Práctica 2
Detección de Bordes

Visión Computacional Aplicada a la Robótica
UNAM, 2023-2

Francisco Arturo Meza Torres
franciscomeza1918@gmail.com

Resumen

En esta práctica se implementa el filtro de Canny para detección de bordes en video. Primero se da una introducción al tema y las bases teóricas necesarias, posteriormente se desarrolla y analiza cada una de las partes del filtro, que son: filtro Gaussiano, convolución con el kernel, filtro de Sobel, magnitud y ángulo del gradiente, supresión de no máximos y doble umbral (histéresis). Para su análisis se realizan pruebas variando los umbrales y al final se reportan los resultados y se concluye.

1. Introducción

En la visión por computadora, la detección de bordes es un proceso que intenta capturar características de los objetos dentro de las imágenes, como discontinuidades y geometrías. El proceso de detección de bordes sirve para simplificar el análisis de imágenes, reduciendo drásticamente la cantidad de datos que hay que procesar, preserva información útil sobre los límites de los objetos [2]. El filtro debe ser eficiente y confiable, debido a que etapas posteriores de procesamiento dependen de él.

El detector de esquinas de Canny fue desarrollado en 1986 por F. Canny. También conocido como detector óptimo [3]. Los criterios principales son.

Baja tasa de error: lo que significa una buena detección de bordes existentes.

Buena localización: la distancia entre los píxeles de borde detectados y de los reales debe minimizarse.

Respuesta mínima: solo una respuesta del detector por borde.

2. Objetivos

- Aprender a utilizar las funciones de convolución de Open CV.
- Aplicar el filtro de Sobel para cálculo de gradientes.
- Implementar el detector de bordes de Canny.

3. Hipótesis

En la implementación del algoritmo se espera observar que conforme se van agregando las partes que componen el filtro, se identificará mejor cuales pixeles son bordes. Por otra parte, en la prueba de los umbrales, al aumentar el valor del umbral mínimo, se espera eliminar por completo los píxeles que no son bordes.

4. Marco teórico

Para la implementación del filtro de Canny es necesario obtener el operador de Sobel, que es un kernel que permite obtener las derivadas parciales, promediadas con un filtro Gaussiano. El kernel de las derivadas son los siguientes.

$$S_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{1.1}$$

$$S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{1.2}$$

El gradiente de cada píxel se calcula mediante la aproximación de las derivadas parciales.

$$\frac{\partial I}{\partial x} \approx I * S_x = G_x \tag{1.3}$$

$$\frac{\partial I}{\partial y} \approx I * S_y = G_y \qquad (1.4)$$

La forma polar del gradiente es la siguiente.

$$\nabla I = G_m \angle G_a \qquad (1.5)$$

Con la expresión (1.5) se puede obtener la magnitud del gradiente y la fase para cada píxel, lo cual es de utilidad para identificar que píxeles son bordes en una imagen o flujo de video.

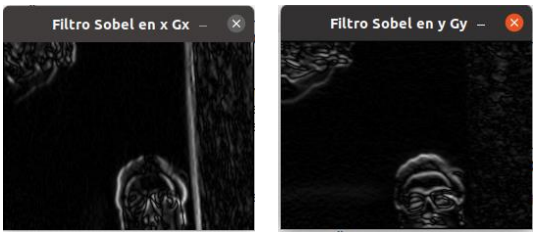
$$G_{m.i,j} = \sqrt{G_{x.i,j}^2 + G_{y.i,j}^2} \qquad (1.6)$$

$$G_{a.i,j} = a\tan 2(G_{y.i,j}, G_{x.i,j}) \qquad (1.7)$$

5. Desarrollo

El Filtro detector de bordes de Canny consta de los siguientes pasos.

1. Se obtiene la derivada en x y en y con un filtro Sobel y se promedia con un filtro Gaussiano.



a) b)

Figura 1. Filtro Sobel a) en x y b) en y.

En las funciones de las derivadas en x y en y, no se divide entre 8 para mantener escalada la respuesta y poder observar el comportamiento en los píxeles, si se divide entre 8, no se logrará percibir los cambios que vaya teniendo la imagen.

2. Se obtiene la magnitud y ángulo del gradiente.

a) b)

Figura 2. a) Magnitud y b) Ángulo del gradiente.

3. Se aplica la supresión de no máximo con base en el ángulo del gradiente.

Figura 3. Supresión de no máximos.

4. Se aplica el doble umbral con base en la magnitud del gradiente, lo que da como resultado el filtro detector de bordes de Canny. En la siguiente figura también se muestra una versión donde los bordes detectados se saturan.

a) *b)*

Figura 3. a) Filtro Canny 1. b) Filtro Canny 1 Saturado.

Por último se muestra el resultado del Filtro cuando las derivadas de x y de y, se dividen entre 8.

a) b)

Figura 3. a) Filtro Canny 2. b) Filtro Canny 2 Saturado.

6. Análisis de resultados

Se realizaron pruebas variando el umbral mínimo del doble umbral, utilizando los valores de las tablas 1, obteniendo los siguientes resultados.

Tabla 1. Parámetros del doble umbral para la prueba .

Umbral mínimo	0	5	10	15
Umbral máximo	255	255	255	255

a) b)

Figura 3. Filtro Canny. a) Con Umin=0. b) Con Umin=5.

a) b)

Figura 3. Filtro Canny. a) Con Umin=10. b) Con Umin=15.

Con el análisis realizado, se logró observar que al aumentar el valor del umbral mínimo, se eliminan los bordes menos significativos para la magnitud del gradiente. Cabe mencionar que al variar el umbral Umax no se observaron grandes cambios, hasta estar muy cerca de Umin, alrededor del valor 40. Los valores de la magnitud del gradiente se encuentran entre 0 y 40 para el flujo de video utilizado en este trabajo.

7. Conclusión

Para la implementación del Filtro de Canny fue necesario comprender y programar en Python las etapas del algoritmo de Canny, donde se observó que una herramienta fundamental es la función de convolución en 2 dimensiones que proporciona Open CV. Por otra parte, la serie de pasos que se siguieron para la implementación del detector de bordes, fue de utilidad para ir analizando cada parte del algoritmo. En conclusión, con el desarrollo y análisis realizado se logró cumplir con los objetivos e hipótesis, obteniendo como resultado un flujo de video con el filtro de Canny aplicado.

8. Referencias

[1] Savedra, J. C. S., González, R. O., Ortega, V. H. G., & Tovar, R. H. (2018). Detección de bordes en tiempo real empleando el filtro de Sobel y tecnología reconfigurable. Pistas Educativas, 35(108).

[2] J. Canny (1986). A Computational Approach to Edge Detection, in IEEE Transactions on Pattern Analysis and Machine Intelligence, 8 (6), 679-698.
doi: 10.1109/TPAMI.1986.4767851.

[3] OpenCV. Open Source Computer Visión. Consultado en: https://docs.opencv.org/4.x/da/d5c/tutorial_canny_detector.html, el 9 de abril de 2023.