INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

Aprendizaje de Maquina

REPORTE DE EXAMEN PRIMER DEPARTAMENTAL:

DETECTOR DE BORDES BASADO EN LA TEORIA CANNY

PRESENTADO POR:

CORTÉS MARTINEZ DILAN

PROFESOR:

OCTAVIO SANCHEZ GARCIA

MEXICO, 2022

Detector de bordes basado en la teoría Canny

C++

Cortés Martinez Dilan

Instituto Politécnico Nacional, Escuela Superior de Cómputo.

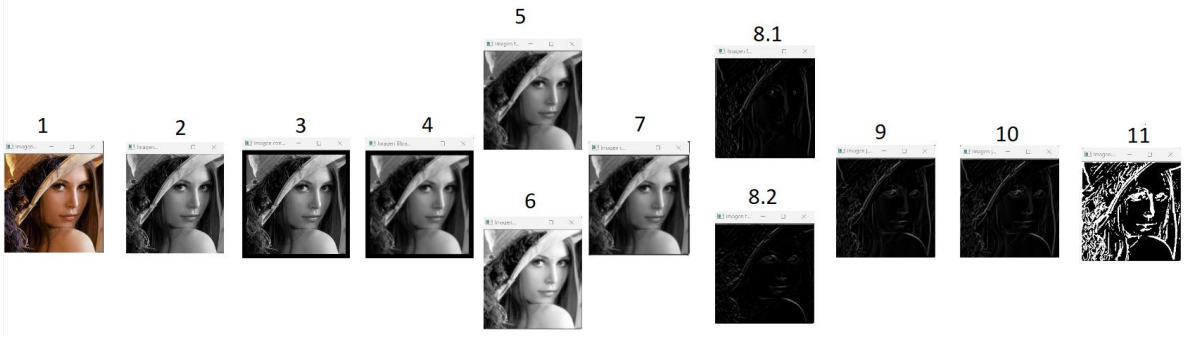
Visión Artificial

26 de octubre de 2022.

De acuerdo con las directrices propuestas por el docente, se elaboro un programa en C++ el cual es capaz de procesar una imagen dando como resultado una imagen de salida donde se aprecien, únicamente y de manera clara, los bordes de la imagen original sin afectar a la imagen de entrada. El proceso se llevo a cabo con la siguiente secuencia dada:

- Imagen original
- Imagen escala de grises
- Imagen suavizada (Kernel y sigma; variables)
- Imagen ecualizada
- Imagen resultada de aplicar |G|
- Imagen detección de borde Canny

Para poder mostrar estas pantallas como resultado, se siguió el siguiente proceso. Cada imagen es obtenida como resultado del proceso llevado a cabo en cada paso correspondiente. Es importante aclarar que cada imagen es la imagen de entrada para el proceso siguiente, por lo que todas fueron necesarias para obtener le imagen final **(11).**



1. Se carga la imagen con la que vamos a trabajar al programa, esto puede ser por medio de un path o asignando la imagen a la carpeta de “Archivos de recursos” en el proyecto en Visual Studio. **(Imagen de salida 1)**
2. Se realizo la conversión de color a escala de grises por medio del método NTSC el cual consiste en multiplicar cada canal (rojo, verde, azul) por un valor dado (0.299 * rojo + 0.587 * verde + 0.114 * azul) y al final sumarlos obteniendo así, el nuevo valor de cada píxel. **(Imagen de salida 2)**

3. Agregamos los bordes auxiliares necesarios para poder aplicar el Kernel (En este paso seria Gauss) a los bordes de la imagen, este valor es obtenido restando la dimensión del Kernel dada menos uno. Ejemplo: Longitud de Kernel = 21; Exceso = 20 píxeles para cada dimensión (10 de cada lado de la figura))
4. Aplicamos el Kernel a la imagen correspondiente en cada uno de los píxeles, exceptuando los excesos de la imagen original (Las filas y columnas de los bordes donde se agregaron puros 0's)
5. Realizamos una eliminación de bordes auxiliares (exceso) para obtener la imagen filtrada por medio de un Kernel Gaussiano. **(Imagen de salida 3)**
6. Realizamos una ecualización a la imagen por medio de su histograma, con el fin de ajustar su luminosidad. **(Imagen de salida 4)**
7. Agregamos los bordes necesarios para aplicar los Kernel de Sobel (al ser constante, siempre se agregan 2 pixeles a la figura por dimensión, uno por cada lado de la figura)
8. Aplicamos los Kernel de Sobel.
    1. Aplicamos el Kernel de Sobel establecido para los bordes del eje X a cada uno de los píxeles, exceptuando los excesos de la imagen original.
    2. Aplicamos el Kernel de Sobel establecido para los bordes del eje Y a cada uno de los píxeles, exceptuando los excesos de la imagen original.
9. Aplicamos la fórmula $|G| = \sqrt{Gx^2 + Gy^2}$, donde Gx representa los valores de los píxeles de la figura 7.1 (recordemos que esta contiene los bordes del eje X de la imagen suavizada) y Gy representa los valores de los píxeles de la figura 7.2 (recordemos que esta contiene los bordes del eje Y de la imagen suavizada)
10. Realizamos una eliminación de bordes auxiliares (exceso) para obtener la imagen filtrada por medio de un Kernel de Sobel $|G|$. **(Imagen de salida 5)**
11. Aplicamos un umbralizado a la imagen, el valor de está fue determinado a prueba y error, decidiendo dejar como valor de umbral el número 12.