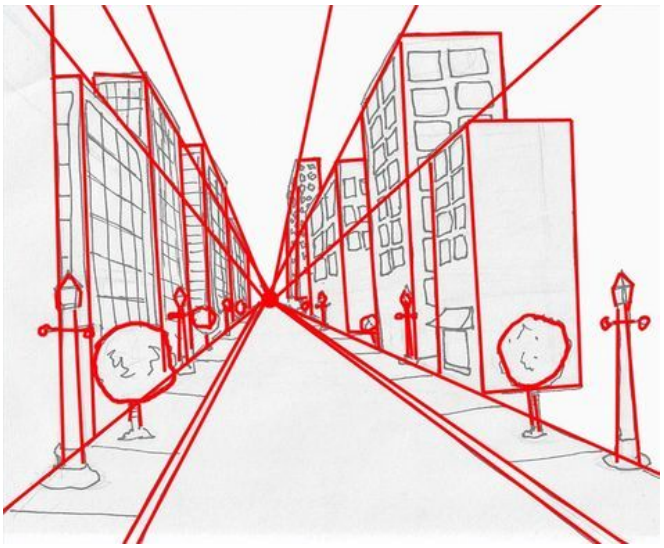
Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Vanishing Point Detection Project



Autores:

Félix Ozcoz Eraso (801108)

Victor Manuel Marcuello Baquero (741278)

24 de Marzo 2024

Índice

Descripción de la aplicación

Aplicación configurable de línea de comandos que permite detectar y visualizar el punto de fuga[1] de una imagen y las rectas paralelas que confluyen a él.

Implementación

Esta sección recoge las especificaciones de desarrollo del código implementado correspondiente a los operadores de Sobel y Canny y la detección del punto de fuga basado en un sistema de votación de líneas que votan por puntos utilizando la transformada de Hough.

Subapartados contenidos:

- Sistema de referencia
- Cálculo de gradiente
- Operador de Sobel
- Operador de Canny
- Hough Transform

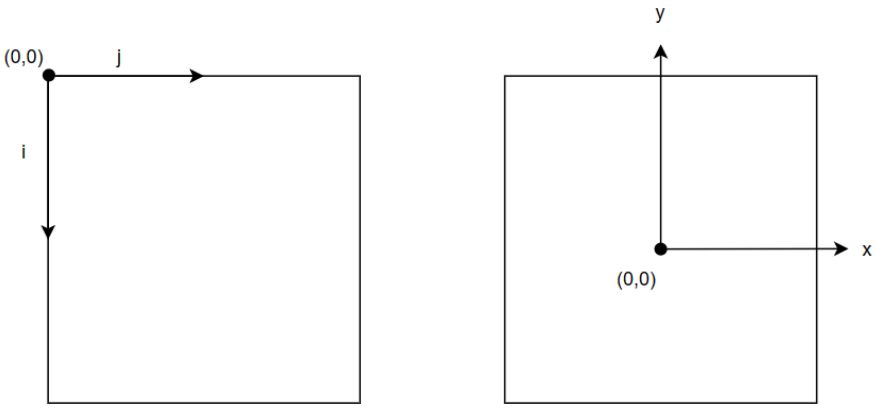
Sistemas de referencia

OpenCV utiliza una representación matricial para las imágenes cuyo origen de coordenadas está situado en la esquina superior izquierda mientras que el espacio espacio de Hough requiere que el origen de coordenadas se localice en el centro de la imagen.

Sean *h* y *w* la altura y ancho de la imagen, respectivamente. Cada punto de la imagen se centra de acuerdo a la siguientes ecuaciones siendo *(x,y)* las coordenadas en el espacio de Hough:

$$x = j - w/2$$
$$y = h/2 - i$$

Figura 1.
Ecuaciones para traslación de coordenadas de representación matricial a espacio de Hough



a) Sistema de referencia en OpenCV

b) Sistema de referencia en el espacio de Hough

Figura 2. *Sistemas de referencia*

Cálculo de gradiente

Parte del preprocesado de detección del punto de fuga consiste en extraer los edges de la imagen a partir de un proceso de convolución y un posterior umbralizado de la imagen u otras técnicas que permitan su extracción.

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

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

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

Figura 3.
Ecuaciones analíticas de cálculo de gradiente

Operador de Sobel

Los **kernels utilizados** tienen un tamaño 3x3 con valores fijos como se muestra en la figura.

Este kernel aumenta la sensibilidad del cambio de valores de la imagen en la sección central del kernel mientras que reduce el ruido en los extremos permitiendo amplificar las diferencias de intensidad de los píxeles en la dirección buscada.

Notar que la dirección de búsqueda es de negativos a positivos tal y como se ha descrito en base al sistema de referencia supuesto.

−1	0	1
−2	0	2
−1	0	1

a) Kernel dirección x

1	2	1
0	0	0
−1	−2	−1

b) Kernel dirección y

Figura 4.
Kernels direccionales de Sobel usados

OpenCV ofrece la función *filter2D* que dado un kernel realiza el **proceso de convolución** en dos dimensiones, así la obtención de los gradiente horizontal y vertical es directa, de igual forma los son la orientación y la magnitud.

Respecto a los valores obtenidos tras la convolución, los **valores máximos y mínimos** vienen determinados por el tamaño de la imagen y el tamaño del kernel.

Se supone una imagen en escala de grises con un rango de colores [0,255], es decir, 8 bits, y por ejemplo se busca el gradiente horizontal.

El valor máximo obtenido para este gradiente puede ser el doble del valor del píxel, que corresponde a la transición de oscuro a claro suponiendo que el oscuro es 0 y el claro 255.

Por otro lado , el mínimo es -510 que ocurre en transiciones de blanco puro a negro puro.

En definitiva el rango de valores sería [-510, 510], por lo tanto se debe adaptar el tipo de la matriz contenedora de los valores, en este caso valdría con 16 bits.

Para el caso de la magnitud del gradiente puede ser mayor ya dado que se elevan al cuadrado los valores de los gradientes horizontales y verticales llegando a un valor máximo de M = 2 * sqrt(255^2+255^2) definiendo el rango de valores [0, M].

Operador de Canny

Los gradiente direccionales se obtienen a partir de una composición entre el filtro gaussiano unidimensional, la primera derivada gaussiana y los datos de la imagen original.

La ecuación analítica de la figura resume el procesamiento que consiste en un primer suavizado direccional de la imagen que pretende preservar aquellas estructuras en la dirección de búsqueda del gradiente, seguido del cálculo del gradiente para encontrar la magnitud y dirección de cambio de la intensidad en cada píxel.

Ambas operaciones gaussianas son parametrizadas por *σ (sigma).* En la primera derivada sigma determina la sensibilidad a los cambios rápidos en la señal.
A medida que *σ* aumenta, el gradiente resultante se suaviza y se vuelve menos sensible a los cambios rápidos de intensidad.
Un valor mayor de *σ* resulta en un gradiente más suave y menos propenso a respuestas "ruidosas".

En el filtro gaussiano unidimensional controla la "anchura" de la campana gaussiana.
A medida que *σ* aumenta, la campana se hace más ancha y suave.
Un valor mayor de *σ* resulta en un mayor suavizado de la señal

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

Figura 5. *Cálculo de gradiente en x e y*

Una vez calculado el gradiente horizontal y vertical, calcular la magnitud y la dirección es directo.
De la misma forma que con el operador Sobel, la convolución se ejecuta con *filter2D* para la dirección de búsqueda requerida.

Proceso de obtención de punto de fuga

[Dinámica de votación: líneas que votan por puntos del horizonte]

La versión implementada supone las siguientes premisas:

- Existe un sólo punto de fuga
- El punto de fuga está situado en el horizonte de la imagen
- El foco de la cámara está alineado con el centro de la imagen

El algoritmo sigue estos pasos:

Preprocesamiento:

1. Thresholding de la imagen
2. Eliminar líneas verticales y horizontales

Cálculo de punto de fuga:

1. Recorrer la imagen en forma matricial
2. Trasladar el punto al espacio de Hough
3. Determina las rectas que lo contienen
4. Localizar punto de intersección con el horizonte
5. Devolver punto de intersección a forma matricial
6. Contabilizar voto
7. El más votado es el punto de fuga

Detectar edges

Una vez extraído el gradiente con los operadores implementados se ha aplicado la técnica de thresholding para eliminar aquellos píxeles con intensidad irrelevante, es decir, aquellos que no pertenecen a bordes.

Eliminar edges horizontales y verticales

El siguiente paso es eliminar los puntos que no contribuyen a la localización del punto de fuga, esto son aquellos cuya orientación del gradiente es paralela al eje x o y. Se determina una margen (alfa) de variación sobre los ejes para tener mayor alcanzabilidad ya que la dirección de esos puntos no será exactamente paralela a uno de los ejes. Este planteamiento se muestra en la siguiente figura:

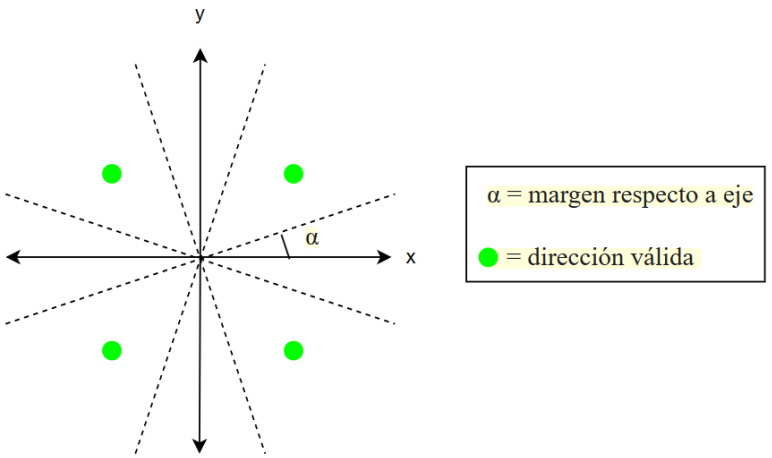


Figura 6.
Rango de orientaciones de gradiente permitida para valoración como punto de interés

Cálculo de rectas en espacio de Hough

A continuación se traslada traslada cada punto desde el sistema de referencia matricial de openCV al espacio de Hough como se ha descrito previamente en la subsección Sistemas de referencia.

Sea *(x,y)* el punto trasladado al espacio de Hough, ahora se determina una de las rectas a la que pertenece definida por los parámetros (ρ, θ) según la ecuación de una recta en el espacio de Hough determinada por la siguiente ecuación:

$$\rho = x * cos(\theta) + y * sin(\theta)$$

Figura 7.
Ecuación de una recta en el espacio de Hough definida por (ρ, θ)

Obtenidos los parámetros (ρ, θ) se determina el punto de corte con el horizonte en el espacio de Hough. Detectado el punto se traslada a la representación matricial y se contabiliza el voto sobre el punto de corte.

El punto del horizonte más votado es el punto de fuga.

Resultados

Los resultados mostrados corresponden al gradiente ajustado para ser ploteado tal y como se indica en las diapositivas, de esta forma se puede comprobar su corrección.

Contenido:

- Operador de Sobel
  - imagen prueba: *poster.pgm*

- Operador de Canny
  - imagen prueba: *poster.pgm*

- Transformada de Hough
  - imagen 1: *pasillo2.pgm*
  - imagen 2: *pasillo3.pgm*
  - Imagen 3: *iluminacion.jpg*

Este enlace redirecciona a un repositorio de Github donde se pueden localizan las imágenes y el código desarrollado: [repositorio]

Imágenes de prueba para el operador de Sobel y Canny

Figura 8.
Imagen de prueba
para operadores

Operador de Sobel

Imagen de prueba: *poster.pgm*

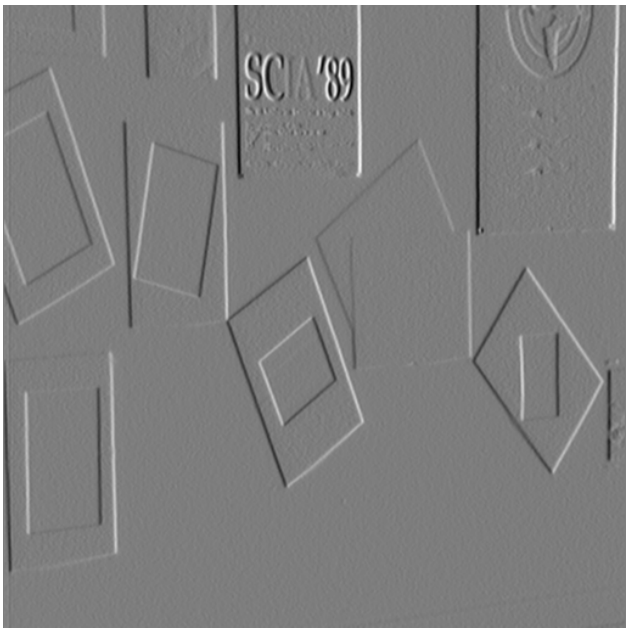


Figura 9.
Gradiente de Sobel en x

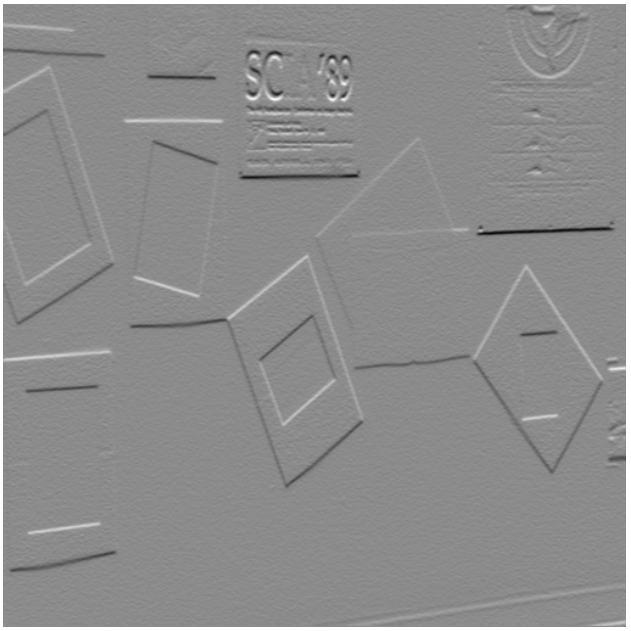


Figura 10.
Gradiente de Sobel en y

Figura 11.
Módulo del gradiente Sobel

Figura 12.
Dirección del gradiente Sobel

Operador de Canny

Imagen de prueba: *poster.pgm*

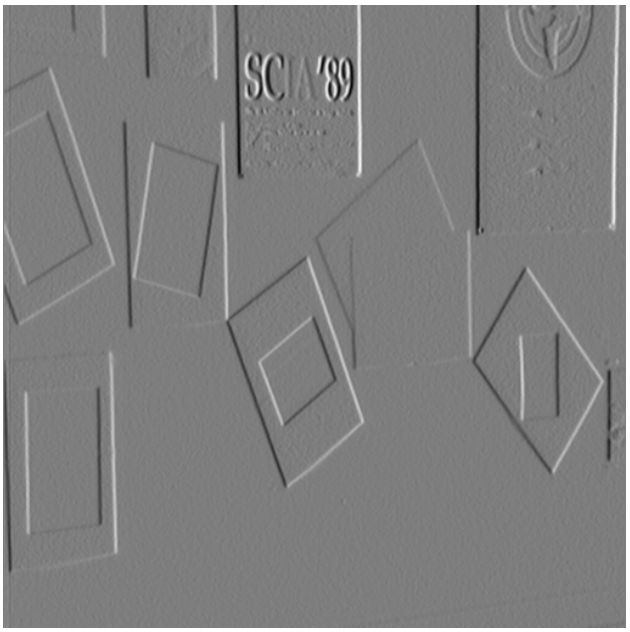


Figura 13.
Gradiente de Canny en x

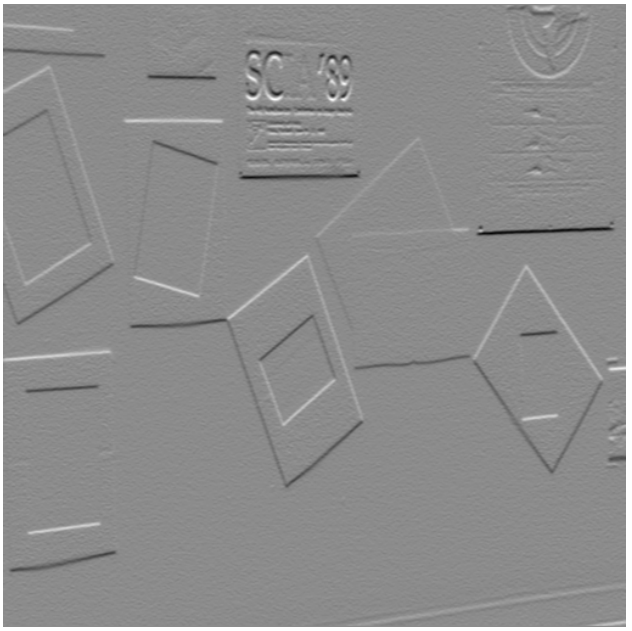


Figura 14.
Gradiente de Canny en y

Figura 15.
Módulo del gradiente Canny

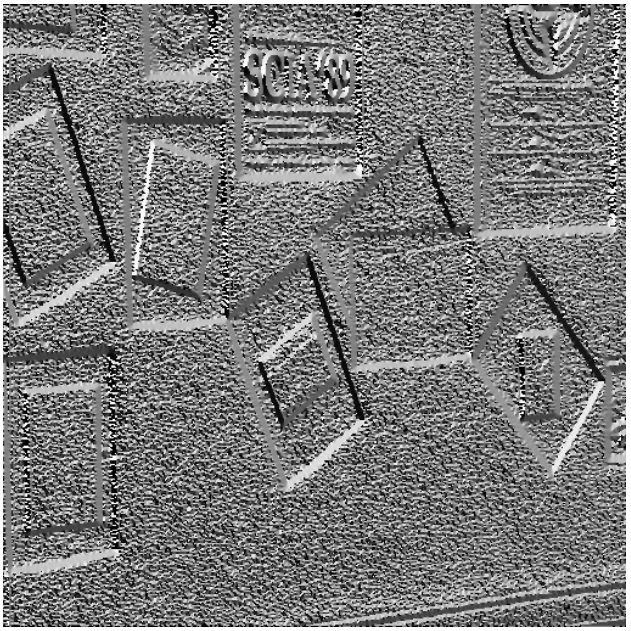


Figura 16.
Dirección del gradiente Canny

Transformada de Hough

Imagen 1

Imagen de prueba: *pasillo2.pgm*

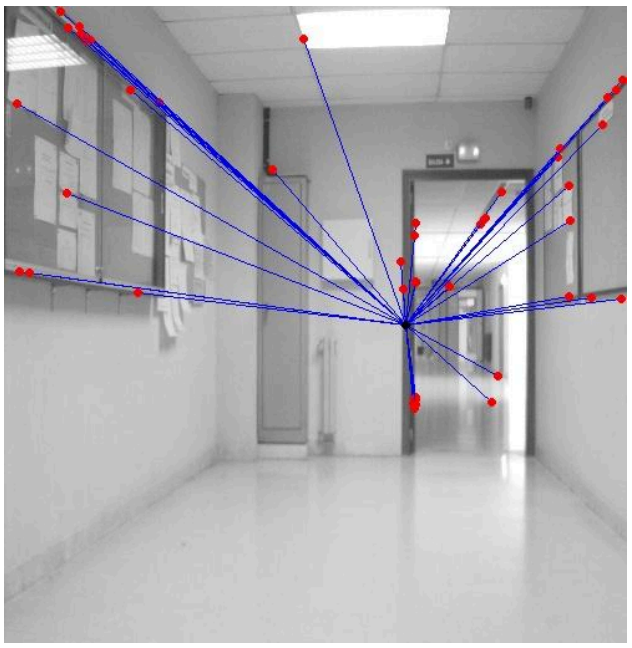


Figura 17.
Detección de punto de fuga con threshold = 100

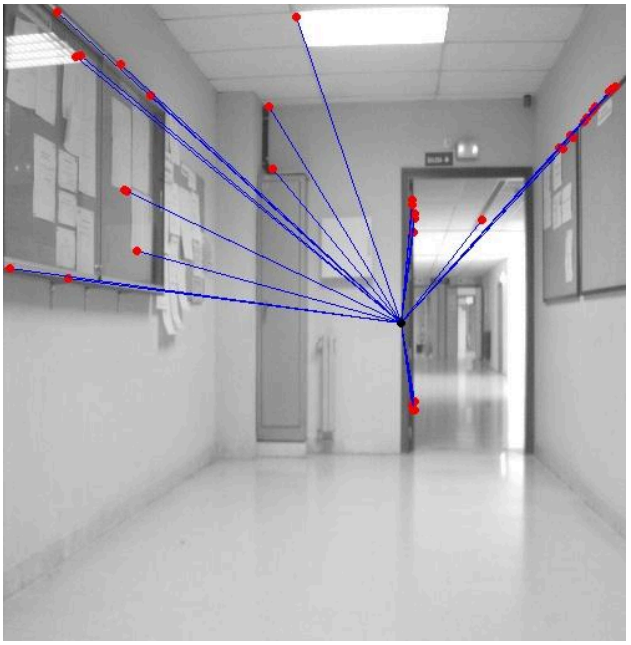


Figura 18.
Detección de punto de fuga con threshold = 150

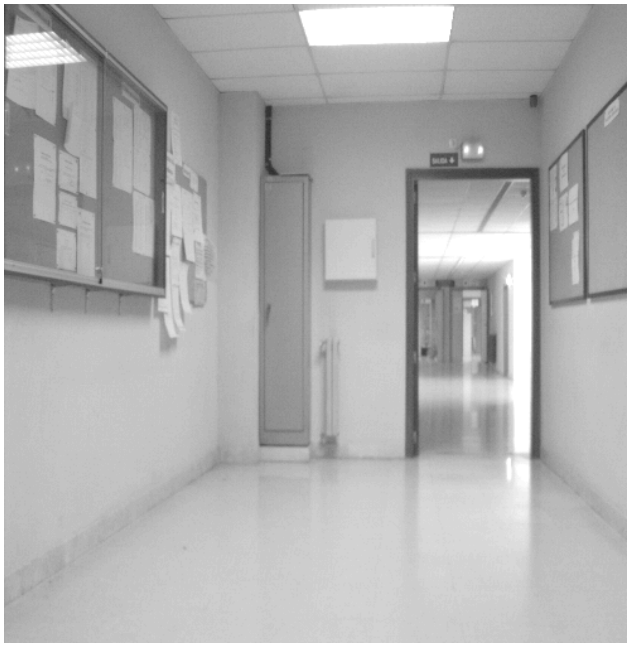


Figura 19. *Imagen de prueba original*

Imagen 2

Imagen de prueba: *pasillo3.pgm*

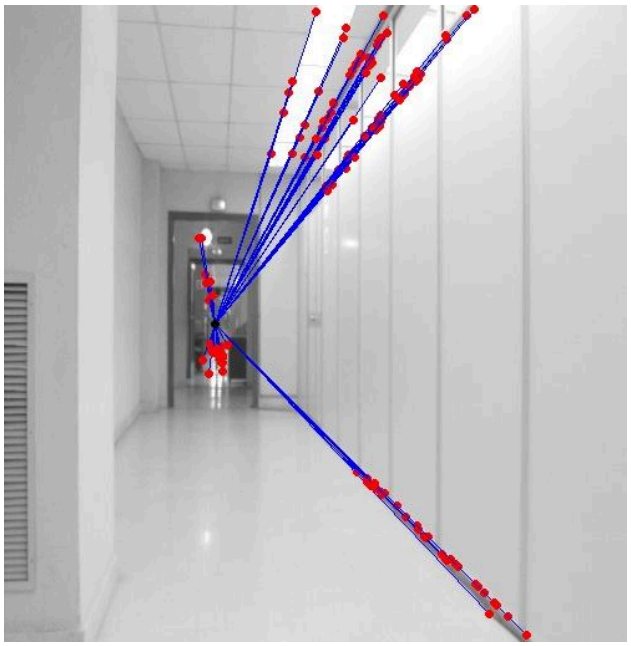


Figura 20.
Detección de punto de fuga con threshold = 100

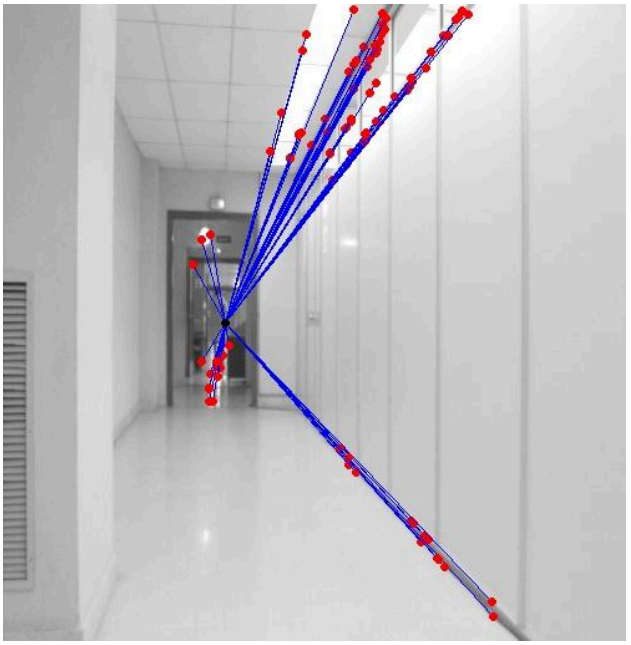


Figura 21.
Detección de punto de fuga con threshold = 150

Figura 22. *Imagen de prueba original*

Imagen 3

Imagen de prueba: *iluminacion.jpg*

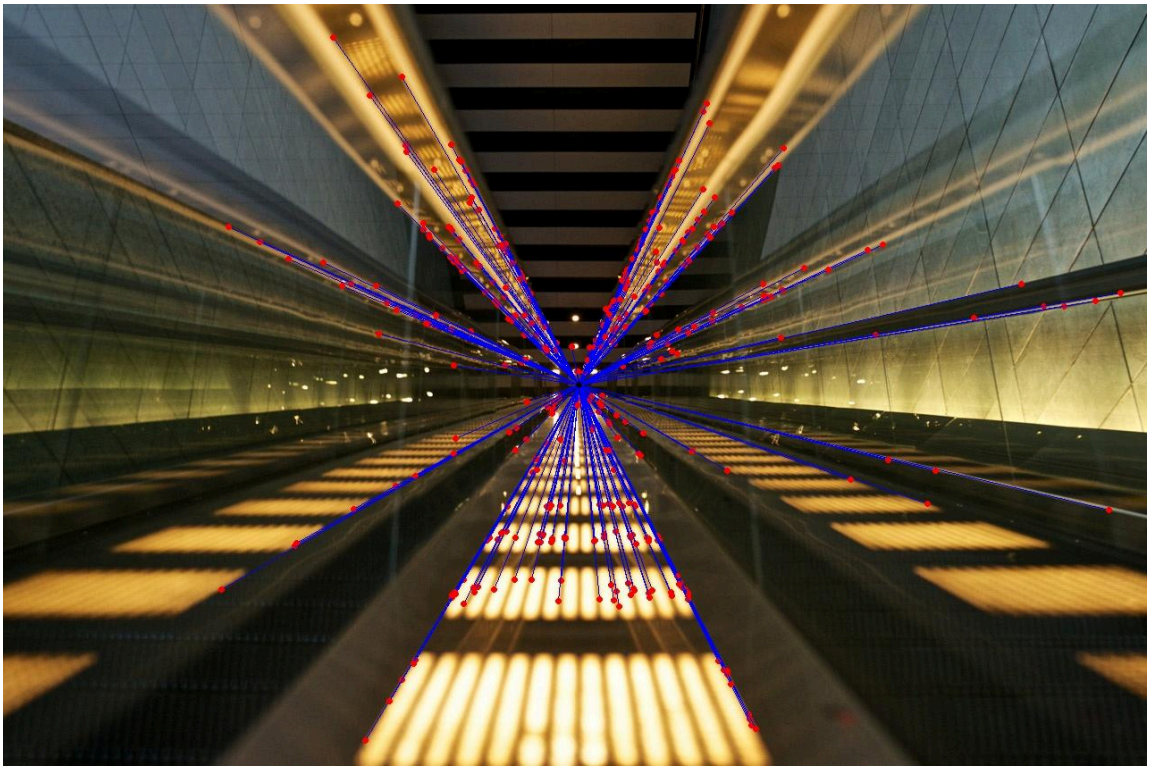


Figura 23. *Detección de punto de fuga*

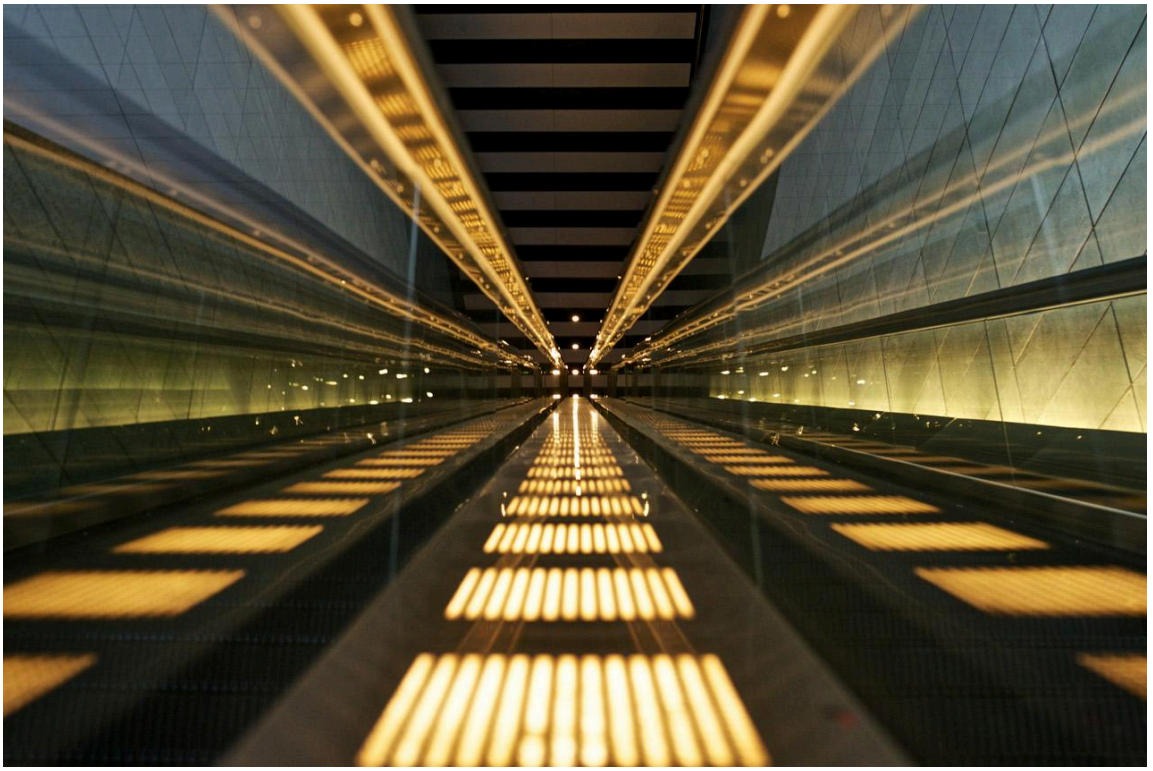


Figura 24. *Imagen original*

Opcionales

Contiene incluído un video demostrativo de la aplicación implementada. Se adjunta con este documento en la entrega.