Práctica 2.
Contornos

Alberto Lardiés, 735976
Devid Dokash, 780131

26 de marzo de 2023

Índice

1. Decisiones de diseño generales

El objetivo de este trabajo es desarrollar con OpenCV un programa capaz de detectar contornos y utilizarlos para obtener el punto de fuga de un pasillo:

1. Para el suavizado de las imágenes, se ha usado la función `cv.GaussianBlur(frame,(3,3),0)`, con un kernel de 3x3 debido a que era el kernel con mejores resultados.
2. Para el calculo de los gradientes se ha hecho uso del operador Sobel implementado desde OpenCV con un kernel de 3x3, con la misma motivación que el punto anterior.
3. Para el operador de Canny, se hace uso del operador de Sobel, que utilizará el módulo y la orientación para un módulo más preciso.
4. Para la transformada de Hough, las votaciones se realizan mediante intersecciones, mediante las coordenadas polares de la linea horizontal del centro y las coordenadas de cada punto, el punto en coordenadas cartesianas que más intersecciones tenga, será el punto de fuga.
5. Se ha implementado una pequeña interfaz que permite subir una imagen o visualizar los efectos en vivo.
6. En esta entrega no podrá haber vídeo.

2. Gradientes

En esta primera parte, se han implementado dos operadores, inicialmente, el de Sobel/Schorr:

1. Se suaviza la imagen mediante una matriz convolucional de tamaño 3x3 aplicando un filtrado de blur Gaussiano y así reduciendo el ruido de los píxeles de la imagen.

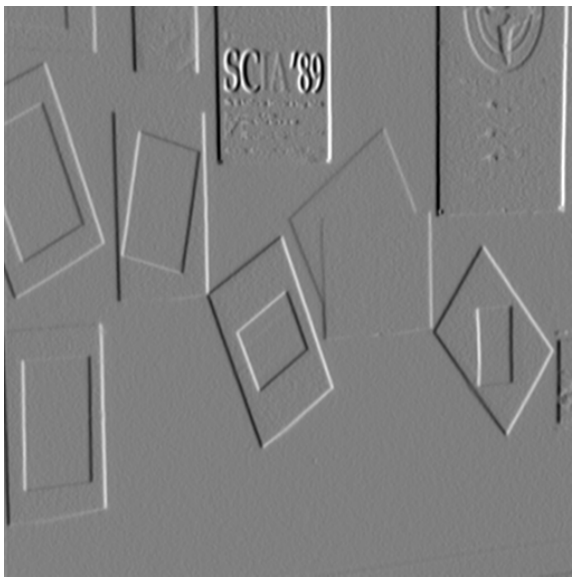

Figura 1: Gradiente en el eje X.

2. Se hallan los distintos gradientes (X e Y) de la imagen mediante el operador de Sobel ya implementado de OpenCV, mediante un kernel de 3x3 para ambos casos.

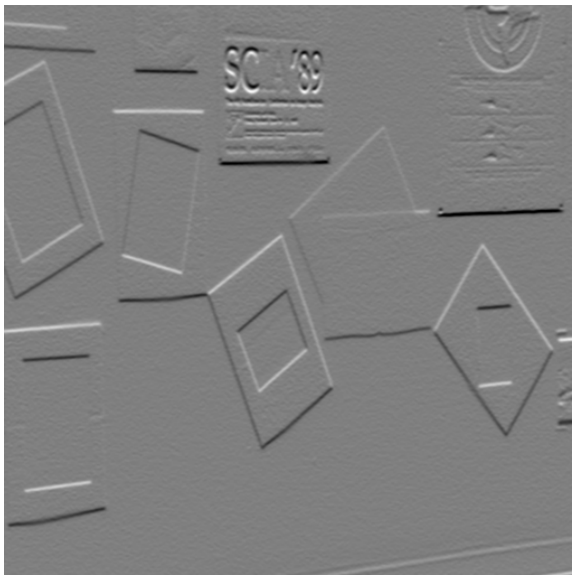

Figura 2: Gradiente en el eje Y.

3. Se hallá de manera adicional, el gradiente en ambas direcciones (XY), que es la combinación de las dos, `cv.addWeighted(gX, 0.5, gY, 0.5, 0)`.

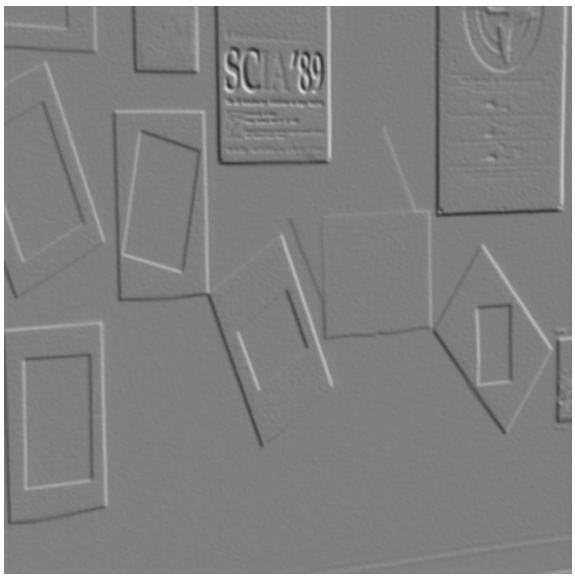

Figura 3: Gradiente en el eje X e Y.

4. Se calcula el modulo mediante los gradientes en X e Y, $modulo = sqrt(G_x^2 + G_y^2)$:

Figura 4: Módulo del gradiente con Sobel

5. Finalmente, se calcula la orientación, en radianes, mediante los gradientes en X e Y, $orientacion = arctan2(G_y, G_x)$:

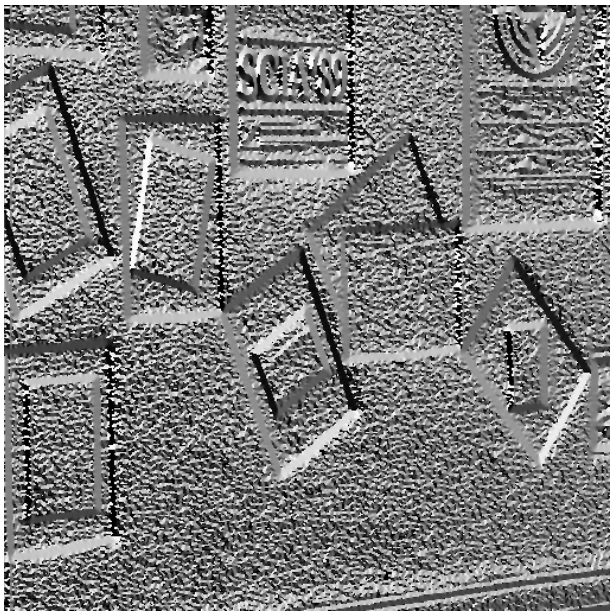

Figura 5: Orientación del gradiente

Una vez implementado Sobel, se implementa Canny:

Figura 6: Módulo del gradiente con Canny.

1. La parte del suavizado, los gradientes, el módulo y la orientación quedan abstraídas dentro del operador de Sobel implementado.
2. Se asegura que los ángulos de la orientación del gradiente queden entre -180 y 180 grados. Ahora en adelante, se hará uso del módulo obtenido con Sobel y se le aplicarán las siguientes etapas.
3. Se aplica *Supresión de no máximos*, de esta manera, se eliminan todos aquellos píxeles de los bordes que son suaves limpiando los bordes y quedando marcados solo por aquellos píxeles con mayor intensidad.
4. Se aplica la umbralización de la imagen mediante *Double threshold*, que consiste en el uso de dos umbrales mínimos y máximos para eliminar aquellos bordes que estén fuera de este rango que se comporten más como ruido que como bordes importantes.
5. Finalmente, se ha de aplicar la *Hysteresis*, técnica para conectar los bordes inconexos y eliminar aquellos que se consideren ruido. Esta etapa ha dado problemas y no ha sido posible implementar esta etapa, aun así, los resultados han sido buenos.

Observando los valores mínimos y máximos, se puede observar que son bastante disparas:

- En el caso de los gradientes, esto se debe a la máscara del operador Sobel, que en el peor de los casos, hace que los valores de los mismos estén entre -4*255 y 4*255.
- Aún transformando las matrices de gradiente, los valores sufrían de desbordamiento, por lo que se ha necesito implementar y aplicar una propia función de normalización de los datos.
- En el caso del módulo, se observa que los valores de Canny están dentro del rango esperado, esto se debe a la normalización de los mismos, algo que con Sobel no ocurre, con un máximo de 454.

Matriz	Sobel		Canny	
	Mínimo	Máximo	Mínimo	Máximo
Gradiente en X	-75.5	315.0	-75.5	315.0
Gradiente en Y	-55.0	355.0	-55.0	355.0
Gradiente en XY	-260.0	272.0	-260.0	272.0
Módulo	0.0	454.004	25	255
Orientación	-128.0	127.899	-128.0	127.899

Figura 7: Mínimos y máximos de las diferentes matrices.

3. Punto de fuga

Para el cálculo del punto de fuga se ha implementado una función, $Hough_transform$, que dados el módulo y la orientación del gradiente de una imagen y un umbral de threshold, devuelve el punto de fuga de la imagen según los bordes calculados por canny. Para ello, para cada píxel, si su módulo de gradiente es mayor que el umbral, se calcula la recta que pasa por ese punto, de acuerdo a su ángulo de gradiente, en coordenadas polares. Si intersecta con la línea del horizonte de la imagen, se añade un voto al punto del horizonte donde intersectan. Al final devuelve el punto del horizonte que más votos tenga.

Después, se dibuja en la imagen original una cruz con centro en el punto de fuga para marcarlo.

Los resultados son los siguientes:

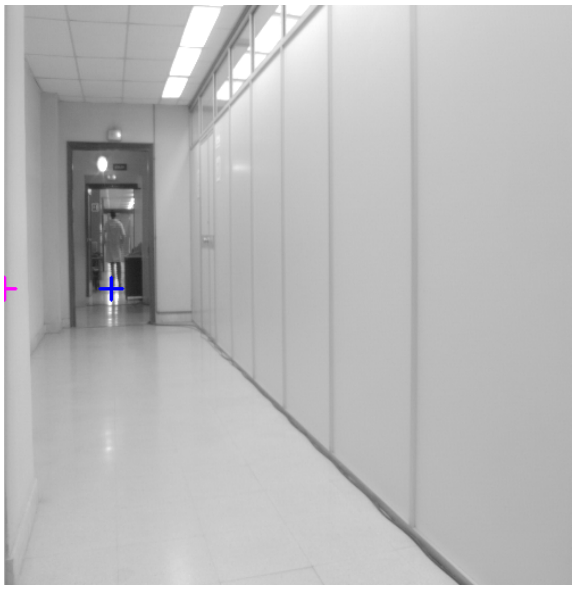

Figura 8: Punto de fuga en pasillo1.pgm.

Figura 9: Punto de fuga en pasillo2.pgm.

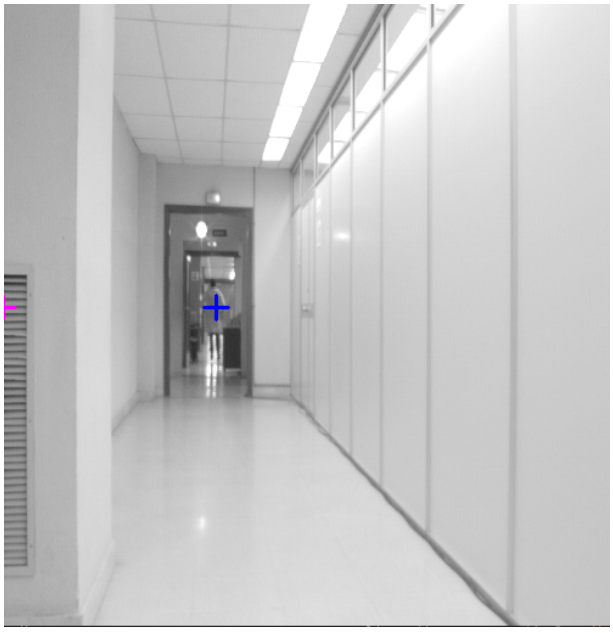

Figura 10: Punto de fuga en pasillo3.pgm.

Como se puede observar, en las imágenes pasillo1.pgm y pasillo3.pgm se encuentra el punto de fuga del pasillo se encuentra con el módulo y la orientación de Sobel (cruz azul) pero no con Canny (cruz rosa) mientras que para pasillo2.pgm es al revés. En el caso de la imagen pasillo2.pgm se cree que es porque las líneas del pasillo son más suaves y Sobel no las detecta bien mientras que Canny las detecta sin problemas.

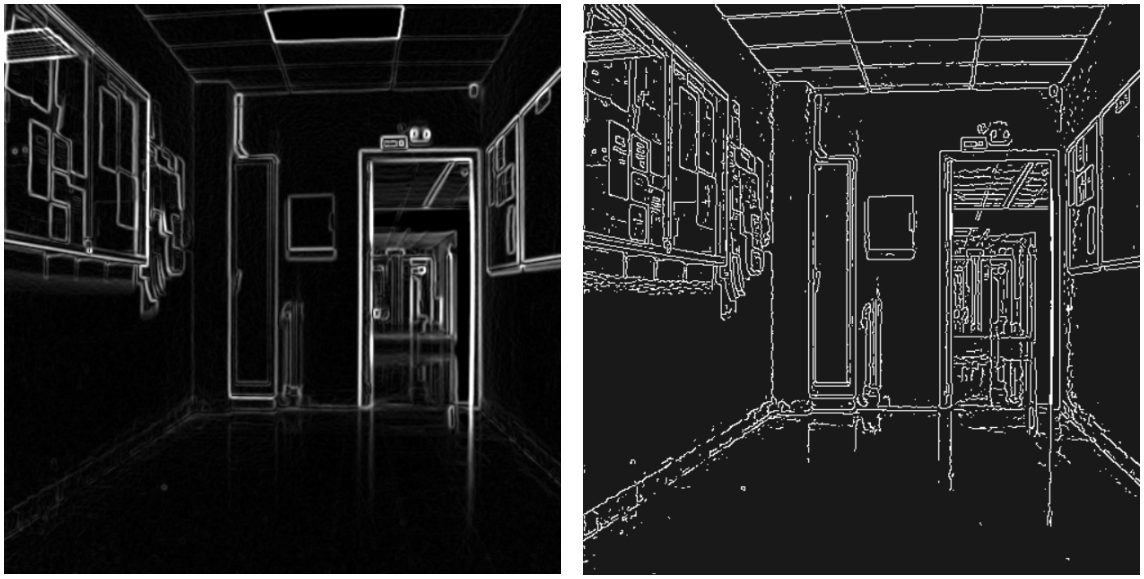

Figura 11: Comparación de los modulos de Sobel (izquierda) y Canny (derecha) del pasillo 2.