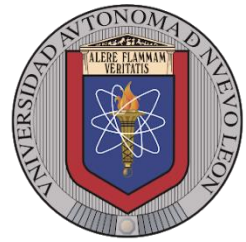


UANL

UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA Y ELECTRICA

FIME

Nombre: Roberto Erick Aguilar Morales

Matricula: 1871004

Carrera: Ingeniero en Tecnologías del Software

3.– Filtros

Materia: VISION COMPUTACIONAL LABORATORIO

Docente: RAYMUNDO SAID ZAMORA PEQUEÑO

Hora: N1-N2 **Días: Miércoles**

Fecha: 06/10/24

Objetivo

Utilizar filtros para eliminar perturbaciones en la imagen

Marco teórico

Procesamiento Digital de Imágenes

El procesamiento digital de imágenes es una disciplina dentro de la visión por computadora que se encarga de modificar, mejorar o analizar imágenes mediante técnicas computacionales. En este contexto, la reducción de ruido es un proceso fundamental, ya que las imágenes suelen verse afectadas por diversas perturbaciones o "ruidos" que pueden distorsionar la información visual importante.

Uno de los enfoques más utilizados para la reducción de ruido en imágenes es el filtrado, que implica modificar los valores de los píxeles para atenuar o eliminar las perturbaciones. Los filtros de mediana y moda son dos métodos no lineales comúnmente utilizados en este ámbito.

Filtrado de Mediana

El filtro de mediana es un método no lineal que reemplaza el valor de cada píxel en una imagen con la mediana de los valores de los píxeles vecinos. La mediana es el valor intermedio de un conjunto de números, por lo que este tipo de filtro es muy efectivo para eliminar el ruido de tipo "sal y pimienta" sin afectar significativamente los bordes o detalles finos de la imagen.

Funcionamiento del Filtro de Mediana:

1. Se selecciona una parte de la imagen y se recorta para obtener los valores de su matriz y ordenarlos de menor a mayor
2. Obtenemos la mediana de esos valores y lo guardamos para reemplazarlo
3. Se promedia los valores de la imagen y dependiendo la selección, serán reemplazados por el valor de la mediana.

Este proceso asegura que el valor resultante sea representativo de los píxeles circundantes, eliminando valores atípicos que podrían ser causados por ruido.

Aplicaciones del Filtro de Mediana:

- Eliminación de ruido en imágenes afectadas por interferencias.
- Conservación de los bordes de objetos en la imagen, ya que, a diferencia de otros métodos como el filtro promedio, no difumina los contornos.

Filtrado de Moda

El filtro de moda es otro método no lineal en el que el valor de cada píxel en una imagen se reemplaza por la moda de los valores en el vecindario de píxeles circundantes. La moda es el valor que aparece con mayor frecuencia en un conjunto de datos, lo que hace que este filtro sea adecuado para situaciones en las que la imagen tiene áreas de valores repetidos.

Funcionamiento del Filtro de Moda:

1. Se selecciona una parte de la imagen y se recorta para obtener los valores de su matriz y ordenarlos de menor a mayor
2. Obtenemos la moda de esos valores y lo guardamos para reemplazarlo
3. Se promedia los valores de la imagen y dependiendo la selección, serán reemplazados por el valor de la moda.

En algunos casos, la imagen puede tener más de una moda en el vecindario (múltiples valores con la misma frecuencia). Para esos casos, se puede emplear un filtro multimodal, en el cual se utilizan todas las modas para generar diferentes versiones de la imagen filtrada y luego se selecciona la mejor solución, generalmente basada en el valor más pequeño.

Aplicaciones del Filtro de Moda:

- El filtro de moda es útil en imágenes donde los valores son mayormente homogéneos o discretos, como en imágenes de códigos binarios o segmentaciones.
- Es útil cuando la presencia de ruido produce cambios en los valores que son raros en el vecindario de píxeles.

Reducción de Ruido Mediante Filtros No Lineales

La reducción de ruido en imágenes es esencial para mejorar la calidad visual y la precisión de análisis posteriores, como la detección de bordes o la segmentación. Los filtros no lineales, como los filtros de mediana y moda, son preferidos en muchas situaciones porque:

- No promueven el difuminado o suavizado generalizado de los bordes (como lo hacen los filtros lineales, como el promedio).
- Son robustos frente a valores atípicos o extremos que podrían distorsionar el resultado.

Implementación en Python y OpenCV

En ambos archivos proporcionados, se utiliza la biblioteca OpenCV, ampliamente conocida en el campo de la visión por computadora, para la carga y manipulación de imágenes. Las funciones implementadas en los scripts permiten aplicar estos

filtros de manera flexible, adaptándose a las características específicas de las imágenes:

- **Cálculo de matrices de dispersión**: Una técnica utilizada para medir la desviación de los valores de píxeles respecto al promedio general de la imagen, identificando áreas de perturbación.
- **Filtrado adaptativo**: En los scripts, el usuario puede ajustar el **umbral** para determinar qué píxeles serán reemplazados, lo que da control sobre el grado de filtrado aplicado.

Histograma de Imágenes

Un **histograma de imagen** es una representación gráfica que muestra la distribución de los valores de intensidad de los píxeles en una imagen. Es una herramienta útil para comprender el rango de tonos en la imagen y puede ayudar a identificar ruido o anomalías. En los códigos proporcionados, se utilizan histogramas para visualizar tanto la imagen original como la imagen filtrada, lo que facilita el análisis del proceso de reducción de ruido.

Introducción

Ambos archivos, P03FiltroMediana.py y P03FiltroModa.py, implementan diferentes métodos de filtrado de imágenes con el propósito de eliminar ruido. Utilizan la librería OpenCV para el procesamiento de las imágenes y NumPy para las operaciones de matrices. Las imágenes se convierten a escala de grises, y se aplican diferentes técnicas de filtrado.

Código del Filtro de Mediana: Mediana.py

El filtro de mediana es utilizado para reemplazar los valores de los píxeles perturbados con la mediana de los valores de los píxeles vecinos, eliminando así las perturbaciones sin perder los bordes de la imagen.

Importación de Bibliotecas

El código comienza con la importación de las bibliotecas necesarias:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

- cv2: Se utiliza para manejar las imágenes y realizar operaciones con OpenCV.
- numpy: Para el manejo de matrices de imágenes y cálculos.
- matplotlib.pyplot: Para la visualización de histogramas.

Creación de la Matriz de Dispersión

La función crear_matriz_dispersion crea una matriz que mide la dispersión de los valores de los píxeles en comparación con un valor promedio de la imagen:

```
# Funcion para crea la matriz de dispersion de la imagen original en escala
de grises
def crear_matriz_dispersion(imagen, promedio):
    nueva_matriz_de_dispersion = imagen.copy()

    for i in range(imagen.shape[0]):
        for j in range(imagen.shape[1]):

            nueva_matriz_de_dispersion[i][j] = abs(imagen[i,j] - promedio) *
100 / promedio

    return nueva_matriz_de_dispersion
```

Esta función calcula la desviación porcentual de cada píxel respecto al promedio, lo que permite identificar los píxeles perturbados.

Generación de Histograma de la Imagen

La función histograma_imagen genera el histograma de la imagen original en escala de grises:

```
# Funcion para crear la matriz de la imagen original en escala de grises
def histograma_imagen(imagen):
    plt.hist(imagen.ravel(), bins=256, range=[0, 256], color='gray',
alpha=0.7)

    plt.title('Histograma de Barras - Imagen Original')
    plt.xlabel('Valor de Intensidad (0-255)')
    plt.ylabel('Número de Píxeles')

    plt.show()
```

Aquí se visualiza la distribución de las intensidades de los píxeles de la imagen, lo que es útil para observar la presencia de ruido.

Además también se mestra el histograma de la matriz de dispersión para tener una idea de los valores mas alejados del resto y eliminarlos

Aplicación del Filtro de Mediana

La función crear_imagen_filtrada aplica el filtro de mediana, reemplazando los píxeles perturbados por el valor de la mediana:

```
# Crear la imagen final del filtrado de imagen con la mediana
```

```
def crear_imagen_filtrada(imagen, matriz_de_dispersion, filtro,
imagen_filtrada_con_mediana, mediana):
    # Esta funcion de OpenCv ya esta hecha para quitar el ruido de la imagen
pero no encaja con los metodos enseñados
    #matriz_de_dispersion = cv2.medianBlur(imagen, 5)

    for i in range(imagen.shape[0]):
        for j in range(imagen.shape[1]):
            if matriz_de_dispersion[i][j] >= filtro:
                imagen_filtrada_con_mediana[i][j] = mediana
```

Ejecución Principal

El bloque principal del código carga la imagen, la convierte a escala de grises, la recorta y aplica los procedimientos anteriores:

```
# Cargar la imagen en escala de grises
imagen_color = cv2.imread("imagen.png")
if imagen_color is None:
    print("Error: No se pudo cargar la imagen")
else:
    # Convertir la imagen a escala de grises
    imagen = cv2.cvtColor(imagen_color, cv2.COLOR_BGR2GRAY)

    print("Dimensiones de la imagen en escala de grises:", imagen.shape)
    print("                                                Y    X")

    # Leer las coordenadas de inicio
    x_inicio = int(input("Escoja la posición en X (columna): "))
    y_inicio = int(input("Escoja la posición en Y (fila): "))

    # Definir las dimensiones del recorte (62 en Y, 40 en X)
    x_final = x_inicio + 40
    y_final = y_inicio + 62

    # Asegurarse de que los límites no excedan el tamaño de la imagen
    if x_final > imagen.shape[1] or y_final > imagen.shape[0]:
        print("Error: Las coordenadas exceden el tamaño de la imagen.")
    else:
        # Recortar la imagen utilizando slicing de NumPy
        imagen_recortada = imagen[y_inicio:y_final, x_inicio:x_final]

        # Crear diferentes imagenes para mostrar el resultado
        imagen_filtrada_con_mediana = imagen_recortada.copy()
        matriz_de_dispersion = imagen_recortada.copy()
```

```
        # Promedio de la imagen
        promedio = np.mean(imagen_recortada)

        # Mediana de la imagen
        mediana = np.median(imagen_recortada)

        # Creacion de la matriz de dispersion
        matriz_de_dispersion = crear_matriz_dispersion(imagen_recortada,
promedio)

        # Histograma de la imagen original en escala de grises
        histograma_imagen(imagen_recortada)

        # Crear el histograma de la matriz dispersion con informacion para
el usuario
        histograma_matriz_dispersion(matriz_de_dispersion)

        # Filtro que se le va a colocar a la imagen (a partir de qué número
de la matriz de dispersion va a reemplazar los valores en la imagen)
        print("Mediana de la imagen (valor por el que va a cambiar los
pixeles seleccionados): ", mediana)
        print("Valor recomendado con las imagenes de prueba: 97")
        filtro = int(input("Introduce el filtro que quieres colocar (0 -
100): "))

        # Creacion de la imagen dependiendo de los datos obtenidos
anteriormente y la respuesta del usuario
        crear_imagen_filtrada(imagen_recortada, matriz_de_dispersion,
filtro, imagen_filtrada_con_mediana, mediana)

        # Promedio de la imagen filtrada resultante
        promedio_final = np.mean(imagen_filtrada_con_mediana)

        # Obtener la matriz de dispersion resultante
        matriz_de_dispersion_resultante =
crear_matriz_dispersion(imagen_filtrada_con_mediana, promedio_final)

        # Dispersion de la matriz resultante final
        print("El valor final de la sumatoria de la matriz resultante es la
siguiente: ", np.sum(matriz_de_dispersion_resultante))
```

Mostrar y guardar la imagen

Finalmente muestra los resultados obtenidos al procesar la imagen, sus matrices y los guarda en la misma carpeta donde se este ejecutando el mismo programa

```
        # Guardar matrizes de imágenes procesadas
        np.savetxt('imagen_original.csv', imagen_recortada, delimiter=',',
fmt='%d')
        np.savetxt('matriz_de_dispersion_mediana.csv', matriz_de_dispersion,
delimiter=',', fmt='%d')
        np.savetxt('imagen_filtrada_con_mediana.csv',
imagen_filtrada_con_mediana, delimiter=',', fmt='%d')
        np.savetxt('matriz_de_dispersion_mediana_resultante.csv',
matriz_de_dispersion_resultante, delimiter=',', fmt='%d')

        # Mostrar imagenes procesadas
        cv2.imshow("Imagen original", imagen_recortada)
        cv2.imshow("Matriz de dispersión mediana", matriz_de_dispersion)
        cv2.imshow("Imagen filtrada con mediana",
imagen_filtrada_con_mediana)
        cv2.imshow("Matriz de dispersion resultante",
matriz_de_dispersion_resultante)

        cv2.waitKey(0)
        cv2.destroyAllWindows()

        # Guarar imagenes en los archivos
        cv2.imwrite("imagen_filtrada_con_mediana.png",
imagen_filtrada_con_mediana)
```

En cuanto a el filtro de Moda es exactamente igual solo cambia la aplicación del filtro de mediana, en este caso siendo la aplicación de la moda y la multimoda

```
# Funciones para crear las modas de las imagenes
def imagen_filtrada(imagen):
    for i in range(imagen.shape[0]):
            for j in range(imagen.shape[1]):
                if matriz_de_dispersion[i][j] >= filtro :
                    imagen_filtrada_con_moda[i][j] = moda

def imagen_filtrada_multimoda(imagen):
    modas_matrices = []
    modas_sumas = []

    # Crear una imagen filtrada por cada moda
    for moda in modas:
```

```
        imagen_temp = np.zeros_like(imagen)  # Crear una copia vacía de la
imagen

        for i in range(imagen.shape[0]):
            for j in range(imagen.shape[1]):
                if matriz_de_dispersion[i][j] >= filtro:
                    imagen_temp[i][j] = moda

        # Calcular la suma de la matriz filtrada
        suma_matriz = np.sum(imagen_temp)
        modas_matrices.append(imagen_temp)
        modas_sumas.append(suma_matriz)

    # Encontrar la moda que genere la suma más pequeña
    indice_menor_suma = np.argmin(modas_sumas)
    imagen_filtrada_con_moda = modas_matrices[indice_menor_suma]

    print(f"La moda seleccionada es {modas[indice_menor_suma]} con suma
{modas_sumas[indice_menor_suma]}")

    return imagen_filtrada_con_moda
```

Cálculos y resultados

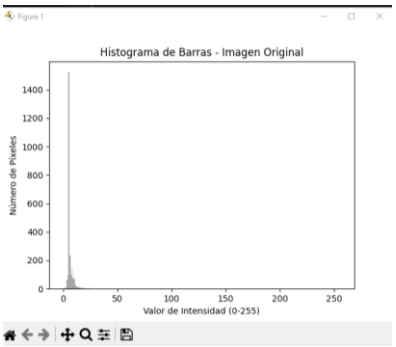
P03FiltroMediana.py

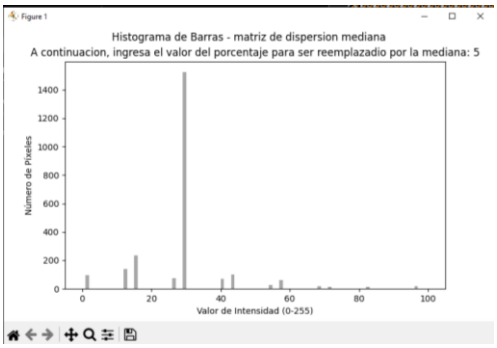
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Funcion para crea la matriz de dispersion de la imagen original en escala
de grises
def crear_matriz_dispersion(imagen, promedio):
    nueva_matriz_de_dispersion = imagen.copy()

    for i in range(imagen.shape[0]):
        for j in range(imagen.shape[1]):

            nueva_matriz_de_dispersion[i][j] = abs(imagen[i,j] - promedio) *
100 / promedio

    return nueva_matriz_de_dispersion

# Funcion para crear la matriz de la imagen original en escala de grises
def histograma_imagen(imagen):
```

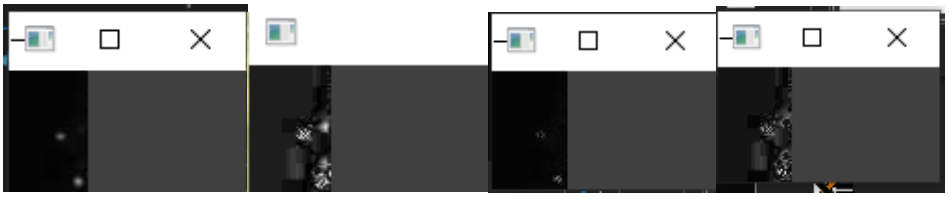
```
    plt.hist(imagen.ravel(), bins=256, range=[0, 256], color='gray',
alpha=0.7)

    plt.title('Histograma de Barras - Imagen Original')
    plt.xlabel('Valor de Intensidad (0-255)')
    plt.ylabel('Número de Píxeles')

    plt.show()

# Función para crear la matriz de la dispersion de los pixeles
def histograma_matriz_dispersion(matriz_de_dispersion):

    # Histograma de la matriz de dispersion
    plt.hist(matriz_de_dispersion.ravel(), bins=100, range=[0, 100],
color='gray', alpha=0.7)

    plt.suptitle('Histograma de Barras - matriz de dispersion mediana')
    plt.title('A continuacion, ingresa el valor del porcentaje para ser
reemplazadio por la mediana: %i' %mediana)
    plt.xlabel('Valor de Intensidad (0-255)')
    plt.ylabel('Número de Píxeles')

    plt.show()

# Crear la imagen final del filtrado de imagen con la mediana
def crear_imagen_filtrada(imagen, matriz_de_dispersion, filtro,
imagen_filtrada_con_mediana, mediana):
    # Esta funcion de OpenCv ya esta hecha para quitar el ruido de la imagen
pero no encaja con los metodos enseñados
    #matriz_de_dispersion = cv2.medianBlur(imagen, 5)

    for i in range(imagen.shape[0]):
        for j in range(imagen.shape[1]):
            if matriz_de_dispersion[i][j] >= filtro:
                imagen_filtrada_con_mediana[i][j] = mediana

# Cargar la imagen en escala de grises
imagen_color = cv2.imread("imagen.png")
if imagen_color is None:
    print("Error: No se pudo cargar la imagen")
else:
    # Convertir la imagen a escala de grises
    imagen = cv2.cvtColor(imagen_color, cv2.COLOR_BGR2GRAY)

    print("Dimensiones de la imagen en escala de grises:", imagen.shape)
```

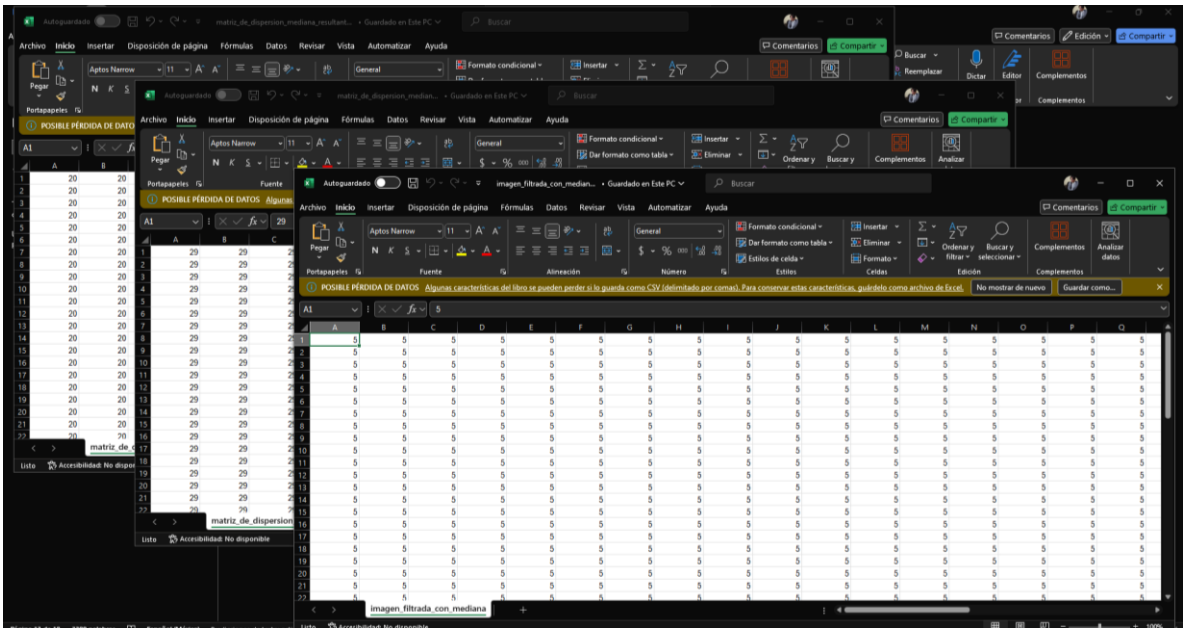
```
    print("                                                Y    X")

    # Leer las coordenadas de inicio
    x_inicio = int(input("Escoja la posición en X (columna): "))
    y_inicio = int(input("Escoja la posición en Y (fila): "))

    # Definir las dimensiones del recorte (62 en Y, 40 en X)
    x_final = x_inicio + 40
    y_final = y_inicio + 62

    # Asegurarse de que los límites no excedan el tamaño de la imagen
    if x_final > imagen.shape[1] or y_final > imagen.shape[0]:
        print("Error: Las coordenadas exceden el tamaño de la imagen.")
    else:
        # Recortar la imagen utilizando slicing de NumPy
        imagen_recortada = imagen[y_inicio:y_final, x_inicio:x_final]

        # Crear diferentes imagenes para mostrar el resultado
        imagen_filtrada_con_mediana = imagen_recortada.copy()
        matriz_de_dispersion = imagen_recortada.copy()

        # Promedio de la imagen
        promedio = np.mean(imagen_recortada)

        # Mediana de la imagen
        mediana = np.median(imagen_recortada)

        # Creacion de la matriz de dispersion
        matriz_de_dispersion = crear_matriz_dispersion(imagen_recortada,
promedio)

        # Histograma de la imagen original en escala de grises
        histograma_imagen(imagen_recortada)

        # Crear el histograma de la matriz dispersion con informacion para
el usuario
        histograma_matriz_dispersion(matriz_de_dispersion)

        # Filtro que se le va a colocar a la imagen (a partir de qué número
de la matriz de dispersion va a reemplazar los valores en la imagen)
        print("Mediana de la imagen (valor por el que va a cambiar los
pixeles seleccionados): ", mediana)
        print("Valor recomendado con las imagenes de prueba: 97")
        filtro = int(input("Introduce el filtro que quieres colocar (0 -
100): "))
```

```
        # Creacion de la imagen dependiendo de los datos obtenidos
anteriormente y la respuesta del usuario
        crear_imagen_filtrada(imagen_recortada, matriz_de_dispersion,
filtro, imagen_filtrada_con_mediana, mediana)

        # Promedio de la imagen filtrada resultante
        promedio_final = np.mean(imagen_filtrada_con_mediana)

        # Obtener la matriz de dispersion resultante
        matriz_de_dispersion_resultante =
crear_matriz_dispersion(imagen_filtrada_con_mediana, promedio_final)

        # Dispersion de la matriz resultante final
        print("El valor final de la sumatoria de la matriz resultante es la
siguiente: ", np.sum(matriz_de_dispersion_resultante))

        # Guardar matrizes de imágenes procesadas
        np.savetxt('imagen_original.csv', imagen_recortada, delimiter=',',
fmt='%d')
        np.savetxt('matriz_de_dispersion_mediana.csv', matriz_de_dispersion,
delimiter=',', fmt='%d')
        np.savetxt('imagen_filtrada_con_mediana.csv',
imagen_filtrada_con_mediana, delimiter=',', fmt='%d')
        np.savetxt('matriz_de_dispersion_mediana_resultante.csv',
matriz_de_dispersion_resultante, delimiter=',', fmt='%d')

        # Mostrar imagenes procesadas
        cv2.imshow("Imagen original", imagen_recortada)
        cv2.imshow("Matriz de dispersión mediana", matriz_de_dispersion)
        cv2.imshow("Imagen filtrada con mediana",
imagen_filtrada_con_mediana)
        cv2.imshow("Matriz de dispersion resultante",
matriz_de_dispersion_resultante)

        cv2.waitKey(0)
        cv2.destroyAllWindows()

        # Guarar imagenes en los archivos
        cv2.imwrite("imagen_filtrada_con_mediana.png",
imagen_filtrada_con_mediana)
```









P03FiltroModa.py

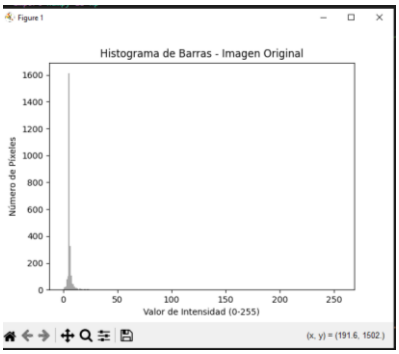
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Funcion para crea la matriz de dispersion de la imagen original en escala
de grises
def crear_matriz_dispersion(imagen, promedio):
    nueva_matriz_de_dispersion = imagen.copy()

    for i in range(imagen.shape[0]):
```

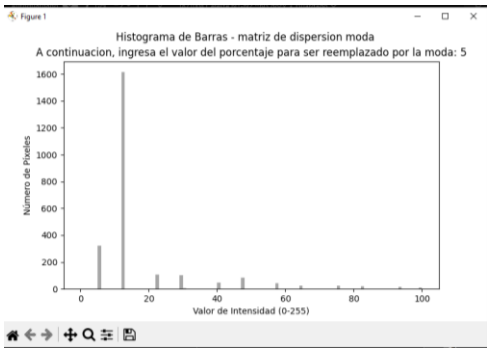
```
        for j in range(imagen.shape[1]):

            nueva_matriz_de_dispersion[i][j] = abs(imagen[i,j] - promedio) *
100 / promedio

    return nueva_matriz_de_dispersion

# Funcion para crear la matriz de la imagen original en escala de grises
def histograma_imagen(imagen):
    plt.hist(imagen.ravel(), bins=256, range=[0, 256], color='gray',
alpha=0.7)

    plt.title('Histograma de Barras - Imagen Original')
    plt.xlabel('Valor de Intensidad (0-255)')
    plt.ylabel('Número de Píxeles')

    plt.show()

# Función para crear la matriz de la dispersion de los pixeles
def histograma_matriz_dispersion(matriz_de_dispersion):

    # Histograma de la matriz de dispersion
    plt.hist(matriz_de_dispersion.ravel(), bins=100, range=[0, 100],
color='gray', alpha=0.7)

    plt.suptitle('Histograma de Barras - matriz de dispersion moda')
    plt.title('A continuacion, ingresa el valor del porcentaje para ser
reemplazado por la moda: %i' %moda)
    plt.xlabel('Valor de Intensidad (0-255)')
    plt.ylabel('Número de Píxeles')

    plt.show()

# Funciones para crear las modas de las imagenes
def imagen_filtrada(imagen):
    for i in range(imagen.shape[0]):
            for j in range(imagen.shape[1]):
                if matriz_de_dispersion[i][j] >= filtro :
                    imagen_filtrada_con_moda[i][j] = moda

def imagen_filtrada_multimoda(imagen):
    modas_matrices = []
    modas_sumas = []

    # Crear una imagen filtrada por cada moda
```

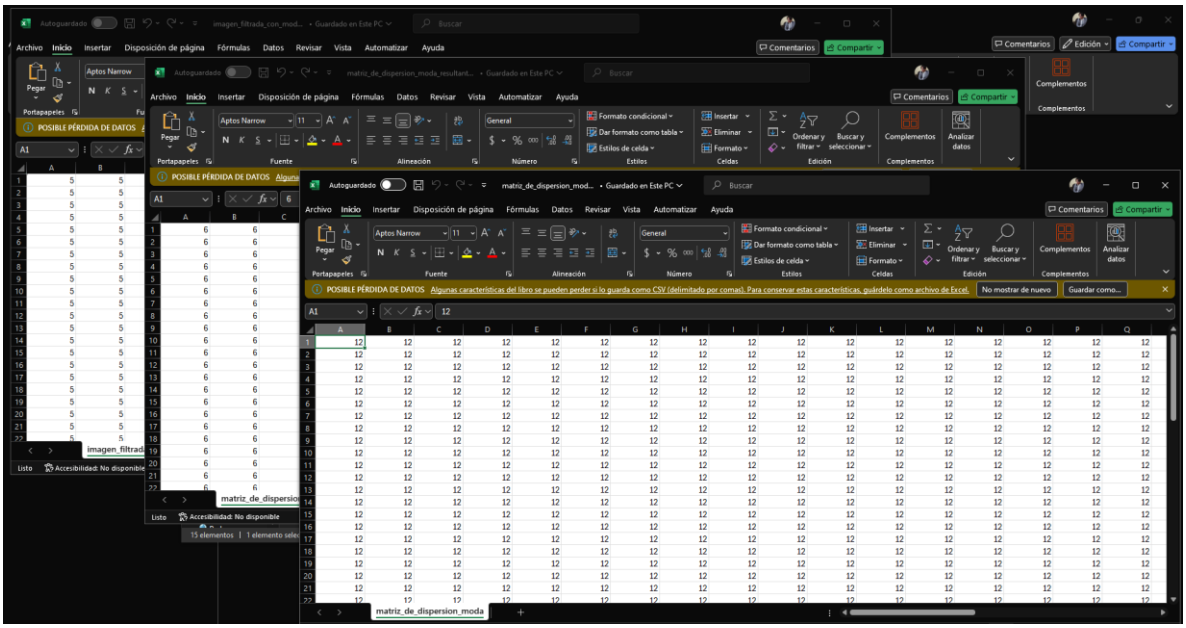
```
    for moda in modas:
        imagen_temp = np.zeros_like(imagen)  # Crear una copia vacía de la
imagen

        for i in range(imagen.shape[0]):
            for j in range(imagen.shape[1]):
                if matriz_de_dispersion[i][j] >= filtro:
                    imagen_temp[i][j] = moda

        # Calcular la suma de la matriz filtrada
        suma_matriz = np.sum(imagen_temp)
        modas_matrices.append(imagen_temp)
        modas_sumas.append(suma_matriz)

    # Encontrar la moda que genere la suma más pequeña
    indice_menor_suma = np.argmin(modas_sumas)
    imagen_filtrada_con_moda = modas_matrices[indice_menor_suma]

    print(f"La moda seleccionada es {modas[indice_menor_suma]} con suma
{modas_sumas[indice_menor_suma]}")

    return imagen_filtrada_con_moda

# Cargar la imagen en escala de grises
imagen_color = cv2.imread("imagen.png")
if imagen_color is None:
    print("Error: No se pudo cargar la imagen")
else:
    # Convertir la imagen a escala de grises
    imagen = cv2.cvtColor(imagen_color, cv2.COLOR_BGR2GRAY)

    print("Dimensiones de la imagen en escala de grises:", imagen.shape)
    print("                                                Y    X")

    # Leer las coordenadas de inicio
    x_inicio = int(input("Escoja la posición en X (columna): "))
    y_inicio = int(input("Escoja la posición en Y (fila): "))

    # Definir las dimensiones del recorte (62 en Y, 40 en X)
    x_final = x_inicio + 40
    y_final = y_inicio + 62

    # Asegurarse de que los límites no excedan el tamaño de la imagen
    if x_final > imagen.shape[1] or y_final > imagen.shape[0]:
        print("Error: Las coordenadas exceden el tamaño de la imagen.")
```

```
    else:
        # Recortar la imagen utilizando slicing de NumPy
        imagen_recortada = imagen[y_inicio:y_final, x_inicio:x_final]

        # Crear diferentes imagenes para mostrar el resultado
        imagen_filtrada_con_moda = imagen_recortada.copy()
        matriz_de_dispersion = imagen_recortada.copy()

        # Promedio de la imagen original
        promedio = np.mean(imagen_recortada)

        # Los siguientes pasos son para obtener la moda o multiples modas
        # Aplanar la imagen recortada
        valores, conteos = np.unique(imagen_recortada.flatten(),
return_counts=True)

        # Obtener el valor máximo de frecuencia
        max_frecuencia = np.max(conteos)

        # Obtener todas las modas (valores con la máxima frecuencia)
        modas = valores[conteos == max_frecuencia]

        # Si hay más de una moda
        if len(modas) > 1:
            print(f"Hay múltiples modas: {modas}")
        else:
            print(f"La moda es: {modas[0]}")
            moda = modas[0]
        # Hasta aqui sabemos las modas o moda que se tiene en la imagen

        # Creacion de la matriz de dispersion
        matriz_de_dispersion = crear_matriz_dispersion(imagen_recortada,
promedio)

        # Histograma de la imagen original en escala de grises
        histograma_imagen(imagen_recortada)

        # Crear el histograma de la matriz dispersion con informacion para
el usuario
        histograma_matriz_dispersion(matriz_de_dispersion)

        # Filtro que se le va a colocar a la imagen (a partir de qué número
de la matriz de dispersion va a reemplazar los valores en la imagen)
        print("Mediana de la imagen (valor por el que va a cambiar los
pixeles seleccionados): ", moda)
```

```
        print("Valor recomendado: cercas de 70")
        filtro = int(input("Introduce el filtro que quieres colocar (0 -
100): "))

        # Reemplazar los valores por la moda
        if len(modas) > 1:
            # Si hay mas de una moda usamos un mulltimodal
            imagen_filtrada_multimoda(imagen_recortada)
        else:
            # Si solo es una moda procedemos normal
            imagen_filtrada(imagen_recortada)

        # Promedio de la imagen filtrada resultante
        promedio_final = np.mean(imagen_filtrada_con_moda)

        # Obtener la matriz de dispersion resultante
        matriz_de_dispersion_resultante =
crear_matriz_dispersion(imagen_filtrada_con_moda, promedio_final)

        # Dispersion de la matriz resultante final
        print("El valor final de la sumatoria de la matriz resultante es la
siguiente: ", np.sum(matriz_de_dispersion_resultante))

        np.savetxt('imagen_original.csv', imagen_recortada, delimiter=',',
fmt='%d')
        np.savetxt('matriz_de_dispersion_moda.csv', matriz_de_dispersion,
delimiter=',', fmt='%d')
        np.savetxt('imagen_filtrada_con_moda.csv', imagen_filtrada_con_moda,
delimiter=',', fmt='%d')
        np.savetxt('matriz_de_dispersion_moda_resultante.csv',
matriz_de_dispersion_resultante, delimiter=',', fmt='%d')

        # Mostrar la imagen recortada
        cv2.imshow("Imagen original", imagen_recortada)
        cv2.imshow("matriz de dispersion moda", matriz_de_dispersion)
        cv2.imshow("imagen filtrada con moda", imagen_filtrada_con_moda)
        cv2.imshow("Matriz de dispersion resultante",
matriz_de_dispersion_resultante)

        cv2.waitKey(0)
        cv2.destroyAllWindows()

        # Guardar la imagen en los archivos
        cv2.imwrite("imagen_filtrada_con_moda.png",
imagen_filtrada_con_moda)
```







Conclusiones

Ambos algoritmos son efectivos para la eliminación de ruido en imágenes afectadas por perturbaciones. El filtro de mediana es particularmente útil cuando se necesita conservar los bordes de los objetos en la imagen, mientras que el filtro de moda es útil en imágenes con valores discretos o homogéneos.

La implementación en Python mediante la librería OpenCV permite una manipulación eficiente de las imágenes y la interactividad con el usuario para ajustar los parámetros de filtrado, además complementándolo con NumPy es

posible hacer los cálculos suficientes para crear estos filtros, mientras tanto matplotlib nos ayuda a visualizar mediante graficas que elementos eliminar

Bibliografía

OpenCV modules. OpenCV. (n.d.). https://docs.opencv.org/4.x/index.html

NumPy Org. (2024, June 17). NumPy. https://numpy.org/

Matplotlib 3.5.3 documentation#. Matplotlib documentation - Matplotlib 3.5.3 documentation. (2012). https://matplotlib.org/3.5.3/index.html