

Asignatura	Datos del alumno	Fecha
Percepción Computacional	Apellidos: Anchondo Trejo Nombre: Victor Eduardo	20/Junio/2021

## Laboratorio: Eliminación de anomalías de la imagen

### Filtro de la Mediana

El filtro de la mediana pertenece a los filtros de paso bajo, también conocidos como filtros de suavizado, su objetivo es reducir el ruido y/o efectos que pueden presentarse en una imagen a consecuencia del proceso de captura, digitalización y transmisión. El filtro de la mediana consiste en remplazar el valor de la señal por la estimación de la mediana en una ventana de tamaño finito. Comúnmente este filtro es utilizado para eliminar ruido impulsivo como el salt & pepper.

### Ruido "Salt & Pepper"

Este tipo de ruido se caracteriza principalmente por cubrir de forma dispersa toda la imagen con una serie de pixeles blancos y negros, suele producirse cuando la señal de la imagen es afectada por intensas y repentinhas perturbaciones o impulsos.

### Solución

#### Agregando librerías

```
In [1]: # Libreria opencv
import cv2
# Libreria NumPy
import numpy as np
#Libreria OS
import os
```

Función para leer una matriz subset de la "matrix" pasada como parámetro del tamaño especificado

En caso de que la "matrix" original ya no contenga mas valores, los valores restantes en la matriz nueva se quedaran con el valor -1

```
In [2]: def getSubMatrix(matrix, fromColumnIndex, toColumnIndex, fromRowIndex, toRowIndex):
    # Crea una lista conteniendo "colsSize" listas y cada una con "rowsSize" e inicializa cada elemento con -1
    windowMatrix = [[-1 for x in range(toColumnIndex - fromColumnIndex)] for y in range(toRowIndex - fromRowIndex)]
    totalColumns = matrix.shape[0]
    totalRows = matrix.shape[1]

    for x in range(fromColumnIndex, toColumnIndex):
        if totalColumns > x:
            for y in range(fromRowIndex, toRowIndex):
                if totalRows > y:
                    windowMatrix[x - fromColumnIndex][y - fromRowIndex] = matrix[x][y]

    return windowMatrix
```

Asignatura	Datos del alumno	Fecha
Percepción Computacional	Apellidos: Anchondo Trejo Nombre: Victor Eduardo	20/Junio/2021

Función que recibe una matriz de elementos, lo convierte en un arreglo unidimensional, elimina los valores -1 y calcula la mediana de ese arreglo

```
In [3]: def calculateMedianForMatrix(matrix):
    # Convierte el arreglo bidimensional en un arreglo plano
    flattenWindow = np.concatenate(matrix)

    # Elimina los elementos -1 que puedan existir dentro del arreglo para que no interfiera con el calculo de la
    # mediana
    flattenWindow = [i for i in flattenWindow if i != -1]

    # Ordena el arreglo
    flattenWindow.sort()

    # Obtiene el indice del centro
    middleIndex = len(flattenWindow) // 2

    if len(flattenWindow) % 2 == 0:
        # Cuando el numero de elementos del arreglo es par, se calcula la media de los elementos centrales
        median = (flattenWindow[middleIndex - 1] + flattenWindow[middleIndex])//2
    else:
        # Cuando el numero de elementos del arreglo es impar, el elemento del centro es la mediana
        median = flattenWindow[middleIndex]

    return median
```

Función que escribe el valor en la "matrix" en el rango especificado

```
In [4]: def setValueInMatrix(matrix, value, fromColumnIndex, toColumnIndex, fromRowIndex, toRowIndex):
    totalColumns = matrix.shape[0]
    totalRows = matrix.shape[1]

    for x in range(fromColumnIndex, toColumnIndex):
        if totalColumns > x:
            for y in range(fromRowIndex, toRowIndex):
                if totalRows > y:
                    matrix[x][y] = value

    return matrix
```

Función que regresa una lista ordenada alfabéticamente de archivos con la extensión especificada en el directorio especificado

```
In [5]: def getTestPicturesInPath(path, extension):
    pictures = []
    for file in os.listdir(path):
        if file.endswith(extension):
            pictures.append(os.path.join(path, file))

    pictures.sort()
    return pictures
```

## Inicia lógica principal

```
In [6]: # Declaracion de la lista que contendra la lista de las listas de la imagen original y la imagen ya filtrada
pictureList = []

In [7]: # Obtiene e imprime la lista de imagenes jpeg dentro del directorio pictures/
picturePathList = getTestPicturesInPath("pictures/", ".jpeg")
print("Imagenes a procesar: " + str(len(picturePathList)))
```

Imagenes a procesar: 9

Asignatura	Datos del alumno	Fecha
Percepción Computacional	Apellidos: Anchondo Trejo Nombre: Victor Eduardo	20/Junio/2021

```
In [8]: # Ciclo para procesar cada una de las imágenes
for picturePath in picturePathList:
    print("Procesando: " + picturePath)

    # Lee la imagen en escala de grises (esta variable no se modificará)
    img = cv2.imread(picturePath, cv2.IMREAD_GRAYSCALE)
    # Lee la imagen en escala de grises (a esta variable es a la que se le harán modificaciones)
    cleanImg = cv2.imread(picturePath, cv2.IMREAD_GRAYSCALE)

    # Obtiene el número de columnas y filas
    cols = img.shape[0]
    rows = img.shape[1]

    # El tamaño de la venta se asigna a 5 para hacer una matriz de 5x5
    windowHeight = 5

    # Itera las columnas de la matriz de la imagen original con un step = 1
    for x in range(0, cols):
        # Define la variable con el valor de x + windowHeight
        toColumnIndex = x + windowHeight

        # Itera las filas de la matriz de la imagen original con un step = 1
        for y in range(0, rows):
            # Define la variable con el valor de y + windowHeight
            toRowIndex = y + windowHeight

            # Obtiene la sub matriz de windowHeight x windowHeight (3x3) de la matriz original
            matrixWindow = getSubMatrix(img, x, toColumnIndex, y, toRowIndex)

            # Calcula la mediana
            median = calculateMedianForMatrix(matrixWindow)

            # Guarda el valor de la mediana en la variable cleanImg
            cleanImg = setValueInMatrix(cleanImg, median, x, toColumnIndex, y, toRowIndex)

    # Agrega la imagen original y la imagen filtrada a la lista
    pictureList.append([img, cleanImg])

# Itera en paralelo las listas picturePathList (La lista de los paths de las imágenes) y PictureList
# (La lista de las imágenes limpias y originales)
for picturePath, filteredPicture in zip(picturePathList, pictureList):
    print("Mostrando imagen: " + picturePath)

    # Concatena la imagen original con la imagen filtrada para ser mostrada en una venta, una al lado de otra
    compare = np.concatenate((filteredPicture[0], filteredPicture[1]), axis=1)

    # Muestra la imagen original a la izquierda y la imagen filtrada a la derecha
    cv2.imshow(picturePath, compare)

    # Hace una pausa hasta que se presione cualquier tecla
    cv2.waitKey(0)

    # Cierra la ventana
    cv2.destroyAllWindows(picturePath)
```

Procesando: pictures/01.jpeg  
 Procesando: pictures/02.jpeg  
 Procesando: pictures/03.jpeg  
 Procesando: pictures/04.jpeg  
 Procesando: pictures/05.jpeg  
 Procesando: pictures/06.jpeg  
 Procesando: pictures/07.jpeg  
 Procesando: pictures/08.jpeg  
 Procesando: pictures/09.jpeg  
 Mostrando imagen: pictures/01.jpeg  
 Mostrando imagen: pictures/02.jpeg  
 Mostrando imagen: pictures/03.jpeg  
 Mostrando imagen: pictures/04.jpeg  
 Mostrando imagen: pictures/05.jpeg  
 Mostrando imagen: pictures/06.jpeg  
 Mostrando imagen: pictures/07.jpeg  
 Mostrando imagen: pictures/08.jpeg  
 Mostrando imagen: pictures/09.jpeg

Asignatura	Datos del alumno	Fecha
Percepción Computacional	Apellidos: Anchondo Trejo Nombre: Victor Eduardo	20/Junio/2021

## Resultados

En cada imagen esta la imagen original (con ruido) a la izquierda y la imagen a la derecha es a la que se le aplicó el filtro de la mediana.

