Segmentando las líneas de un carril en carretera

Erick Rodríguez Hernández.
Maestría en Tecnología de Cómputo,
Centro de Innovación y Desarrollo Tecnológico en Cómputo
erickcarman822@hotmail.com

***Resumen*- El presente trabajo muestra diferentes técnicas de procesamiento de imágenes para poder segmentar las líneas presentes dentro de una carretera que sirven para delimitar los carriles por los cuales debe de circular un automóvil.**

***Palabras Clave*- Canny, Hough, HLS, Gaussian Filter.**

I. Introducción

La identificación de carriles en la carretera es una tarea común realizada por todos los conductores humanos para asegurar que sus vehículos se encuentren dentro de las restricciones de carril durante la conducción, así como para asegurarse de las condiciones de tránsito para así minimizar las posibilidades de colisiones con otros coches debido a que no se esté viajando dentro del carril.

Del mismo modo, es una tarea crítica para un vehículo autónomo de realizar. Resulta que el reconocimiento de las marcas de carril en las carreteras es posible utilizando algunas técnicas conocidas de visión por computadora. Dichas técnicas se tratarán a continuación

II. Desarrollo

Para realizar el procesamiento de imágenes se realizaron los programas en Python usando la librería OpenCV [1], las técnicas que se utilizaron se describen a continuación:

- La imagen original utilizada para realizar las pruebas para el procesamiento mostrada en la figura (1) se convierte al espacio de color HLS.
- Se convierte la imagen a escala de grises.
- Se crea una máscara de pixeles amarillos y blancos.
- Aplicamos un filtro Gausiano para reducir el ruido en la imagen.
- Realizamos la detección de bordes con Canny.
- Creamos una máscara que nos muestre solo la región de interés en la imagen.
- Dibujamos líneas sobre la imagen que son encontradas por el método de Hugh.

Fig. 1 Imagen de prueba para aplicar las técnicas de procesamiento.

La entrada en cada paso es la línea de salida del paso anterior, por ejemplo, aplicamos la detección de los bordes con Canny a la imagen que previamente se le aplicó un filtrado Gaussiano.

A. *Conversión a diferentes espacios de color*

Teniendo la imagen original se procedió a explorar la visualización de la imagen en otro espacio de color, como lo es el Hue, Saturation, Lightness (Matiz, Saturación, Luminosidad) o HSL por sus siglas en inglés, que nos permite realizar una mejor segmentación de las líneas de color blanco y amarillo. La figura (2) nos muestra la visualización de la imagen una vez que convirtió de RGB a HSL.

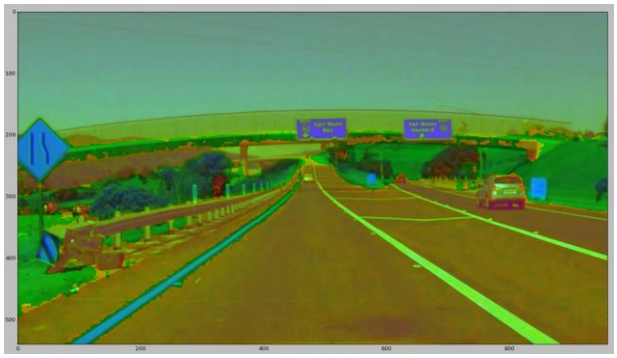

Fig 2. Imagen procesada en el espacio de color HSL.

B. *Escala de grises*

Estamos interesados en poder detectar las líneas blancas o amarillas de una imagen de una carretera, estas van a mostrar un contraste particularmente grande cuando la imagen se encuentra en escala de grises, lo cual, el convertir a escala de grises nos ayudará para poder realizar un tratamiento más sencillo de la imagen.

C. *Máscara de pixeles amarillos y blancos*

Para que pudiéramos tener un segmentado de las líneas más eficiente, se procedió a realizar un enmascaramiento de la imagen procesada en HSL, tomando en cuenta los valores superiores e inferiores para poder detectar pixeles en colores amarillos y blancos. Una vez procesada la máscara, se le aplicó a la imagen en escala de grises dicho enmascaramiento como se muestra en la figura (3), ayudando aún más a reducir el ruido dentro de la imagen.

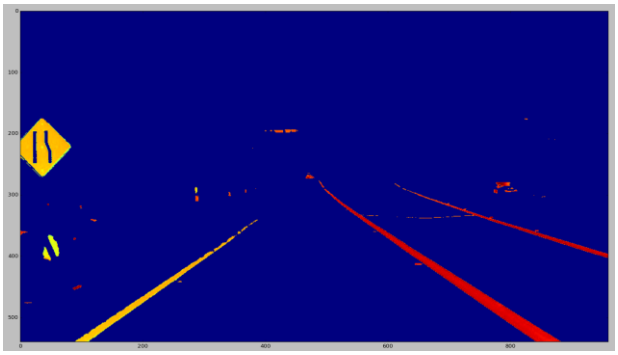

Fig. 3 Imagen con el enmascaramiento de pixeles blancos y amarillos.

D. *Filtrado Gaussiano*

El filtrado Gaussiano es una técnica de pre-procesamiento utilizado para suavizar los bordes de una imagen tratando de reducir el ruido. Dicho filtro lo aplicamos intentando reducir el número de líneas que se han detectado, teniendo en cuenta que no tenemos que aplicar un valor muy grande en el valor de su Kernel, sino de lo contrario nos será más difícil poder detectar los bordes que delimitan la carretera de la imagen.

E. *Detección de Bordes con Canny*

Ahora que hemos procesado suficiente la imagen, podemos aplicar un Detector de Bordes de Canny [2], cuya función es identificar los bordes de una imagen y descartar todos los demás datos. Podemos encontrar el gradiente de borde y dirección para cada pixel dada la siguiente ecuación:

$$\begin{aligned} Edge\_Gradient\ (G) &= \sqrt{G_x^2 + G_y^2} \\ Angle\ (\theta) &= \tan^{-1}\left(\frac{G_y}{G_x}\right) \end{aligned} \tag{1}$$

La implementación requiere de dos parámetros de umbral, uno alto y uno bajo que determinan si se incluye o no al borde en cuestión. Un umbral captura la intensidad de cambio de un punto dado. Cualquier punto más allá del umbral alto se incluirá en nuestra imagen resultante, mientras que los puntos intermedios entre los valores de umbral dados, sólo se incluirán si están próximos a nuestro umbral alto. Los bordes que están por debajo de nuestro umbral bajo se descartan tal como lo muestra la figura (4).

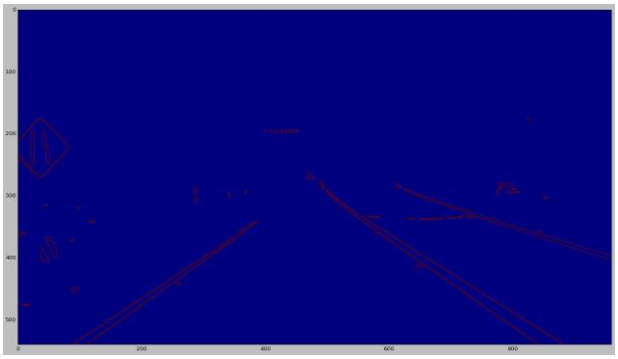

Fig. 4 Imagen donde se muestra la detección de bordes utilizando Canny.

F. *Región de interés*

Nuestro siguiente paso es determinar una región de interés y descartar cualquier línea que se encuentre fuera de este polígono.

Dadas las imágenes procesadas podemos determinar una región donde queremos que los carriles permanezcan y que lo demás sea descartado, tal como nos muestra la figura (5).

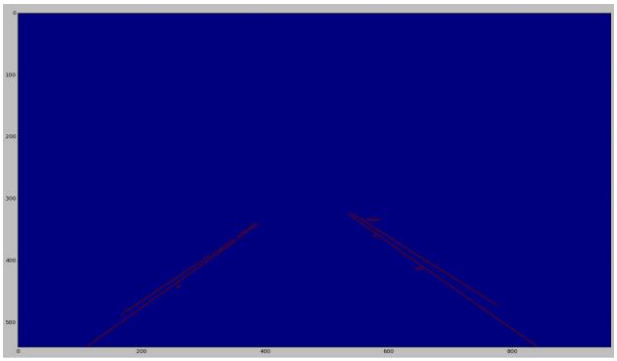

Fig. 5 Imagen dónde sólo queda nuestra región de interés.

G. *Transformada de Hough*

El siguiente paso es aplicar la técnica de la Transformada de Hough para extraer líneas y dibujarlas en la imagen. El objetivo de la Transformada de Hough es encontrar líneas identificando todos los puntos que se encuentran en ellas. Esto se hace mediante la conversión de nuestro sistema actual denotado por el eje (x, y) a uno paramétrico donde los ejes son (m, b). En este plano las líneas son representadas como puntos, los puntos se presentan como líneas y las líneas de intersección significa que el mismo punto está en varias líneas.

H. *Procesamiento final*

Una vez que se realizó la transformada de Hough para encontrar todas las líneas que componen al carril de la carretera, se continúa ahora haciendo un promediado de todas las líneas para poder así manejar solo una línea que defina el carril, esto se determina primero definiendo si las líneas corresponden al carril izquierdo o derecho verificando la pendiente que manejan. Así ahora podemos sumar y promediar las líneas izquierdas con todas las izquierdas y las derechas con todas las derechas. Dando como resultado final la figura (6)

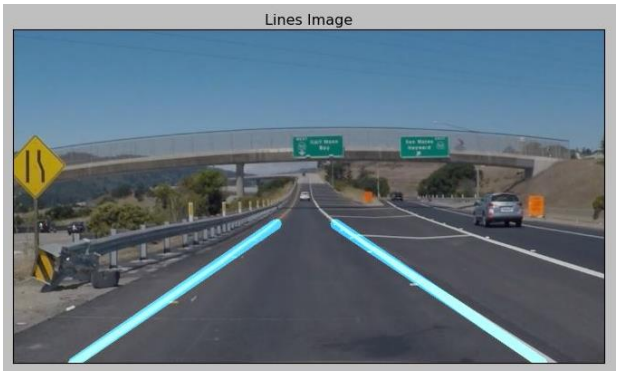

Fig. 6 Imagen final una vez aplicado las técnicas de procesamiento.

III. Conclusiones

El desarrollo de nuevas computadoras con mayores capacidades de procesamiento, ha dado lugar al desarrollo de aplicaciones para el procesamiento de imágenes en sistemas en tiempo real, hoy en día vemos que se encuentra en la cúspide de desarrollo de vehículos autónomos.

La práctica desarrollada da lugar a tratar de crear nuevas soluciones a segmentación de carriles durante una curva.

Durante el promediado de la suma de las líneas cuando se trató de procesar un video se presentaron fallas al aparecer una división entre cero; como trabajo a corto plazo se procederá a la mejora del algoritmo.

Referencias

[1]. OpenCV, http://docs.opencv.org/2.4/index.html.
[2]. Canny Edge Detection http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html