 6 de filtrado usando la imagen *mujerRuido.PNG*

1. El sistema le pregunta al usuario si desea ingresar la imagen manualmente o si desea seleccionar la imagen.
 - (a) El usuario ingresa 2 (Seleccionando la imagen)
2. El sistema le pide seleccionar la imagen y a la vez la ruta de la imagen que desea filtrar en la ventana de tkinter.
 - (a) El usuario selecciona la imagen mujerRuido.PNG
3. El sistema le notifica que el nombre de la imagen seleccionada es: mujerRuido.PNG
4. El sistema le informa que la imagen se cargó exitosamente.
5. El sistema le pide seleccionar la ruta donde desea guardar la imagen filtrada:
 - (a) El usuario selecciona la ruta: C:/Users/dynxl/OneDrive/Imágenes.

6. El sistema le pide al usuario un nombre con el que desea guardar la imagen filtrada.
 - (a) El usuario ingresa “mujerRuido filtrado”.
7. El sistema le pide al usuario el tamaño de la ventana.
 - (a) El usuario ingresa: 5
8. El sistema le pregunta al usuario cuál algoritmo de ordenamiento le gustaría usar, desplegando las opciones.
 - (a) El usuario ingresa 2 (Quick Sort)
9. El sistema realiza el proceso de filtrado, ordenamiento (con el algoritmo de quick sort), cálculo de la mediana y construcción de la imagen filtrada.
10. El sistema muestra el tiempo de ejecución del filtrado en milisegundos, que es 1652671625990.
11. El sistema le indica que se guardó en la ruta: C:/Users/dynxl/OneDrive/Imágenes/mujerRuido filtrado.png

Resultado esperado de la prueba: Se espera ver en el directorio de las imágenes del usuario la imagen filtrada de mujerRuido.PNG con menor ruido. En la figura 8 se puede observar el resultado del filtrado de la prueba 6.



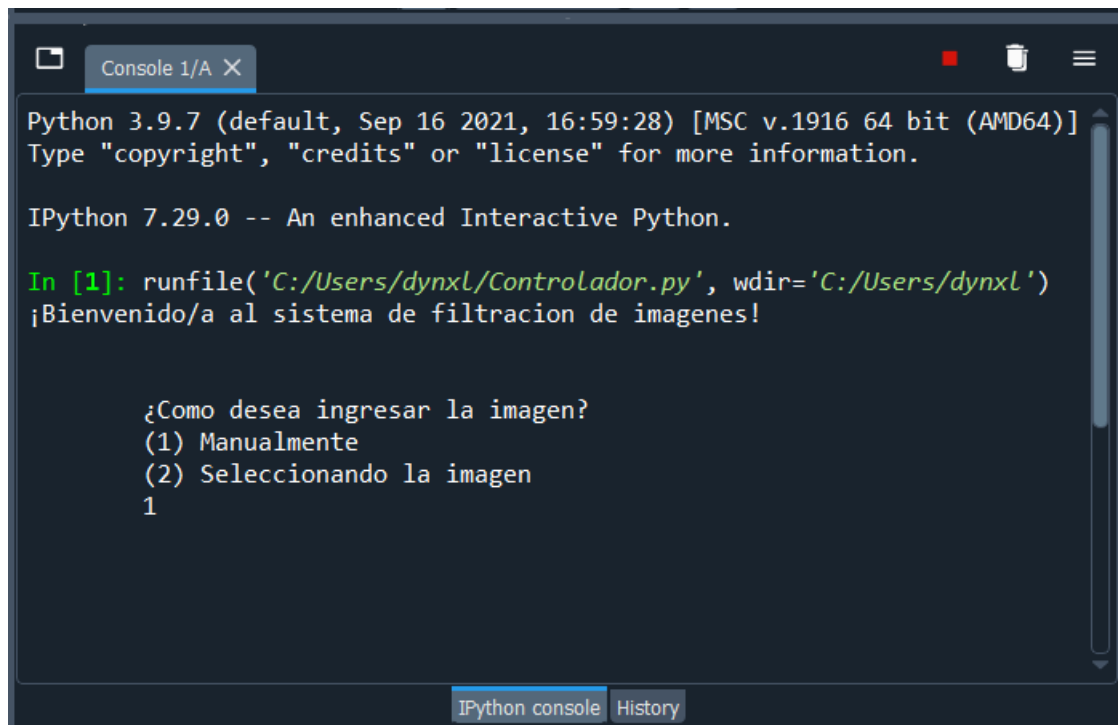
Figura 8: Prueba 6

4.8 Comparación del tiempo de ejecución

Se puede observar por las pruebas realizadas, que el tarda más milisegundos el proceso de filtrado cuando las imágenes son más grandes y con ventanas más grandes. Además de que el tipo de algoritmo toma un papel muy importante, esto lo podemos ver por ejemplo en la prueba 1 y prueba 3, ambas filtran las mismas matrices y con el mismo tamaño de la ventana, pero con diferentes algoritmos de ordenamiento y se puede notar una diferencia entre milisegundos de una de la otra.

4.9 Manual de usuario

Ingresa imagen manualmente:



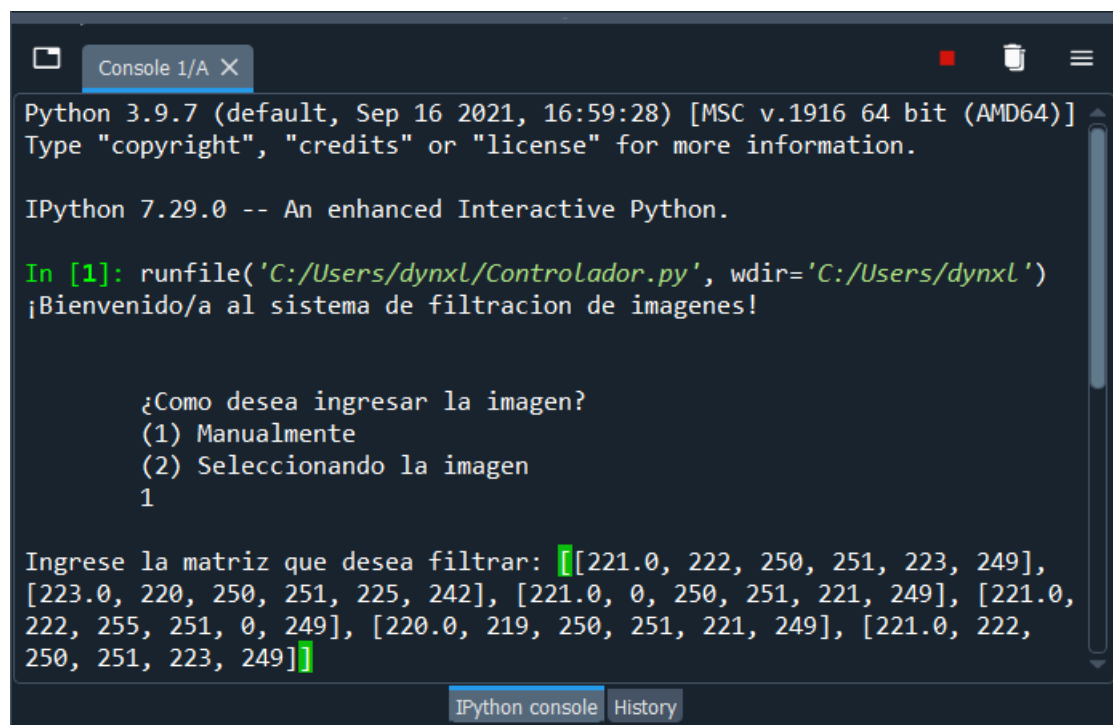
```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/dynxl/Controlador.py', wdir='C:/Users/dynxl')
¡Bienvenido/a al sistema de filtracion de imagenes!

      ¿Como desea ingresar la imagen?
      (1) Manualmente
      (2) Seleccionando la imagen
      1
```

Figura 9: Seleccionar tipo de entrada



```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/dynxl/Controlador.py', wdir='C:/Users/dynxl')
¡Bienvenido/a al sistema de filtracion de imagenes!

      ¿Como desea ingresar la imagen?
      (1) Manualmente
      (2) Seleccionando la imagen
      1

Ingrese la matriz que desea filtrar: [[221.0, 222, 250, 251, 223, 249],
[223.0, 220, 250, 251, 225, 242], [221.0, 0, 250, 251, 221, 249], [221.0,
222, 255, 251, 0, 249], [220.0, 219, 250, 251, 221, 249], [221.0, 222,
250, 251, 223, 249]]
```

Figura 10: Ingresar matriz

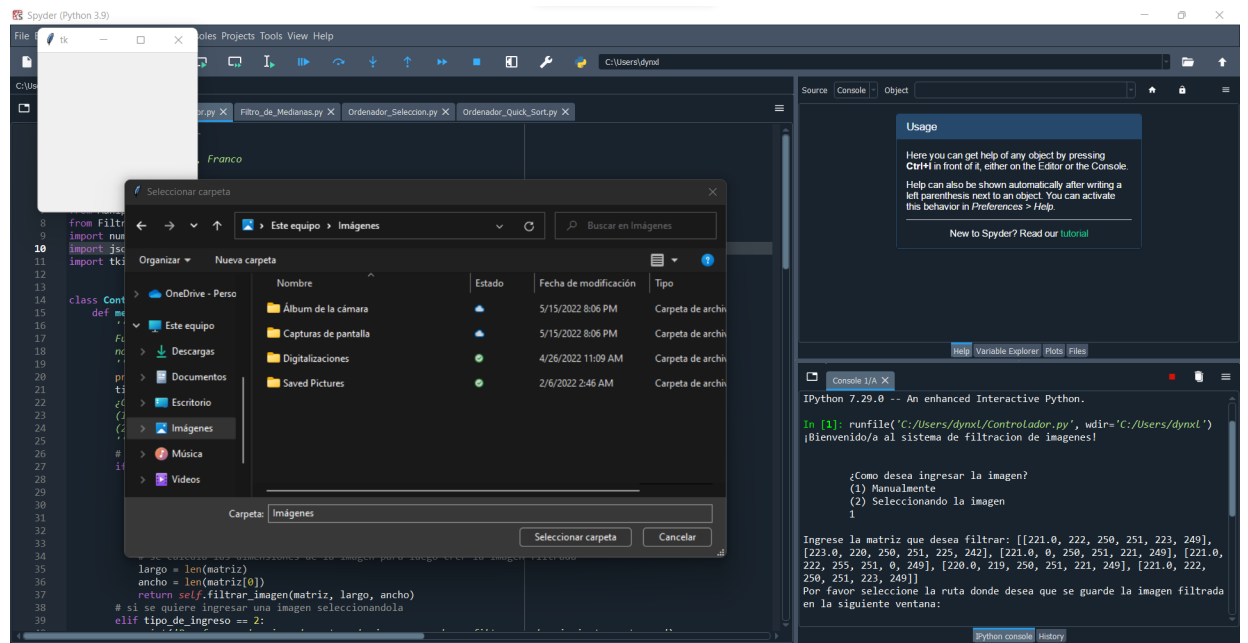
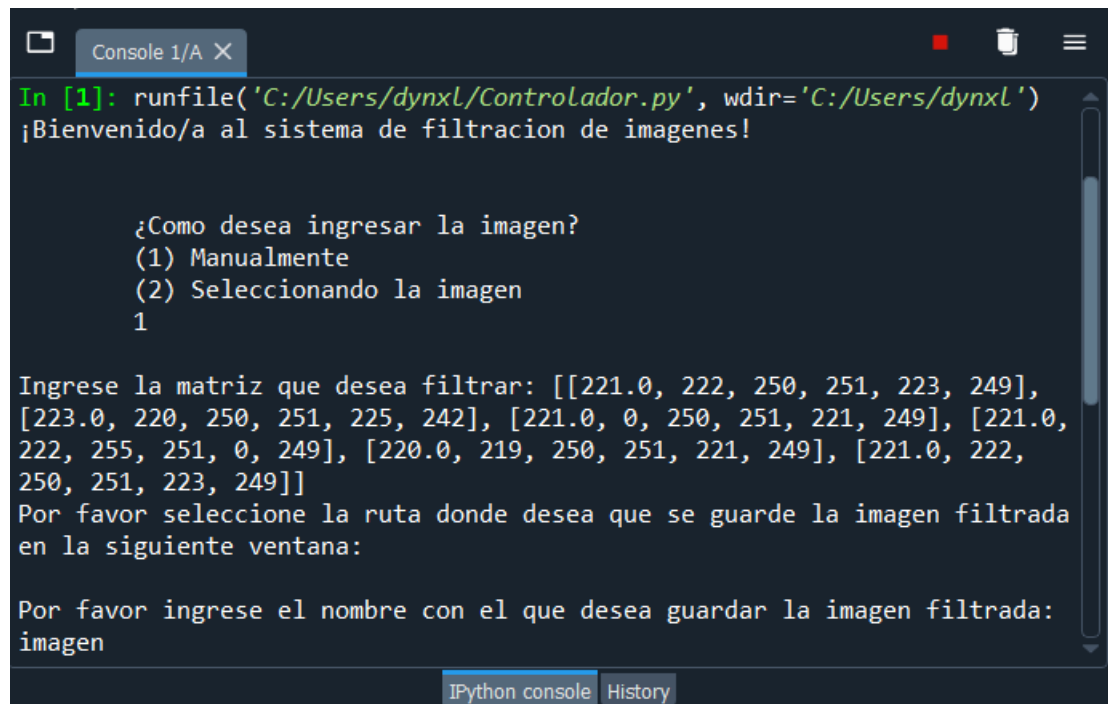


Figura 11: Se selecciona la ruta donde se va a guardar la imagen filtrada



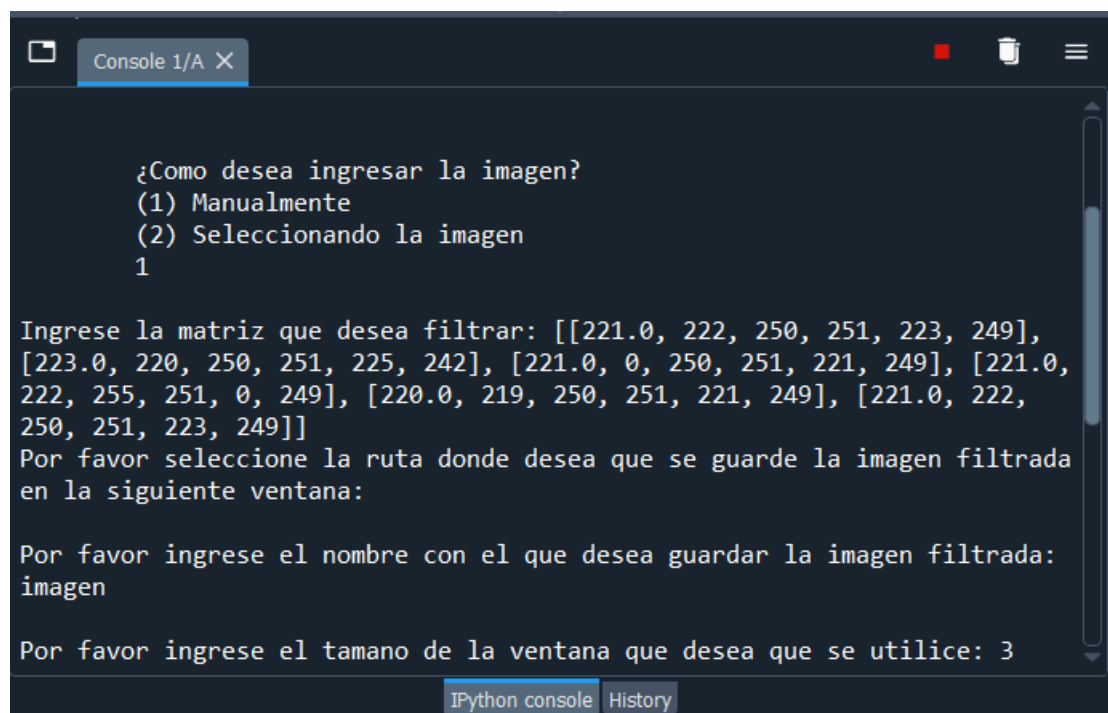
```
Console 1/A X
In [1]: runfile('C:/Users/dynxl/Controlador.py', wdir='C:/Users/dynxl')
¡Bienvenido/a al sistema de filtracion de imagenes!

    ¿Como desea ingresar la imagen?
    (1) Manualmente
    (2) Seleccionando la imagen
    1

Ingrese la matriz que desea filtrar: [[221.0, 222, 250, 251, 223, 249],
[223.0, 220, 250, 251, 225, 242], [221.0, 0, 250, 251, 221, 249], [221.0,
222, 255, 251, 0, 249], [220.0, 219, 250, 251, 221, 249], [221.0, 222,
250, 251, 223, 249]]
Por favor seleccione la ruta donde desea que se guarde la imagen filtrada
en la siguiente ventana:

Por favor ingrese el nombre con el que desea guardar la imagen filtrada:
imagen
```

Figura 12: Se ingresa el nombre que queremos que se guarde la imagen filtrada



```
Console 1/A X

¿Como desea ingresar la imagen?
(1) Manualmente
(2) Seleccionando la imagen
1

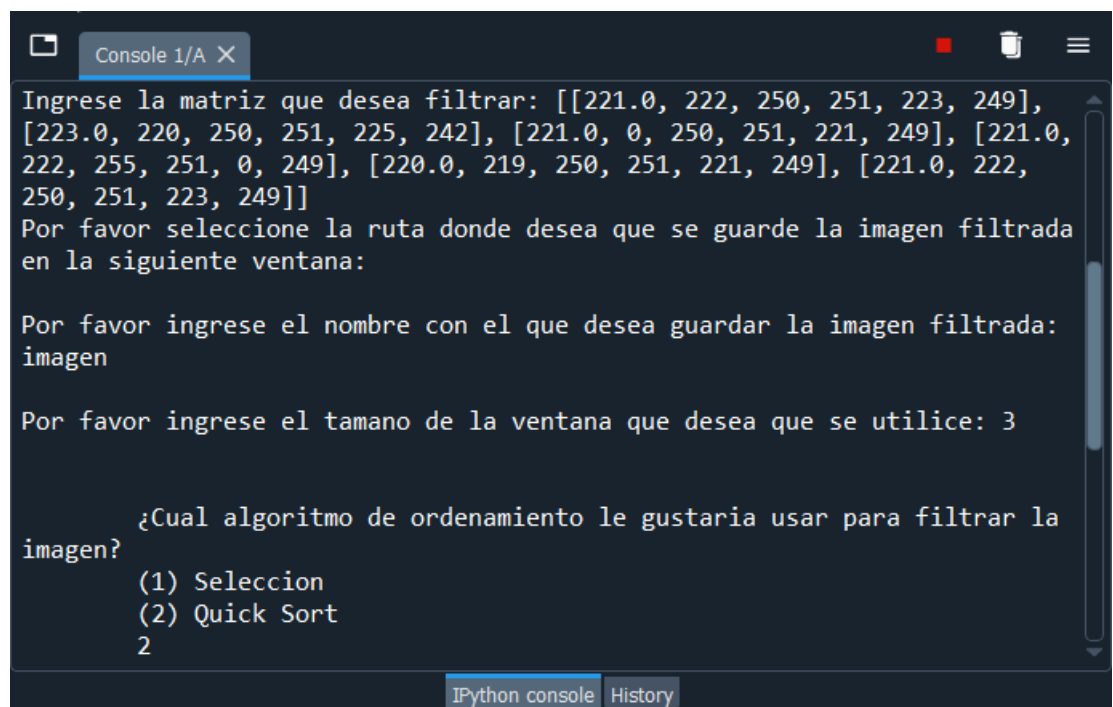
Ingrese la matriz que desea filtrar: [[221.0, 222, 250, 251, 223, 249],
[223.0, 220, 250, 251, 225, 242], [221.0, 0, 250, 251, 221, 249], [221.0,
222, 255, 251, 0, 249], [220.0, 219, 250, 251, 221, 249], [221.0, 222,
250, 251, 223, 249]]
Por favor seleccione la ruta donde desea que se guarde la imagen filtrada
en la siguiente ventana:

Por favor ingrese el nombre con el que desea guardar la imagen filtrada:
imagen

Por favor ingrese el tamaño de la ventana que desea que se utilice: 3

IPython console History
```

Figura 13: Se ingresa el tamaño de la ventana



```
Console 1/A X
Ingrese la matriz que desea filtrar: [[221.0, 222, 250, 251, 223, 249],
[223.0, 220, 250, 251, 225, 242], [221.0, 0, 250, 251, 221, 249], [221.0,
222, 255, 251, 0, 249], [220.0, 219, 250, 251, 221, 249], [221.0, 222,
250, 251, 223, 249]]
Por favor seleccione la ruta donde desea que se guarde la imagen filtrada
en la siguiente ventana:

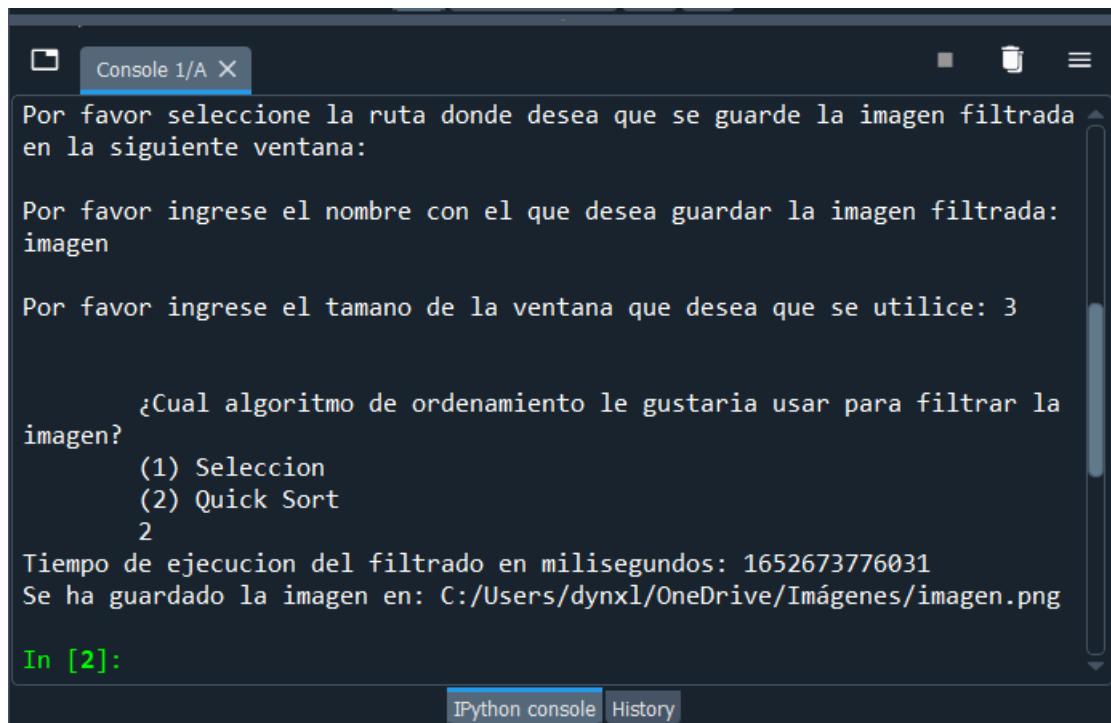
Por favor ingrese el nombre con el que desea guardar la imagen filtrada:
imagen

Por favor ingrese el tamaño de la ventana que desea que se utilice: 3

¿Cual algoritmo de ordenamiento le gustaria usar para filtrar la
imagen?
(1) Selecccion
(2) Quick Sort
2
```

IPython console History

Figura 14: Se selecciona el tipo de algoritmo que queremos utilizar



```
Console 1/A X
Por favor seleccione la ruta donde desea que se guarde la imagen filtrada
en la siguiente ventana:

Por favor ingrese el nombre con el que desea guardar la imagen filtrada:
imagen

Por favor ingrese el tamaño de la ventana que desea que se utilice: 3

¿Cual algoritmo de ordenamiento le gustaria usar para filtrar la
imagen?
(1) Seleccion
(2) Quick Sort
2
Tiempo de ejecucion del filtrado en milisegundos: 1652673776031
Se ha guardado la imagen en: C:/Users/dynxl/OneDrive/Imágenes/imagen.png

In [2]:
```

Figura 15: El sistema notifica de los milisegundos que duró el filtrado y de la ubicación del archivo

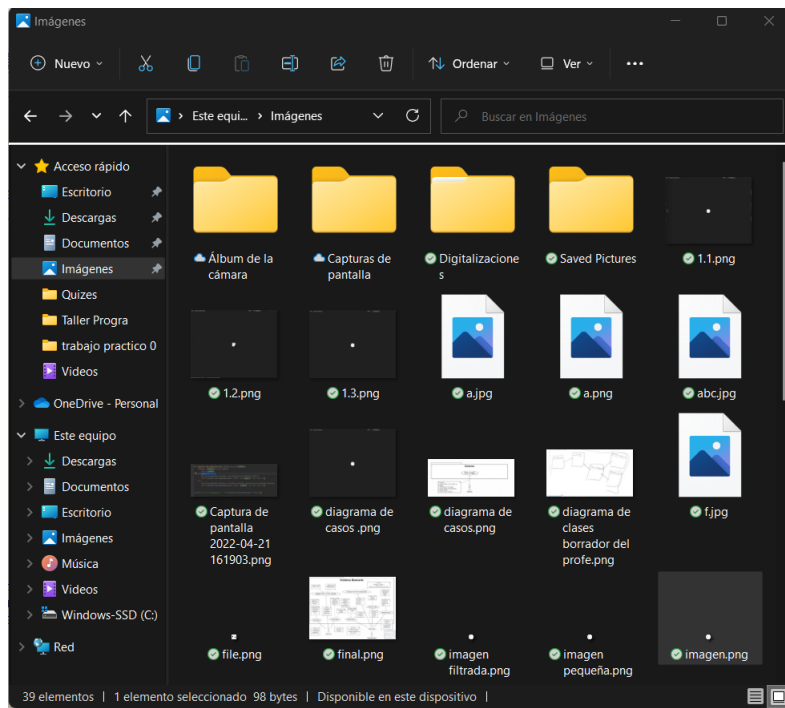
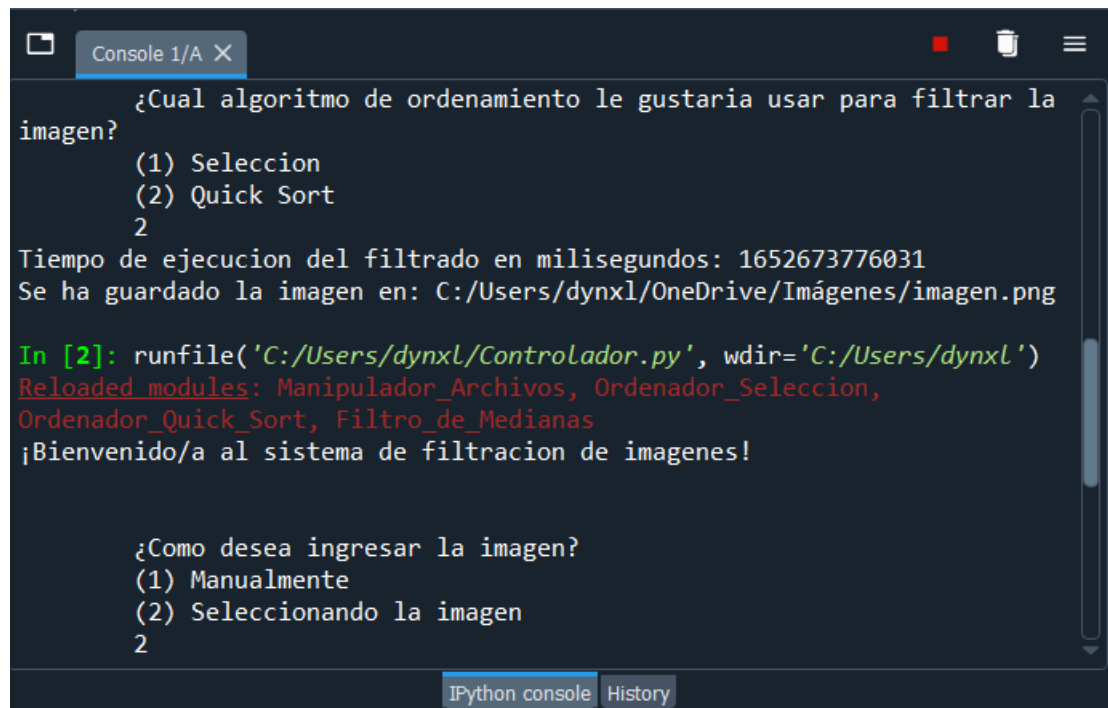


Figura 16: Buscamos la imagen filtrada en nuestra computadora

Seleccionar imagen



```
Console 1/A X
¿Cual algoritmo de ordenamiento le gustaria usar para filtrar la
imagen?
(1) Selecccion
(2) Quick Sort
2
Tiempo de ejecucion del filtrado en milisegundos: 1652673776031
Se ha guardado la imagen en: C:/Users/dynxl/OneDrive/Imágenes/imagen.png

In [2]: runfile('C:/Users/dynxl/Controlador.py', wdir='C:/Users/dynxl')
Reloaded modules: Manipulador_Archivos, Ordenador_Seleccion,
Ordenador_Quick_Sort, Filtro_de_Medias
¡Bienvenido/a al sistema de filtracion de imagenes!

¿Como desea ingresar la imagen?
(1) Manualmente
(2) Seleccionando la imagen
2

IPython console History
```

Figura 17: Seleccionar tipo de entrada

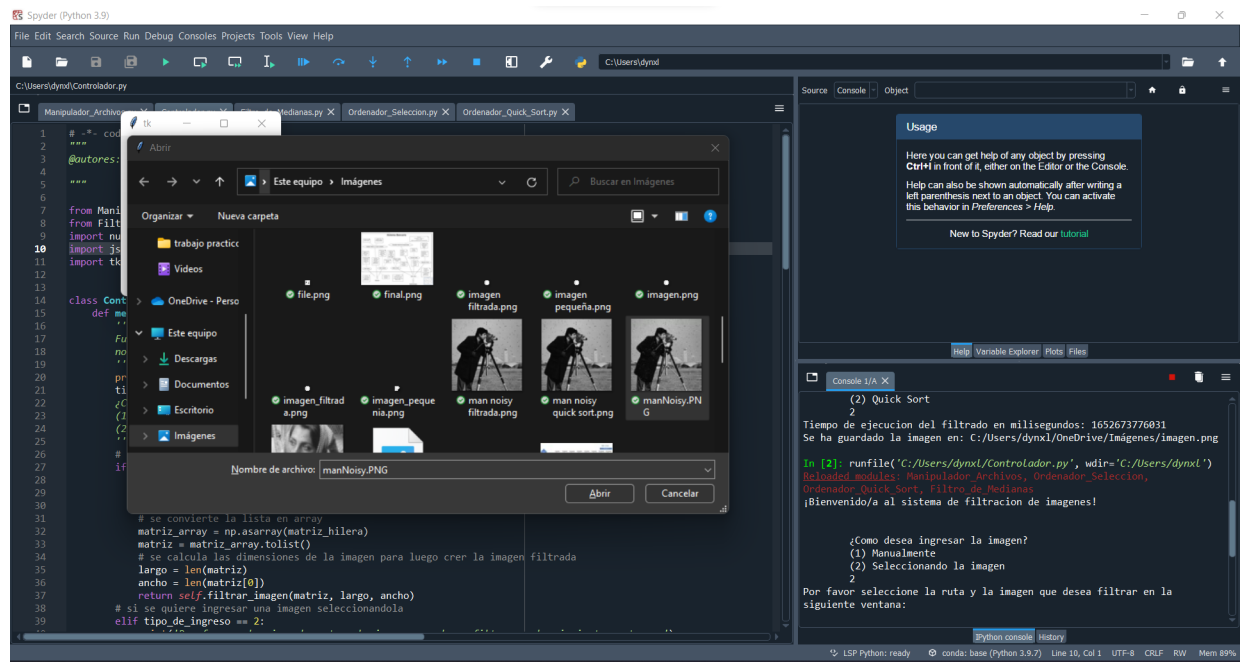


Figura 18: Se selecciona la imagen que queremos filtrar

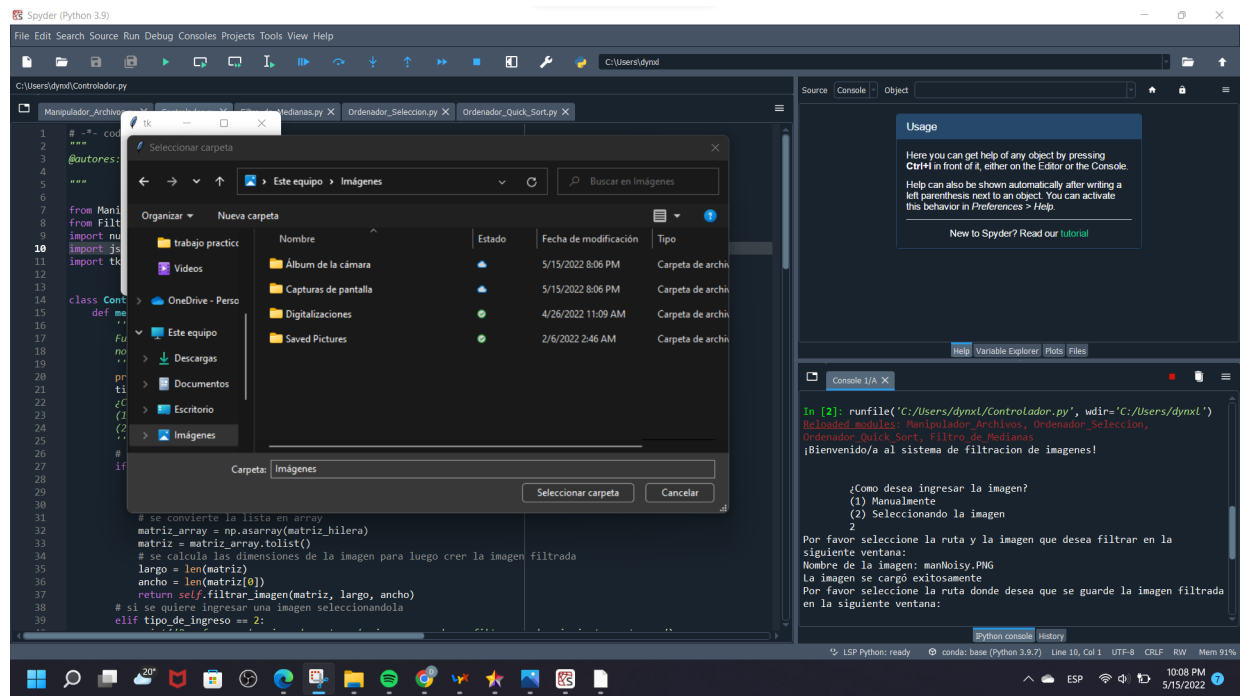
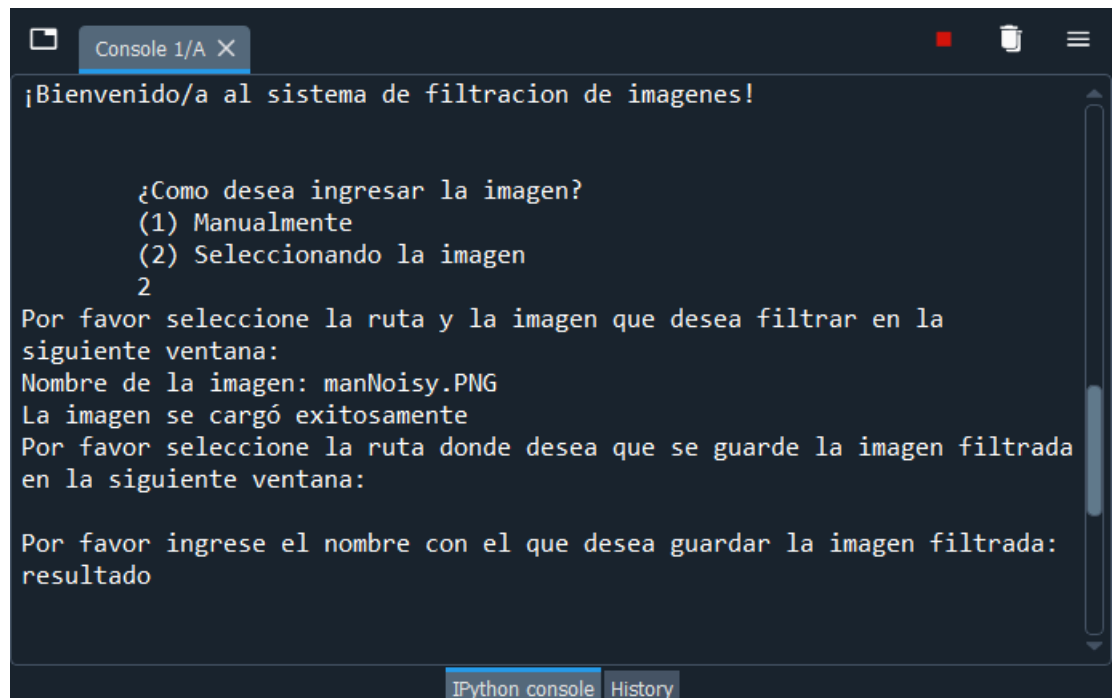


Figura 19: Seleccionamos la ruta donde queremos que se guarde la imagen filtrada

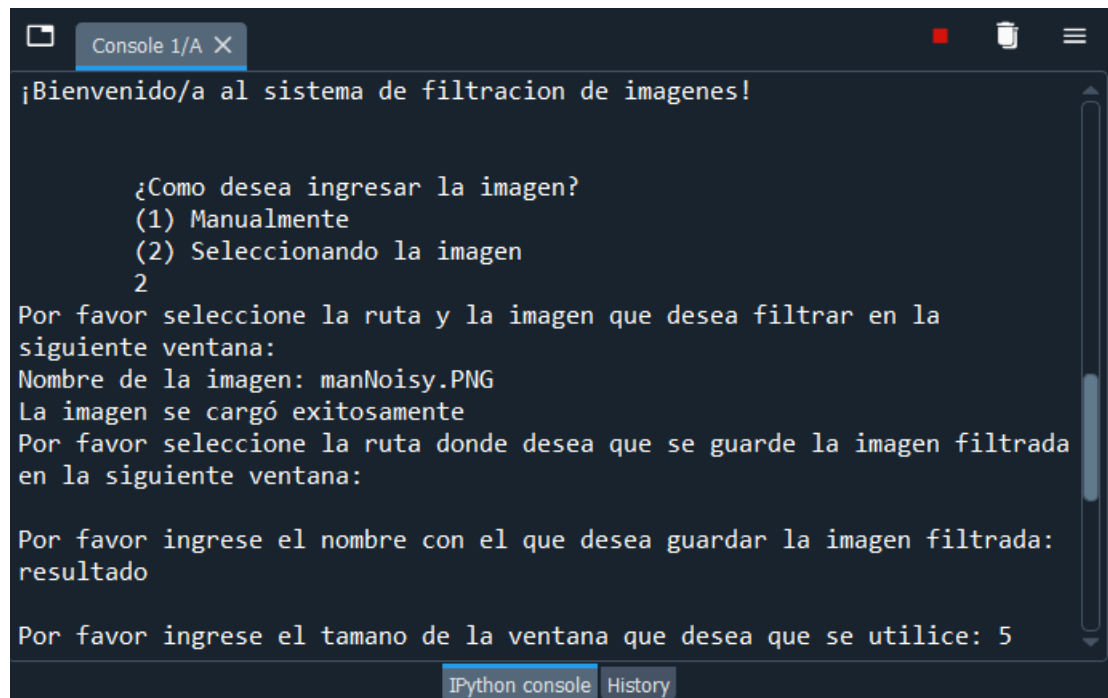


```
Console 1/A X
¡Bienvenido/a al sistema de filtracion de imagenes!

    ¿Como desea ingresar la imagen?
    (1) Manualmente
    (2) Seleccionando la imagen
    2
Por favor seleccione la ruta y la imagen que desea filtrar en la
siguiente ventana:
Nombre de la imagen: manNoisy.PNG
La imagen se cargó exitosamente
Por favor seleccione la ruta donde desea que se guarde la imagen filtrada
en la siguiente ventana:

Por favor ingrese el nombre con el que desea guardar la imagen filtrada:
resultado
```

Figura 20: Ingresamos el nombre con el que queremos que se guarde la imagen filtrada



```
Console 1/A X
¡Bienvenido/a al sistema de filtracion de imagenes!

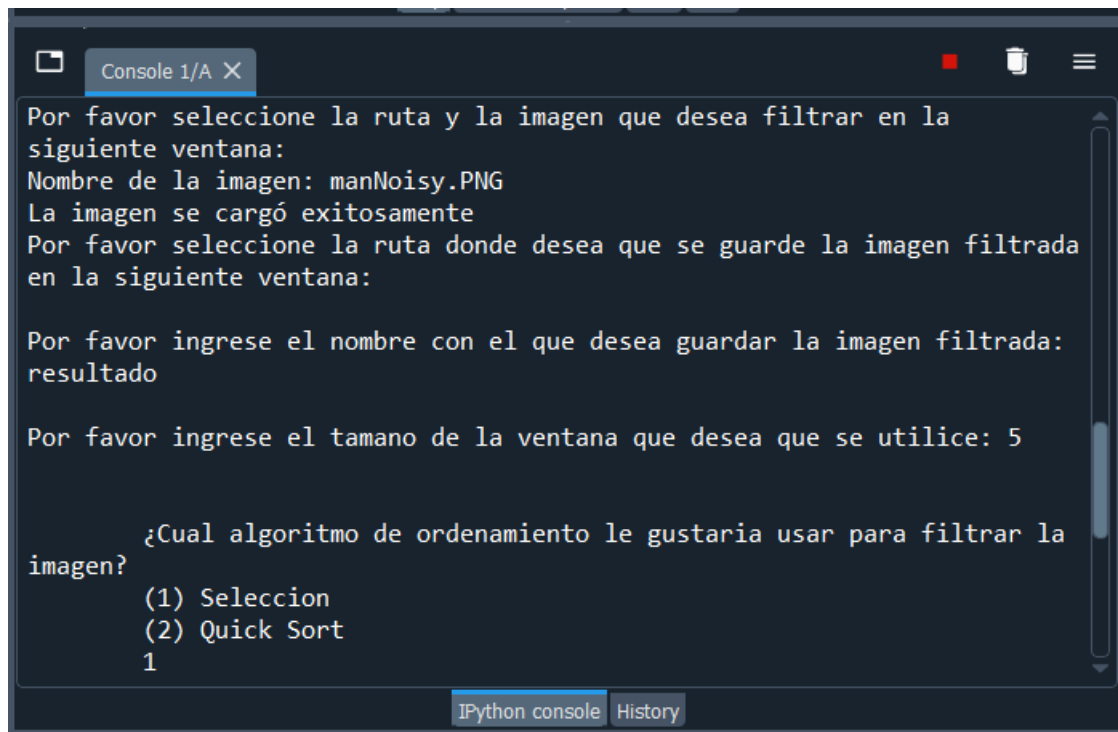
    ¿Como desea ingresar la imagen?
    (1) Manualmente
    (2) Seleccionando la imagen
    2
Por favor seleccione la ruta y la imagen que desea filtrar en la
siguiente ventana:
Nombre de la imagen: manNoisy.PNG
La imagen se cargó exitosamente
Por favor seleccione la ruta donde desea que se guarde la imagen filtrada
en la siguiente ventana:

Por favor ingrese el nombre con el que desea guardar la imagen filtrada:
resultado

Por favor ingrese el tamaño de la ventana que desea que se utilice: 5

IPython console History
```

Figura 21: Ingresamos el tamaño de la ventana



```
Console 1/A X
Por favor seleccione la ruta y la imagen que desea filtrar en la
siguiente ventana:
Nombre de la imagen: manNoisy.PNG
La imagen se cargó exitosamente
Por favor seleccione la ruta donde desea que se guarde la imagen filtrada
en la siguiente ventana:

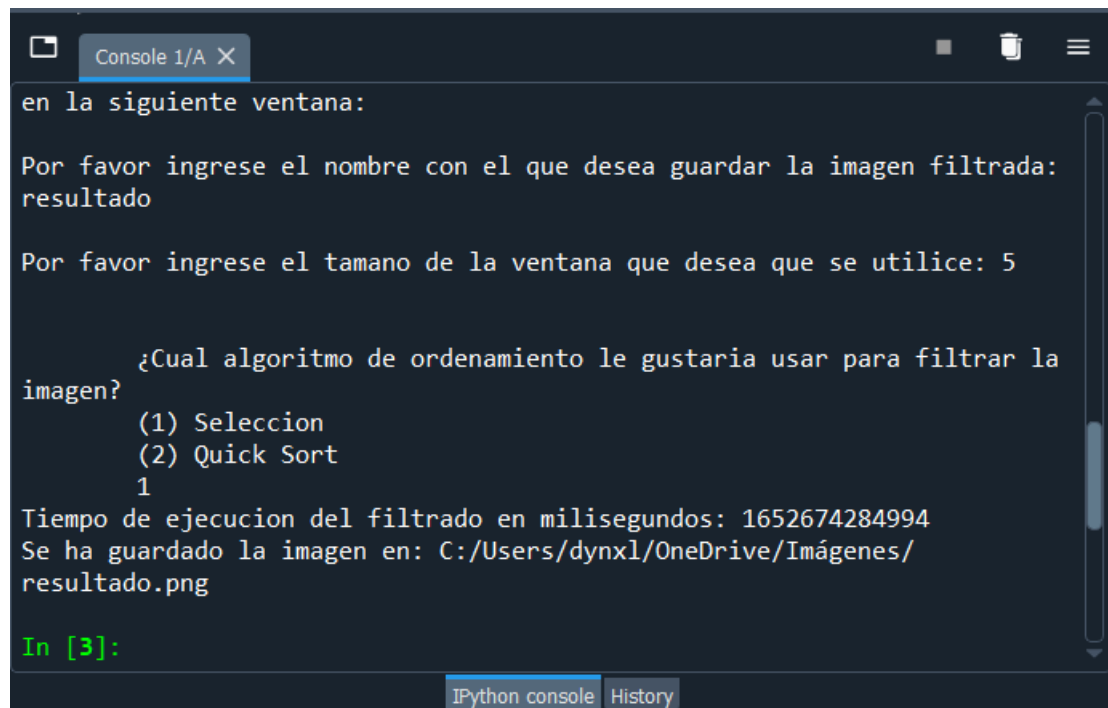
Por favor ingrese el nombre con el que desea guardar la imagen filtrada:
resultado

Por favor ingrese el tamaño de la ventana que desea que se utilice: 5

¿Cual algoritmo de ordenamiento le gustaria usar para filtrar la
imagen?
(1) Selecccion
(2) Quick Sort
1

IPython console History
```

Figura 22: Elegimos el tipo de algoritmo de ordenamiento



```
en la siguiente ventana:

Por favor ingrese el nombre con el que desea guardar la imagen filtrada:
resultado

Por favor ingrese el tamaño de la ventana que desea que se utilice: 5

    ¿Cual algoritmo de ordenamiento le gustaria usar para filtrar la
imagen?
    (1) Selecccion
    (2) Quick Sort
    1
Tiempo de ejecucion del filtrado en milisegundos: 1652674284994
Se ha guardado la imagen en: C:/Users/dynxl/OneDrive/Imágenes/
resultado.png

In [3]:
```

Figura 23: El sistema muestra el tiempo en milisegundos que tardó el filtrado y la ruta donde se encuentra la imagen filtrada

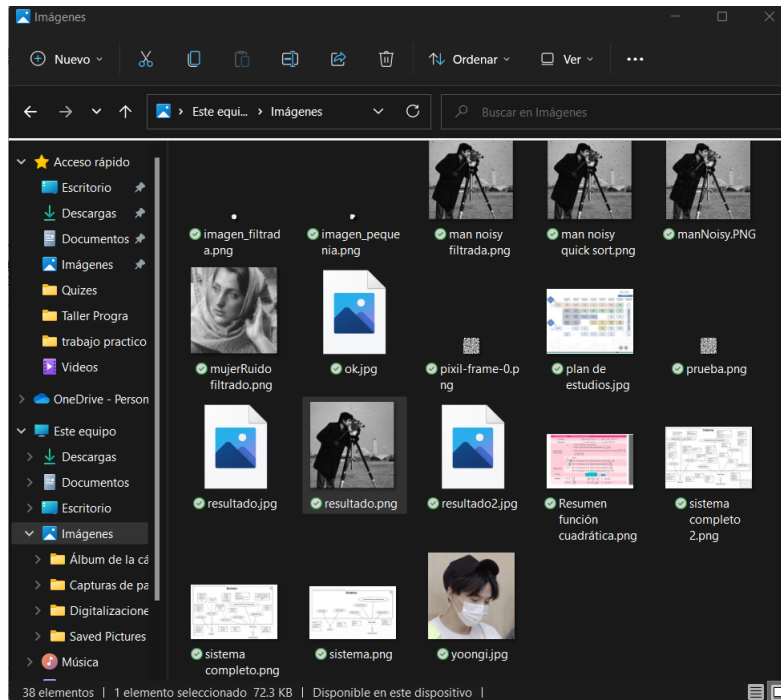


Figura 24: Y encontraremos en la ruta seleccionada, la imagen filtrada.

5 Conclusiones

Para concluir este trabajo práctico, es importante recalcar el aprendizaje dentro de este proceso, ya que nos abre un mundo de oportunidades e ideas que se pueden plasmar en un proyecto de interés hacia un futuro. El procesamiento de imágenes a través de diferentes dispositivos como una cámara celular u otras herramientas usadas para ciertos fines específicos, no están exentos de presentar problemas en la captación de datos como las imágenes, y existe la posibilidad de no tener el resultado deseado. Es ahí, cuando se hace necesario buscar soluciones para la resolución de estos desperfectos, como es el caso del ruido denominado “sal y pimienta”, en este proyecto se intentó desarrollar una propuesta basada en programación recursiva. A modo de conclusión, utilizando la recursividad, se puede llegar a una solución, sin embargo, esta manera de programar necesita cierto poder computacional requerido para el resultado esperado, ocasionando que sea un poco ineficiente y tarde de más procesando imágenes de mayor tamaño. No obstante, es una buena forma de aprender desde una óptica distante el cómo solucionar problemas, aunado a esto, como solución se plantea el usar imágenes comprimidas o hacer uso de librerías especializadas en el procesamiento de imágenes que faciliten la obtención de resultados precisos. Otro aspecto con el que llegamos a la conclusión es que entre más grande sea la

ventana, ocurre que el proceso de filtrado también se alarga bastante, nos dimos cuenta de esto con la prueba de filtrar la imagen de la mujer con una ventana de 11, el cual es una prueba que no pudimos realizar de manera correcta ya que el proceso de filtrado tardó horas e incluso el tipo de algoritmo de ordenamiento toma un papel importante.

6 Bibliografía

Referencias

- [1] Jamil Azzeh, Bilal Zahran, and Ziad Alqadi. Salt and pepper noise: Effects and removal. *JOIV: International Journal on Informatics Visualization*, 2(4):252–256, 2018.