Detección de caminos
IPDI 2° C 2015

Emilio Gasco

24 de marzo de 2020

1. Introducción

La detección de caminos o detección de carril es una función muy utilizada en Sistemas avanzados de asistencia al conductor (ADAS, por su siglas en inglés) y en sistemas de visión robótica. El objetivo de este trabajo fue el desarrolló de un algoritmo que detecte correctamente el borde de caminos mientras se desplazan a través de ellos. Se trabajo con dos vídeos provistos como ejemplo, en la figura 1 se muestran un frame de cada uno. En la siguiente sección se presenta una

(a) (b)

Figura 1: Ejemplos de caminos

vista general del algoritmo y en secciones subsiguientes se profundiza sobre los puntos mas relevantes del mismo, explicando como se mejora o que problema se soluciona con cada decisión.

2. Vista general

Nuestro algoritmo recibe un vídeo como entrada y se procesan todos los frames del mismo. Para cada frame, a fin de aumentar el contraste y lograr cierta robustez a variaciones de iluminación, se ecualiza el histograma del mismo. Luego se aplica detector de bordes canny, los umbrales utilizados son adaptativos y dependen de la media de intensidades del frame.
Como se busca detectar bordes de caminos rectos, y los mismo son rectas se utiliza la transformada de hough para extraer las lineas rectas del frame. La transformada de hough consiste en tomar cada punto de borde detectado y

sumar un voto a cada una de las rectas que pasen por ese punto. Mientas más votos tenga una recta más probable que en la imagen haya aun borde delimitado por la misma. Para representar las rectas se utilizan coordenadas polares:

$$\rho = xcos(\theta) + ysen(\theta)$$

donde ρ es la menor distancia desde el origen a la recta y θ es el angulo entre el eje x y el vector $\vec{\rho}$, ver figura 2. En nuestra implementación ρ puede tomar valores tanto negativos como positivos y el angulo θ varía de 0 a π. Y el espacio de θ se dividió por 180, $\Delta\theta = \frac{\pi}{180}$.

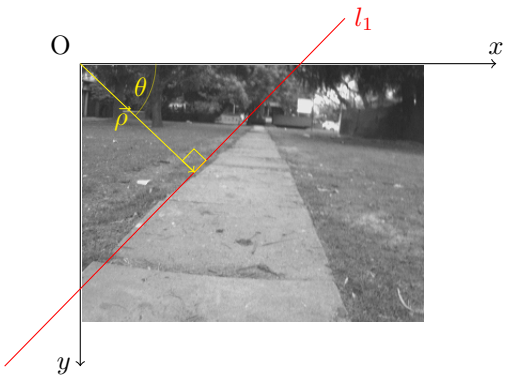


Figura 2: Transformadas de Hough, representación en coordenadas polares de las rectas

Por último, de todas las rectas obtenidas por la transformada de Hough se seleccionan las 2 que mejor ajustan al modelo de camino recto. A continuación un pseudo-código del algoritmo que resume los pasos efectuados en cada iteración.

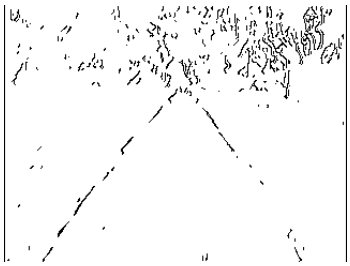
Algorithm 1 Detección de camino

```
procedure PORCESARVIDEO(video)
    while FinDeVideo(video) is false do
        frame ⇐ proximoFrame(video)
        ecualizarHistograma(frame)
        [bordes, u_min, u_max, m_gradiente, dir_gradiente] ⇐ Canny(frame)
        [lineas_izq, lineas_der] ⇐ lineasHough(bordes)
        [idx_izq, idx_der] ⇐ seleccionarBordesCamino(lineas_izq,lineas_der,dir_gradiente)
        if idx_izq is not null and idx_der is not null then
            linea_izq ⇐ lineas_izq[idx_izq]
            linea_der ⇐ lineas_der[idx_der]
            frame ⇐ drawLines(frame,linea_izq,linea_izq)
        end if
        saveFrame(frame)
    end while
end procedure
```

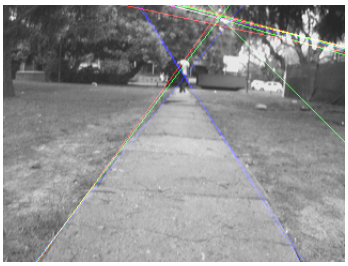
3. Detalles de Implementación

3.1. Ecualización de Histograma

La selección de un un umbral único que correctamente detecte ambos lados del camino puede ser difícil con condiciones de luz cambiantes y cuando no hay simetría con respecto al camino, es decir, con un camino que en un lado limita, por ejemplo, con césped y en el otro lado limita con otro tipo de material, generando módulos de gradientes diferentes para cada borde del camino. La selección de un umbral muy alto nos puede hacer perder uno de los bordes del camino y uno muy bajo introduce falsos bordes complicando la selección de lineas con hough.
A fin de lograr robustez a las variaciones de luz y facilitar detección en ocasiones como la descrita anteriormente se hace una ecualización de histograma antes de detectar los bordes. Obviamente la ecualización de histograma aumenta el contraste en toda la imagen, haciendo que aparezcan "falsos"bordes, por lo que es necesario hacer un suavizado de la imagen como parte de la detección de bordes. En la figura 3 se puede observar observar los bordes y lineas detectadas para un frame sin ecualizar, uno ecualizado sin suavizado y uno con ecualización y suavizado. Si se compara las lineas detectadas entre el frame sin y con suavizado, figuras 3d y 3f , no se nota mucha diferencia y esto se debe a ciertas restricciones en los ángulos de las rectas posibles que se implementa en la transformada de hough.

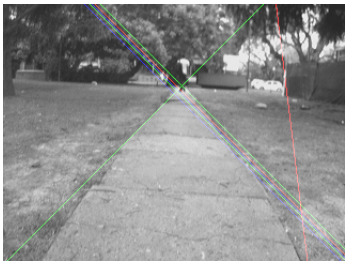


(a) Bordes frame sin ecaulizar



(b) Lineas frame sin ecaulizar

(c) Bordes frame ecualizado, sin suavizado



(d) Lineas frame ecualizado, sin suavizado

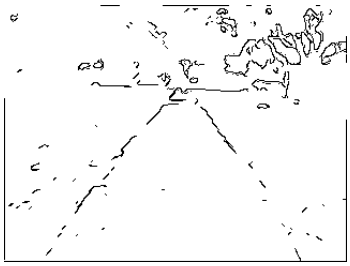
(e) Bordes frame ecualizado y suavizado

(f) Lineas frame ecualizado y suavizado

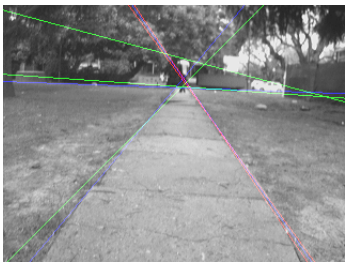
Figura 3: Efectos de ecualización de histograma

3.2. Detección de bordes

Para la detección de bordes se utiliza el algoritmo de canny modificado para utilizar solo una de las direcciones del gradiente a fin minimizar la detección de lineas horizontales que no son de nuestro interés y ademas considerarlas tiene impacto en la cantidad de cómputos necesarios al aplicar hough. En la figura 4 se muestra los diferentes resultados obtenidos al utilizar modulo de gradiente y gradiente en una dirección.



(a) Bordes detectados usando modulo gradiente



(b) Lineas detectadas a partir de bordes en figura 4a

(c) Bordes detectados usando modulo gradiente

(d) Lineas detectadas a partir de bordes en figura 4c

Figura 4: Bordes utilizando modulo e gradiente y derivada en una única dirección

La selección del umbral se realiza en función de la media del modulo de la derivada parcial en x. Se experimento con múltiples valores y los mejores resultados se obtuvieron con $u_{min} = 2\mu$ y $u_{max} = \frac{3}{2}u_{min}$

3.3. Transformada de Hough

En las primeras versiones del algoritmo se aplicaba hough y se buscaban ambos bordes del camino al mismo tiempo. El algoritmo seleccionaba todas las rectas que tuviesen mas votos que $0{,}9 \cdot max\_votos$, donde max_votos son los votos recibidos por la recta más votada. Este enfoque solo funcionaba bien en frames donde ambos bordes tenían longitudes similares. En casos como los que se muestran en la figura 5 para que el borde mas corto fuese seleccionado era necesario bajar el umbral a valores que aumentaban considerablemente el número de lineas encontradas, aumentando la cantidad de procesamiento necesario y dificultando el proceso de selección de los bordes.

(a) (b)

Figura 5: Bordes de camino de longitudes diferentes

Para resolver el problema se modificó el algoritmo para que trabajara con 2 umbrales en simultaneo y considerando solo una porción de la matriz acumuladora de votos. Primero se busca el borde izquierdo, las lineas con más votos con $\rho > 0$ y $0 \leq \theta \leq \frac{\pi}{2}$. Luego se busca el borde derecho, lineas con θ entre $\frac{\pi}{2} \leq \theta \leq \pi$. Esto mejoro la selección de lineas en la mayoría de los casos, pero el echo de tener umbral relativamente bajo hacía aparecer rectas horizontales por acumulación de votos de bordes pequeños a lo ancho de la imagen o imperfecciones en el camino.
Para lograr una detección de bordes mas robusta se decidió dividir la imagen en lado izquierdo y derecho, generando una matriz de votos independiente para cada sección,esto permitió reducir el impacto de pequeños bordes considerablemente. Se mantienen las restricciones de ρ y θ introducidas en la versión anterior.
Los valores obtenidos al aplicar hough al lado derecho de la imagen están desplazados y es necesarios convertir las coordenadas al sistema original, ver figuras 6 y 7 .Las ecuaciones para las rectas con ρ positivo y negativos, l_1 y l_2 respectivamente, están dadas por:

$$\rho'_1 = x' cos(\theta_1) + y' sen(\theta_1)$$
$$\rho'_2 = x' cos(\theta_2) + y' sen(\theta_2)$$

La intersección de l_1 y l_2 con el eje x en x_1 y x_2 se puede calcular de las siguiente forma:

$$x_1 = \frac{\rho'}{cos(\theta'_1)}$$
$$x_2 = \frac{\rho'}{cos(\theta'_2)}$$

Siendo N_2 las distancia entre el eje O e O', se puede calcular los ρ del sistema de coordenadas original con la siguiente ecuación:

$$\rho_1 = x_1 cos(\pi - \theta'_1) = \left(N_2 - \left| \frac{\rho'_1}{cos(\pi - \theta'_1)} \right| \right) cos(\pi - \theta'_1) = N_2 cos(\pi - \theta'_1) - \rho'_1$$
$$\rho_2 = (x_2 + N_2) cos(\theta 2') = \left(\left| \frac{\rho'_2}{cos(\pi - \theta'_2)} \right| + N_2 \right) cos(\pi - \theta'_2) = N_2 cos(\pi - \theta'_2) + \rho'_2$$

En el tercer termino se puede sacar el modulo dado que θ_1 y θ_2 son mayores que $\frac{\pi}{2}$ por lo que el angulo evaluado en por el coseno siempre va a estar entre 0

y $\frac{\pi}{2}$, generando un resultado positivo. Luego se puede calcular ρ en el sistema original para cualquier signo de ρ' con la siguiente ecuación:

$$\rho = N_2 cos(\pi - \theta') - \rho'$$

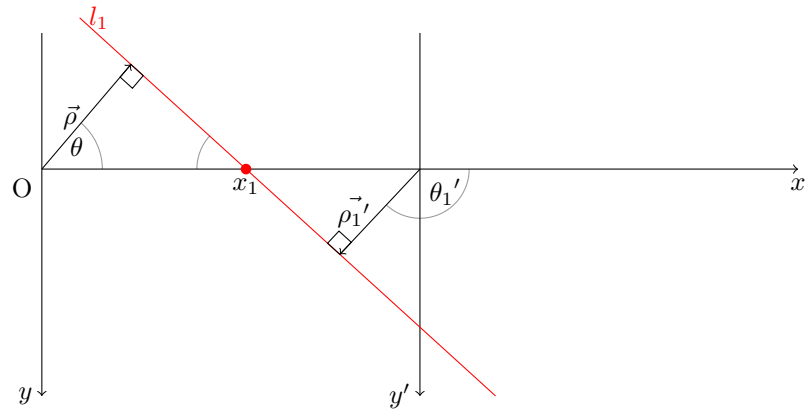


Figura 6: Transformación de coordenadas con $\rho > 0$

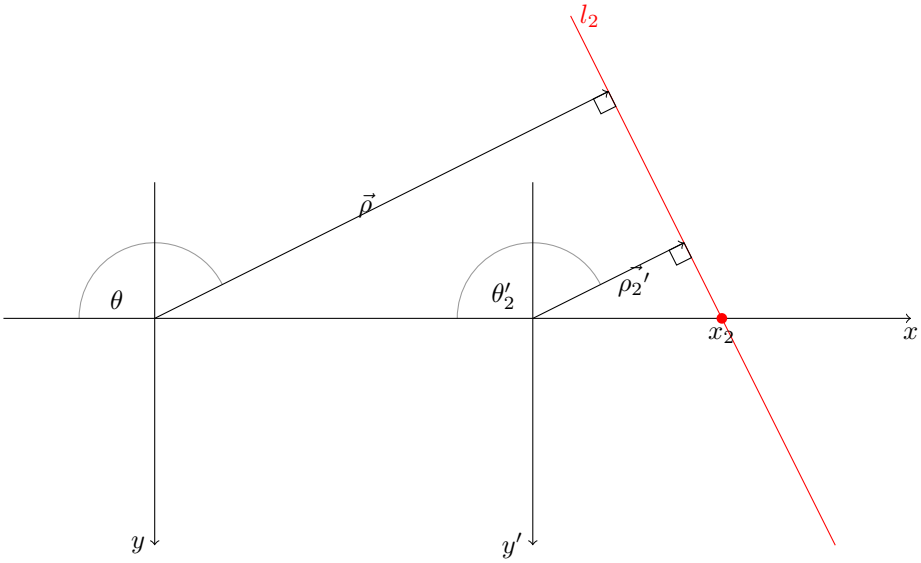


Figura 7: Transformación de coordenadas para rectas con $\rho < 0$

3.4. Selección de lineas

La detección de lineas mediante la transformada de hough puede devolver mas de 1 recta por borde de camino. A fin de seleccionar las lineas que mejor ajustan al modelo de camino recto, se utiliza la propiedad de que ambos bordes del camino van a tener dirección de gradiente opuestas. Para cada par de bordes izquierdos y derechos se calcula la intersección de las lineas y se suman las

direcciones del gradiente punto a punto, seleccionando las rectas que minimizan la suma. Se consideran los puntos desde la intersección(punto de fuga) hasta el mayor y de la recta mas corta.

Algorithm 2 Selector de bordes de camino

```
procedure seleccionarBordesCamino(lineas_izq,lineas_der,dir_gradiente)
    suma_min ⇐ ínf
    linea_izq ⇐ Null
    linea_der ⇐ Null
    for all lz in lineas_izq do
        for all ld in lineas_der do
            [x, y] ⇐ calcularInterseccion(lz,ld)
            dir_izq ⇐ obtenerAngulosGradienteEnLinea(lz)
            dir_der ⇐ obtenerAngulosGradienteEnLinea(ld)
            max_y ⇐ max(largo(dir_izq), largo(dir_der))
            tsum = 0
            for i from max(1,y) until max_y do
                tsum + =⇐ |dir_izq(i) + dir_der(i)|
            end for
            if tsum < min_sum then
                linea_izq ⇐ lz
                linea_der ⇐ ld
            end if
        end for
    end for
    return [linea_izq, linea_der]
end procedure
```

Este procedimiento demostró ser útil solo cuando las rectas devueltas por hough están çercanas.[a] los bordes del camino. Es decir la transformada hough devuelve una recta horizontal o con θ cercano a $\frac{pi}{2}$, el echo de que menos puntos participen de la suma hace que el selector tienda a seleccionar dichas rectas.

4. Conclusión

El algoritmo desarrollado logro detectar satisfactoriamente el borde en los 2 caminos con los que se probo, aunque el buen funcionamiento del algoritmo depende fuertemente de la extracción de lineas efectuada con la transformada de hough.Un mal resultado de la transformada de hough,por lo general, no puede ser recuperado por selector de bordes.

4.1. Bibliografia

Referencias

[1] Anil K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall.

[2] Rafael C.Gonzalez. Richard E.Woods, *Digital Image Processing*, Addison-Wesley. 3ra. Edición.

[3] Hunjae Yoo, Ukil Yang, and Kwanghoon Sohn, *Gradient-Enhancing Conversion for Illumination-Robust Lane Detection*, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 14, NO. 3, SEPTEMBER 2013

[4] Abdulhakam.AM.Assidiq, Othman O. Khalifa, Md. Rafiqul Islam, Sheroz Khan, *Real Time Lane Detection for Autonomous Vehicles*, Proceedings of the International Conference on Computer and Communication Engineering 2008

[5] *http://matlabtricks.com/post-41/understanding-the-hough-transform-2*

[6] *http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm*