Visión Artificial

Título: Practica 8

Nombre: Mario Gerardo Casas Miramontes

Registro: 22310165

Objetivo

Objetivo: Dejar en la imagen solamente los bordes que deseamos y saber cuál es el mejor método.

Funcionamiento

El funcionamiento de esta practica se basa en la detección de bordes dentro de una imagen y a su vez en la comparación de los métodos con los que podemos realizar la detección de los bordes.

Método Laplaciano

Este método se basa en la detección de cambios de intensidad dentro la imagen, para ello primeramente realizamos un suavizado en la imagen en donde podemos establecer el parámetro del blur con una matriz de kernel, una vez hecho eso se aplica el método en laplaciano en la imagen con blur evitando la saturación de negativos. Para mostrar la imagen hacemos una conversión para pasar la imagen a blanco y negro.

Método Sobel:

Este método se puede dividir en los dos ejes para ver la diferencia de aplicación en cada uno. Lo primero es indicar a que eje queremos aplicarle el método a la imagen, también realizamos un método para la eliminación de saturación de negativos y usando también una matriz de kernel. Para mostrar la imagen hacemos una conversión para pasar la imagen a blanco y negro.

Método Canny

Este método tiene mas complejidad dentro de su lógica matemática, debido principalmente a que una vez que tenemos la imagen con blur aplicamos el método de Canny el cual mediante se saca un gradiente y mediante unos umbrales que definimos y que tan cercano este el gradiente a el valor de umbral máximo se tomara como verdadero, si se encuentra entre los umbrales se determinara su estado por vecindades y si no esta en el umbral mínimo no se muestra.

Código Fuente

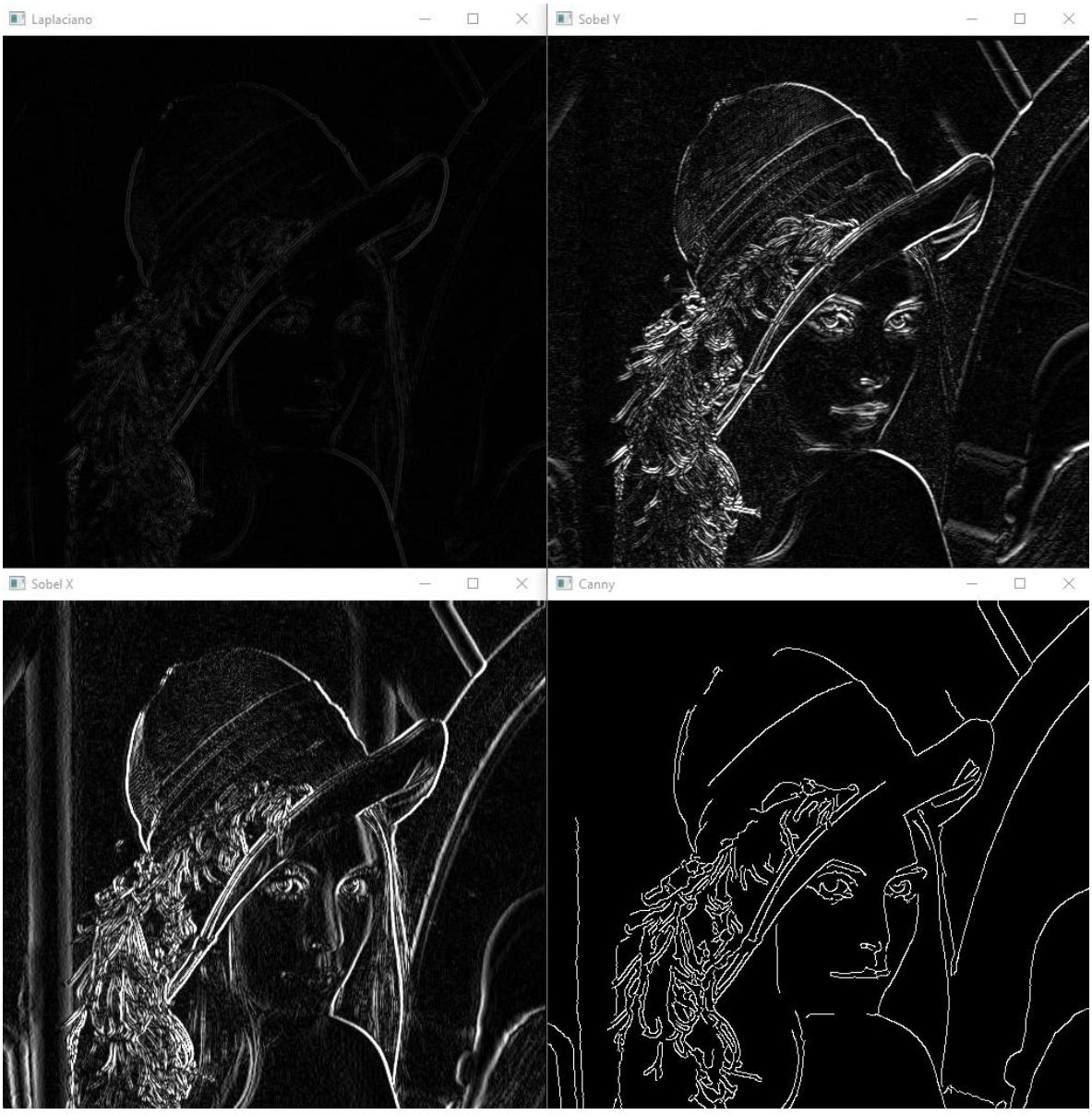
```
#Mario Gerardo Casas Miramontes 22310165

import cv2

import numpy as np


# Cargar imagen en escala de grises

img = cv2.imread('imagen.jpg', cv2.IMREAD_GRAYSCALE)
```

```
# Suavizar para reducir el ruido
blur1 = cv2.GaussianBlur(img, (3, 3), 0) #Metodo para suavizar la imagen


# Laplaciano
laplacian = cv2.Laplacian(blur1, cv2.CV_64F) #Se encarga de aplicar el metodo Laplacian , evitando la saturacion de metodos negativos
laplacian = cv2.convertScaleAbs(laplacian) #Usa los valores de blanco y negro para la imagen
# Sobel en X y Y
sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)  # dx=1, dy=0 y se encarga de los bordes verticales
sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)  # dx=0, dy=1 y se encarga de los bordes horizontales
# Convertir a uint8
sobelx = cv2.convertScaleAbs(sobelx)
sobely = cv2.convertScaleAbs(sobely)
# Aplicar suavizado
blur2 = cv2.GaussianBlur(img, (5, 5), 1.4) #Suavizamos la imagen
# Canny
edges = cv2.Canny(blur2, threshold1=50, threshold2=150)
#Este es el metodo mas complejo porque hace un calculo de gradiente en donde si supera el valor del Thershold
#Si se encuentra encima del 150 se dibuja, pero si es menor que 50 es ruido y se elimina.


# Mostrar resultados
cv2.imshow("Canny", edges)
cv2.imshow("Sobel X", sobelx)
cv2.imshow("Sobel Y", sobely)
cv2.imshow("Laplaciano", laplacian)
```

cv2.waitKey(0)

cv2.destroyAllWindows()

Resultado



LINK de la Practica 8

https://github.com/M-C117/VA/tree/main/Practica%208