López Pérez Leonardo

CROFI

Transformada de Hough

Es una técnica de detección de geometrías que pueden ser descritas mediante un modelo matemático canónico, por ejemplo, las líneas rectas cuya ecuación es:

$$y = mx + b$$

Incluso si esta geometría tuviera secciones en donde es interrumpida o distorsionada, eso quiere decir que utiliza “canny”, cuya función principal es la detección de geometrías mediante variaciones de brillo y contraste, arrojando como resultado una imagen que vuelve blancos los valores de bordes y negro todo lo demás. A partir de dicha imagen, es posible obtener la información necesaria para detectar líneas rectas de entre todos los demás bordes.

En este caso, sabemos que una línea puede ser parametrizada en función de un ángulo θ, de la forma: $x = \rho\cos(\theta)\,; y = \rho\sin(\theta)$ y despejando:

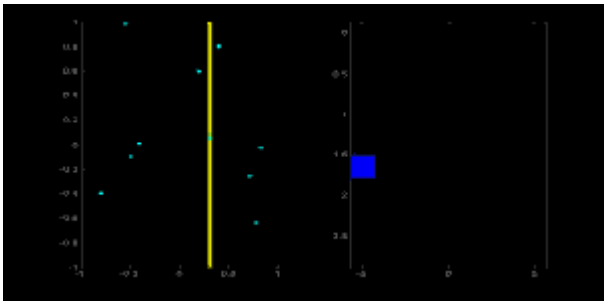
$$\frac{x}{\cos(\theta)} = \rho; \frac{y}{\sin(\theta)} = \rho$$

De tal forma que es posible representar a cada recta en función de los parámetros del ángulos y longitudes de la recta, y podemos coleccionar estos pares de datos en una matriz de dos dimensiones, de esta manera es posible determinar un nivel de precisión óptimo para la detección de líneas en cada fotograma capturado en vídeo, permitiendo una relación tal que el tamaño de la matriz dependerá de la precisión que necesite. Suponiendo una precisión de 1º grado, entonces necesitará 180 columnas. Para ρ la distancia máxima posible es la longitud de la diagonal de la imagen. Así, por ejemplo, si tomamos una precisión de 1 píxel, el número de filas podría ser igual a la longitud diagonal de la imagen.

Por ejemplo:

Considerando una imagen de 100x100 con una línea horizontal en el medio. Para cada para par (x, y) de la imagen el algoritmo comprueba cuáles rectas, definidas en la matriz mediante el par ordenado (ρ, θ), pasan por ese punto (x, y). Entonces, cada vez que un punto (x, y) pertenece a una recta, el contador correspondiente se incrementa en 1. Ilustrando este procedimiento, elegimos el punto (x=0, y=50). Muchas de las rectas con parámetros (ρ, θ), pasarán por ese punto y podrán sus respectivos acumuladores en 1. Sin embargo, sólo la recta de parámetros $\rho = 50$ y $\theta = 90º$ seguirá acumulando por encima de 1. Es decir, todos los puntos con

coordenadas x=0,1,2… 100 e y=150, sumarán uno al contador de la celda $\rho = 50$ y $\theta = 90º$



A continuación, vemos un ejemplo de la implementación de este algoritmo en Python utilizando opencv:

C:\Users\leolp\OneDrive\Documentos\PythonCodes\HoughTransform.py - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

Tarea01_TSM_LLP.py | HoughTransform.py

```python
import cv2
import numpy as np

img = cv2.imread('sudoku.jpg')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray,50,150,apertureSize = 3)

lines = cv2.HoughLines(edges,1,np.pi/180,200)
for rho,theta in lines[0]:
    a = np.cos(theta)
    b = np.sin(theta)
    x0 = a*rho
    y0 = b*rho
    x1 = int(x0 + 1000*(-b))
    y1 = int(y0 + 1000*(a))
    x2 = int(x0 - 1000*(-b))
    y2 = int(y0 - 1000*(a))

    cv2.line(img,(x1,y1),(x2,y2),(0,0,255),2)

cv2.imwrite('houghlines1.jpg',img)
```

Al darle de entrada la siguiente imagen:

Nos arroja de resultado:

Este algoritmo de por sí ya es muy útil por sí mismo, sin embardo necesita muchos cálculos, eso sumado al uso de librerías nos da como resultado un tiempo de cómputo elevado para el procesamiento de fotogramas proporcionados por la cámara de un vehículo autónomo que se desempeña en un ambiente dinámico, por lo tanto, el “boost” de éste algoritmo tiene un enfoque probabilístico que es capaz de procesar con más rapidez las líneas.