Asignatura: Procesamiento de imágenes
Profesor: D. Sc. Gerardo García Gil
Alumno: Jonathan Guillermo Díaz Magallanes.
Registro: 19310153. **Ciclo:** 2022-B
Ingeniería en Desarrollo de Software
Centro de Enseñanza Técnica Industrial (CETI)

Introducción

En el transcurso de ésta práctica se desarrollarán dos filtros; *Salt n' Pepper* y *mediana*. Primero se realizará el *Salt n' Pepper* para añadir ruido a la imagen en cuestión y después el filtro de mediana para eliminar el ruido producido. Ésto se realizará por medio del lenguaje científico MATLAB.

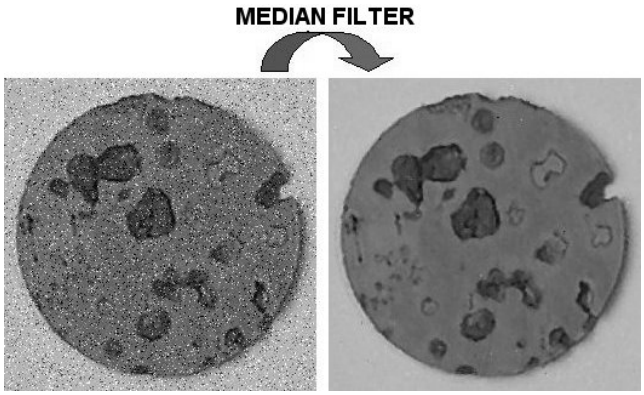
Filtro Salt n' Pepper

El ruido sal y pimienta (salt-and-pepper noise) que se presenta principalmente en imágenes. Se caracteriza principalmente por cubrir de forma dispersa toda la imagen con una serie de píxeles blancos y negros.

Este tipo de ruido suele producirse cuando la señal de la imagen es afectada por intensas y repentinas perturbaciones o impulsos.

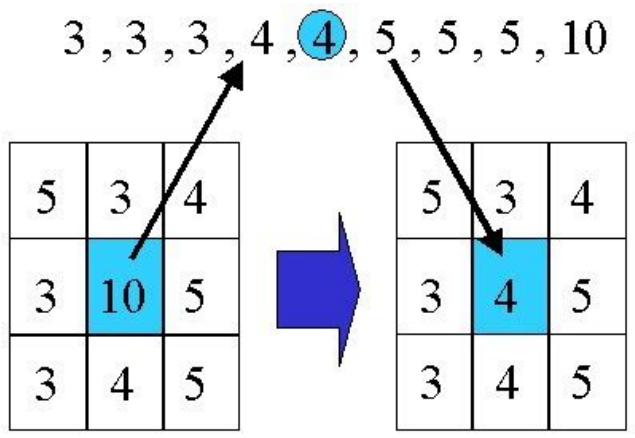
Filtro mediana

El filtro mediano es una técnica de filtrado digital no lineal, a menudo utilizada para eliminar el ruido de una imagen o señal. Dicha reducción de ruido es un paso típico de preprocesamiento para mejorar los resultados del procesamiento posterior. El filtrado mediano es muy utilizado en el procesamiento digital de imágenes porque, bajo ciertas condiciones, preserva los bordes mientras elimina el ruido, también tiene aplicaciones en el procesamiento de señales.



La idea principal del filtro mediano es correr a través de la señal entrada por entrada, reemplazando cada entrada con la mediana de las entradas vecinas. El patrón de vecinos se llama "ventana", que se desliza, entrada por entrada, sobre toda la señal. Para las señales unidimensionales, la ventana más obvia son solo las primeras entradas anteriores y siguientes, mientras que para los datos bidimensionales la

ventana debe incluir todas las entradas dentro de un radio o región elipsoidal dada.



Desarrollo

Preparaciones

El primer filtro que se desarrolló fue el de *Salt n' Pepper*.

Primeramente se leyó la imagen a tratar y se guardó en la variable *I*, además de guardar sus dimensiones en otras variables llamadas *Filas* y *Columnas*.

Después de éso se pasó la imagen a escala de grises con la función *rgb2gray* (La cual es proporcionada por las librerías de MATLAB) para poder hacer su tratamiento con el filtro *Salt n' Pepper*. Al cambiar la imagen a escala de grises obtendremos la siguiente:

Filtro Salt n' Pepper

El filtro de *Salt n' Pepper* consiste en dos funciones for, unos para los puntos blancos y otro para los puntos negros. La dimensión de cada for estará medido con la cantidad de partículas que se desean insertar en la imágen.

Dentro de los for primero se obtendrá una coordenada aleatoria de la imagen por medio de las siguientes funciones:

```
x=round(rand*Filas);
y=round(rand*Columnas);
```

Una vez tenemos las coordenadas aleatorias debemos establecer ciertos límites para que siempre obtengamos coordenadas de píxel que tengan una vecindad de 8 existente. Para ello hacemos éstos límites por medio de funciones if, los cuales serán 4 en total, dos para las coordenadas en *x* y *y* que estén al principio de la matriz de la imagen y otros dos para las coordenadas en *x* y *y* que estén al final de la matriz de la imagen.

```
if (x==0) || (x==1) || (x==2)
  x=3;
end
if (y==0) || (y==1) || (y==2)
  y=3;
end
if (x==Filas) || (x==Filas-1) || (x==Filas-2)
  x=Filas-3;
end
if (y==Filas) || (y==Filas-1) || (y==Filas-2)
  y=Columnas-3;
end
```

Una vez tenemos las coordenadas aleatorias finales vamos a proceder a insertar el valor del píxel en dichas coordenadas. En un alrededor de 2 píxeles para que de un mejor efecto.

```
for i=1:4
  rRandom = randi([-2 2]);
  cRandom = randi([-2 2]);
  Dr(x+rRandom,y+cRandom) = 255;
end
```

Lo anterior se implementa de la misma manera en ambos for con la diferencia en la intensidad a la que

se le va a asignar al píxel, cambiando entre los dos valores 0 y 255.
Al momento de aplicar éste filtro con una intensidad de 5,000 partículas:

Filtro mediana

El filtro mediana consiste en dos for anidados para mapear la imagen y asignarle al píxel actual una intensidad igual a la mediana de los pixeles en una vecindad de 8.
Para obtener la mediana de los píxeles vamos a tener que guardar en una matriz los valores de los pixeles en la vecindad de 8:

```
mat=[B(i-1,j-1), B(i-1,j), B(i-1,j+1), B(i,j-1), B(i,j),...
    B(i,j+1), B(i+1,j-1), B(i+1,j), B(i+1,j+1)];
```

Una vez guardada la matriz, la pasaremos por un método de ordenamiento que en éste caso se usó un método burbuja para ordenar de menor a mayor los valores de la matriz:

```
for x=1:8
  for val=1:8
    if mat(val) > mat(val+1)
      temp = mat(val);
      mat(val) = mat(val+1);
      mat(val+1) = temp;
    end
 end
end
```

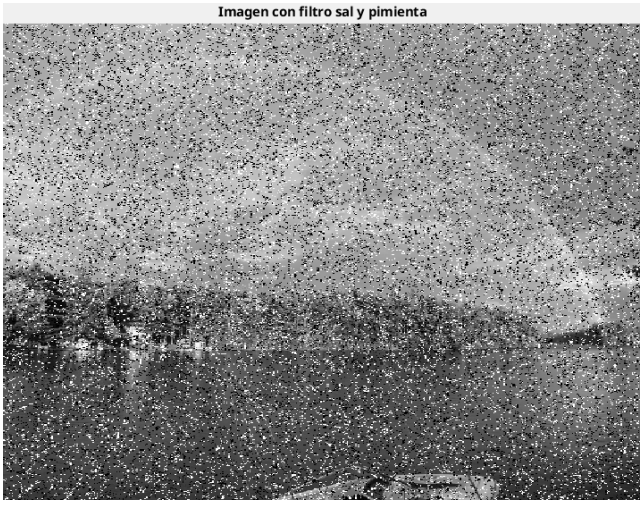
Finalmente al píxel en cuestión le vamos a asignar el valor que esté en la posición 5 de la matriz, el cual sería la mediana:

```
B(i,j)=mat(5)
```

Al momento de aplicar éste filtro a la imagen con el filtro *Salt n' Pepper* que obtuvimos anteriormente nos arroja el siguiente resultado:

El resultado elimina por completo las partículas generadas por el filtro Salt n' Pepper, sin embargo la imagen termina un poco borrosa en comparación a su estado original.

Se debe de mencionar que mientras mayor sea la cantidad de partículas que se encuentren en la imagen, menor va a ser la reducción de dichas partículas. Ésto se puede observar en el siguiente ejemplo donde ésta vez se le añadió una intensidad de 20,000 partículas:



Código

Filtro Salt n' Pepper

```
% Limpiamos los valores

clear all; clc;

% Leemos la imagen

I = imread('arco.jpg');

% Convertimos la imagen a escala de grises y obtenemos dimensiones

I = rgb2gray(I);

[Filas, Columnas, P] = size(I);

% Guardamos la imagen como double

D = double(I);

Dr = D;

% Empezamos con el filtro de ruido sal y pimienta

% Puntos blancos

for v=1:5000

 x=round(rand*Filas);

 y=round(rand*Columnas);

 if (x==0) || (x==1) || (x==2)

  x=3;

 end

 if (y==0) || (y==1) || (y==2)

  y=3;

 end

 if (x==Filas) || (x==Filas-1) || (x==Filas-2)

  x=Filas-3;

 end

 if (y==Filas) || (y==Filas-1) || (y==Filas-2)

  y=Columnas-3;

 end

 for i=1:4

  rRandom = randi([-2 2]);

  cRandom = randi([-2 2]);

  Dr(x+rRandom,y+cRandom) = 255;

 end

end

% Puntos negros
```

```
for v=1:5000
 x=round(rand*Filas);
 y=round(rand*Columnas);
 if (x==0) || (x==1) || (x==2)
  x=3;
 end
 if (y==0) || (y==1)|| (y==2)
  y=3;
 end
 if (x==Filas) || (x==Filas-1) || (x==Filas-2)
  x=Filas-3;
 end
 if (y==Filas) || (y==Filas-1) || (y==Filas-2)
  y=Columnas-3;
 end
 for i=1:4
  rRandom = randi([-2 2]);
  cRandom = randi([-2 2]);
  Dr(x+rRandom,y+cRandom) = 0;
 end
end
Dr = uint8(Dr);
% Area de impresion
figure;
imshow(I)
title('Imagen original')
figure;
imshow(Dr)
title('Imagen con filtro sal y pimienta')
```

Filtro mediana

```
% Limpiamos los valores
clear all; clc;
% Leemos la imagen
I = imread('arco2.jpg');
% Convertimos la imagen a Icala de grises y obtenemos dimensiones
I = rgb2gray(I);
[Filas, Columnas, P] = size(I);
% Mandamos a llamar la función para el añadido de ruido
Dr = ruidoSP(I, 20000);
% Empezamos con el filtro de mediana
B = Dr;
for i=2:Filas-1
  for j=2:Columnas-1
      mat=[B(i-1,j-1), B(i-1,j), B(i-1,j+1), B(i,j-1), B(i,j), B(i,j+1), B(i+1,j-1), B(i+1,j), B(i+1,j+1)];
% Empezamos con el metodo burbuja
    for x=1:8
     for val=1:8
      if mat(val) > mat(val+1)
       temp = mat(val);
       mat(val) = mat(val+1);
       mat(val+1) = temp;
      end
```

```
    end
   end
   B(i,j)=mat(5);
  end
end
% Area de impresion
figure;
imshow(I)
title('Imagen original')
figure;
imshow(Dr)
title('Imagen con filtro sal y pimienta')
figure;
imshow(B)
title('Imagen con filtro mediana')
```

Conclusión

En el desarrollo de ésta práctica no se presentaron problemas mayores al momento de hacer la elaboración de cada uno de los filtros. Se tuvo duda sobre cómo se podían obtener valores aleatorios en MATLAB y por ello se investigó y encontró la función *randi* la cual nos regresa un valor entero aleatorio dentro de los parámetros que se le hayan introducido. Fuera de ello no se presentó complicación o duda alguna.

Referencias

- *Filtro de Mediana. (n.d.). FIRST Robotics Competition Documentation. Retrieved September 25, 2022, from https://docs.wpilib.org/es/stable/docs/software/advanced-controls/filters/median-filter.html*
- *Ingraham, A. K. (2018, November 28). Salt & Pepper Noise and Median Filters, Part I – The Theory. Kyle Ingraham | Engineer. Retrieved September 25, 2022, from https://blog.kyleingraham.com/2016/12/14/salt-pepper-noise-and-median-filters-part-i-the-theory/*
- *Remove Salt and Pepper Noise from Images - MATLAB & Simulink. (n.d.). Retrieved September 25, 2022, from https://www.mathworks.com/help/vision/ug/remove-salt-and-pepper-noise-from-images.html*
- *Wikipedia contributors. (2021, September 10). Median filter. Wikipedia. Retrieved September 25, 2022, from https://en.wikipedia.org/wiki/Median_filter*