MEMORIA T3 CONTORNOS

VISIÓN POR COMPUTADOR

Iñigo Alonso Ruiz – 665959

Alejandro Dieste Cortés - 541892

TABLA DE CONTENIDOS

Índice

T3-CONTORNOS

1. Cálculo del gradiente

Como dice el enunciado, gradiente horizontal y vertical, y módulo y orientación del gradiente. Hemos probado con dos operadores distintos para comparar resultados y poder elegir el qué mejor resultados da. Estos operadores son el de Sobel y el de Canny.

1.1 SOBEL

Antes de aplicar el operador de Sobel, hemos aplicado un Filtro Gaussiano con la función de OpenCV `GaussianBlur.` Esto lo hacemos debido a que se genera ruido en los cálculos del gradiente, del módulo y orientación de este. Este filtro tiene como parámetro sigma, que se puede variar.

Una vez aplicado el filtro, se utiliza la función de OpenCV `Sobel` para calcular el gradiente en 'x' y en 'y'.

Después recorremos cada pixel para calcular el módulo y la orientación del gradiente. Se calculan a partir del gradiente ya obtenido de 'x' e 'y'.

$$\textit{Gradiente} \quad \vec{\nabla} f(x,y) = (\nabla_x, \nabla_y) = \left(\frac{\partial f}{\partial x} \quad , \quad \frac{\partial f}{\partial y} \right)$$

$$\textit{Módulo} \quad |\nabla f| = \sqrt{(\nabla_x)^2 + (\nabla_y)^2}$$

$$\textit{Orientación} \quad \theta = \operatorname{atan} 2(\nabla_y, \nabla_x)$$

Ilustración 1 Fórmula del gradiente, módulo y orientación

Para comprobar los resultados, mostramos cada característica en una imagen. Para ello realizamos algunos ajustes en el gradiente y la orientación para poder mostrarlo por pantalla. Al gradiente se le divide por 2 y se le suma 128. Y a la orientación se le divide por Pi y se multiplica por 128. Estos ajustes se guardan en variables nuevas, ya que los cálculos para la detección del punto de fuga se realizan con los valores originales.

1.2 CANNY

Mascara gaussiana, vectores de gaussiana y de derivada de gaussiana (Constante?)

El operador de Canny optimiza tres criterios: robustez de detección frente al ruido, precisión en la localización y unicidad en la respuesta.

El cálculo del gradiente lo hacemos mediante dos vectores máscaras para optimizar el espacio. Estos vectores se calculan en función de una distribución normal. Y la sigma de esta distribución es variable. El tamaño de los vectores lo hemos establecido en sigma * 5, que es el que mejor resultado nos ha dado.

El primera vector (G') es la derivada de la distribución normal. Se calcula de la siguiente la siguiente forma. Siendo sigma variable.

$$G'_\sigma(x) = \frac{-x}{\sigma^2}\exp\left(\frac{-x^2}{2\sigma^2}\right)$$

Ilustración 2 Derivada de la gaussiana

Para el cálculo del segundo vector (G), hemos utilizado la fórmula de la distribución gaussiana. La media es cero, y la sigma variable.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{\frac{(x-\mu)^2}{2\sigma^2}}$$

Ilustración 3 Distribución gaussiana

Una vez calculados los dos vectores, se calculan los gradientes 'x' e 'y'. Para ello hemos utilizado la función de OpenCV `sepFilter2D` que aplica el filtro dados dos vectores máscara. Para el cálculo de 'x' se multiplica el vector derivada y el vector de gaussiana, y para el gradiente 'y' se multiplica el vector de gaussiana transpuesto por el vector derivada transpuesto y negado. Como en la teoría.

$$\nabla_x = G'_\sigma(x) * G_\sigma(y) * f(i,j)$$
$$\nabla_y = G_\sigma(x) * G'_\sigma(y) * f(i,j)$$

Ilustración 4 Calculo de los gradientes a partir de los dos vectores

El siguiente paso es calcular el módulo y la orientación del gradiente. Los calculamos de la misma manera que hemos hecho con el operador de Sobel. También hemos aplicado los ajustes necesarios al gradiente y la orientación para poder mostrarlos por pantalla en una imagen.

Calculo de gx gy mod ori de todos los pixeles de la imagen

Ajuste para poder mostrarlo en imagen

1.3 TIPOS DE DATOS

Como bien advertía el enunciado de la teoría tuvimos que tener en cuenta los tipos de datos que estábamos utilizando y almacenando en las matrices para el cálculo de los gradientes, el módulo y la orientación.

Tanto en el caso del operador de Sobel como de Canny hemos tenido las siguientes consideraciones. Las matrices que almacenan los gradientes, el módulo y la orientación (con los que más tarde se hacen los cálculos para sacar el punto de fuga) son de tipo float. Sin embargo, las matrices que se utilizan para mostrar los gradientes, el módulo y la orientación, son de tipo char. A la hora de calcular estos, al salir los valores tanto del blurGauss para Sobel como sepFilter2D para el operador Canny son de tipo short. Por lo que estos cálculos se calculan como short, para luego almacenarlos en float o char dependiendo de la matriz.

En el caso del operador de Canny, los tipos de los vectores máscara son short. Como se ha comentado sepFilter2D trabaja con short.

1.4 CONCLUSIÓN

Tras comparar las imágenes resultantes del operador de Sobel y de Canny, llegamos a la conclusión de que el operador de Canny sacaba resultados mejores. El resultado es igual al de los ejemplos de la teoría.

Por ello hemos utilizado los resultados del operador de Canny para los cálculos del punto de fuga, tanto para una fotografía centrada como una en la que la cámara este inclinada.

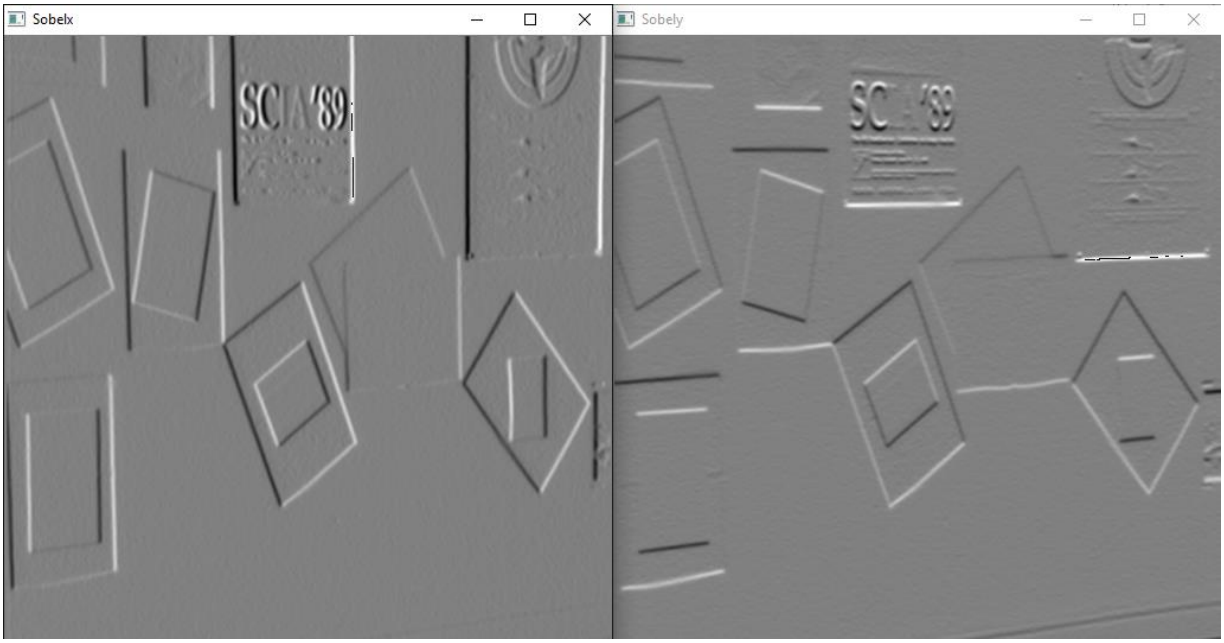

Ilustración 2 Sobel grad x e y

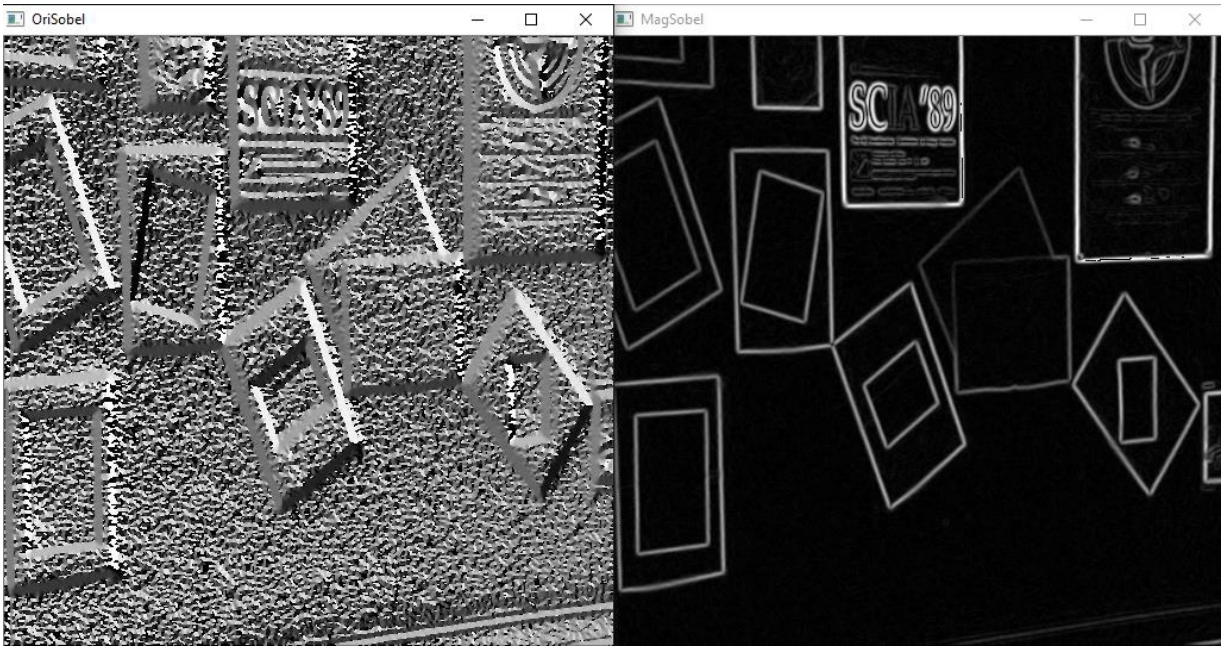

Ilustración 3 Sobel orientación y magnitud

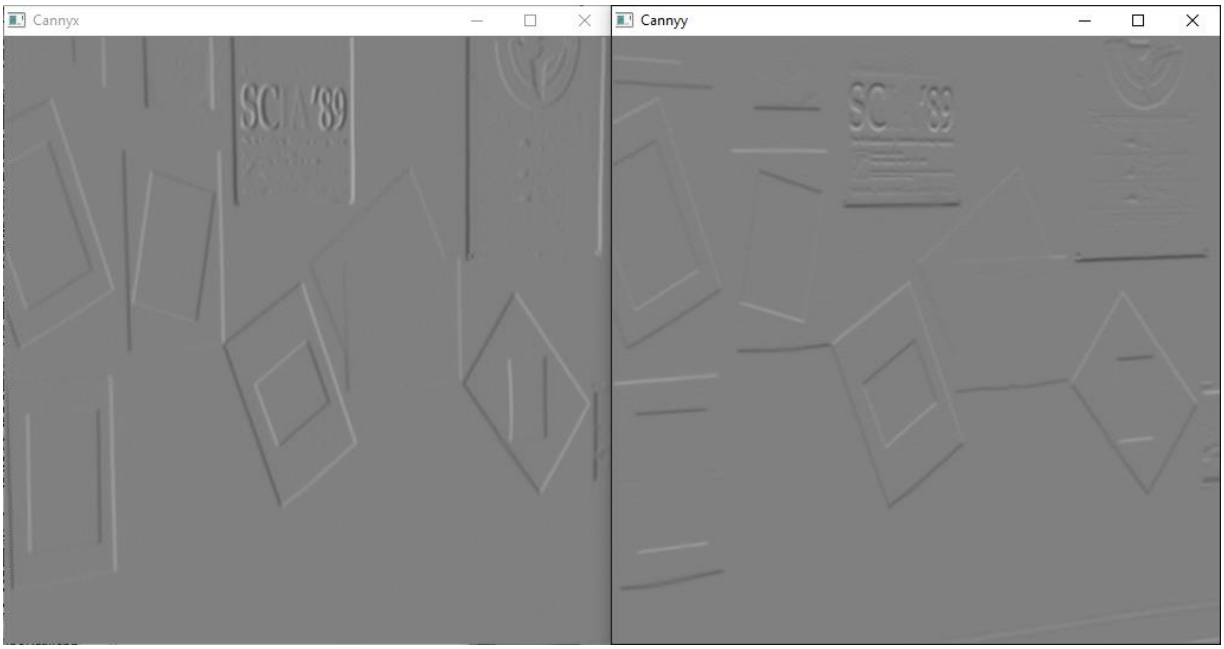

Ilustración 4 Gradiente 'x' e 'y' con el operador de Canny

Ilustración 5 Orientación y módulo del gradiente con el operador de Canny

2. Cálculo del punto de fuga en imagen centrada

Para hallar el punto de fuga utilizamos el algoritmo de la teoría, que se basa en utilizar el módulo y orientación del gradiente de la imagen. Se recorren todos los pixeles y si el módulo supera un umbral, que hemos establecido en 35, se vota la recta correspondiente a la orientación y a rho, que se saca a partir de la posición actual en la que estamos.

La votación consiste en calcular la intersección de dicha recta con la línea central de la imagen. Ya que como está centrada, la línea del horizonte está en el centro de la imagen. En el punto de dicha recta en el que intersecta se le suma un voto. Finalmente se coge el pixel que más votos tenga, y ese será el punto de fuga.

Como optimización, hemos añadido un filtro al algoritmo de la votación, descartando las líneas que son verticales y horizontales (con un error de 0.1), ya que estas no aportan nada a la hora de calcular el punto de fuga.

Aquí los ejemplos de imagen con cámara centrada pasillo2 y pasillo3

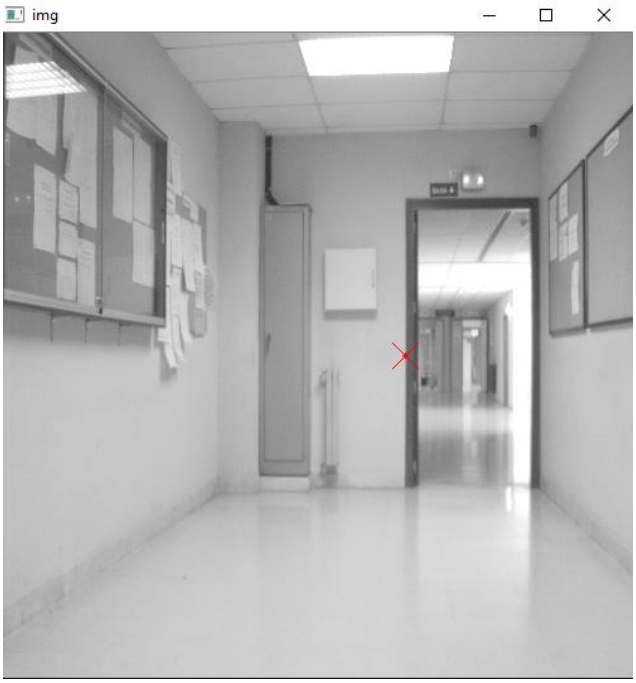

Ilustración 6 Pasillo 2

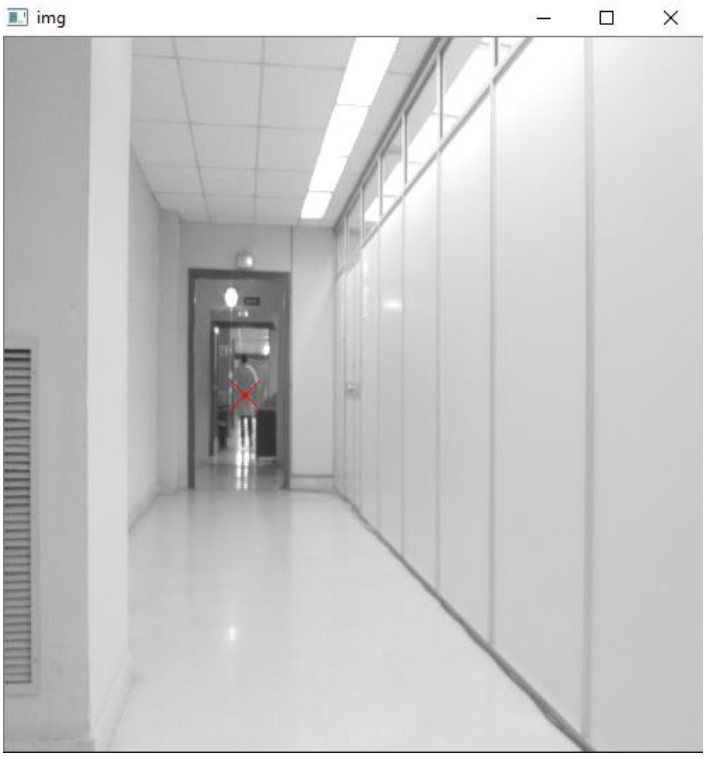

Ilustración 7 Pasillo3

3. Punto de fuga con cámara no horizontal

Para el cálculo del punto de fuga utilizamos también los resultados tras aplicar el operador de Canny para hallar el gradiente, su módulo y la orientación.

Pero ahora el punto de fuga no tiene por qué estar en el centro de la imagen, ya que la cámara está inclinada. Así que no podemos utilizar el mismo algoritmo de antes, hemos hecho algunas modificaciones.

El algoritmo consiste en: para cada pixel nos guardamos su recta correspondiente, establecida por su orientación y rho (que se halla a partir de la posición en la que estamos). Realizamos también el filtro de rectas verticales y horizontales. Una vez guardadas todas las rectas se hace la votación. La votación se realiza 'num_rectas * log(num_rectas)' veces. Esta consiste en elegir dos rectas aleatorias de las almacenadas, se comprueba que no sean la misma, que no se parezcan en orientación y se calcula su intersección. Si la intersección se produce dentro de la imagen, entonces se vota ese punto, incrementando en uno su valor acumulado. Si no se cumple ninguna de estas condiciones no se hace nada.

Una vez realizada la votación, el pixel que más votos tenga es el punto de fuga.

Ejemplos con comparación con el algoritmo para cámara horizontal. Cruz roja algoritmo para cámara horizontal. Cruz rosa algoritmo para cámara inclinada.

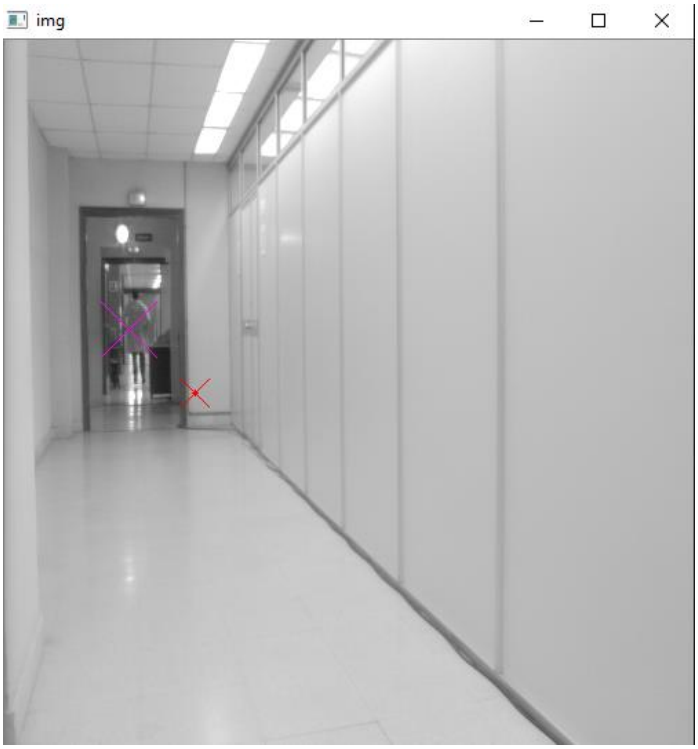

Ilustración 8 Pasillo 1

Ilustración 9 Pasillo2 votando num_rec^2 veces

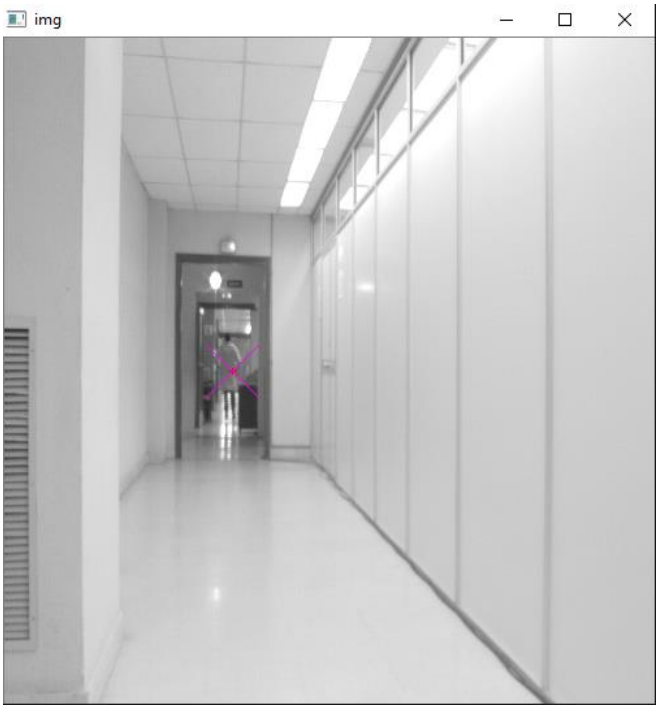

Ilustración 10 Pasillo 3

Como se puede comprobar, el resultado es el mismo para Pasillo2 y 3. Y para el pasillo 1, que la cámara está inclinada, el resultado con este algoritmo es mejor que el anterior.

Para hacerlo en tiempo real, simplemente tenemos un bucle que coge la imagen de la webcam, y aplicamos el operador y el algoritmo de votación a dicha imagen.