INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Departamento de Procesos Tecnológicos e Industriales

Sustentabilidad y Tecnología

PROYECTO DE APLICACIÓN PROFESIONAL (PAP)

Programa de apoyo a centros de investigación externos

ITESO, Universidad Jesuita de Guadalajara

PAP4G03 PAP PROGRAMA DE APOYO A CENTROS DE INVESTIGACIÓN EXTERNOS I

"Detección del área navegable para un vehículo autónomo" en el Centro de Investigación y de Estudios Avanzados plantel Zapopan

PRESENTAN

Programas educativos y Estudiantes

Lic. en Ingeniería Electrónica. Alan Yahir Duran Merchant

Profesores PAP: Dr. Alexander G. Loukianov,

Dr. Luis Enrique González Jiménez

Tlaquepaque, Jalisco, mayo de 2019

Contenido

Resumen 3
Sustento metodológico 3
Descripción del proyecto 9
Resultados obtenidos 13
Trabajo a futuro 17
Referencias 18

Resumen

Se implementó un algoritmo de visión que determina el camino con una sola cámara a partir del método de Otsu [1], detección de líneas de con la Transformada de Hough [2] y manipulación de los espacios de color, presentando dificultades al encontrarse sombras en la imagen, tener bordes poco definidos o existir señales de tránsito no identificadas.

Sustento metodológico

Los sensores basados en visión artificial pueden ser utilizados para varias aplicaciones específicas, como la localización de marcas viales, identificación del área que un vehículo es capaz de navegar dentro del camino que tiene por delante, reconocimiento de señales de tránsito y detección de obstáculos, entre otras [3]. Entre ellos, los sistemas de cámaras monoculares tienen preferencia respecto a los sistemas de cámaras estéreo debido a las ventajas que presentan en términos de costo y la facilidad con la que pueden ser instalados en un vehículo [4]. Sin embargo, los sensores de visión son menos robustos que los radares de onda milimétrica en condiciones de neblina, durante la noche o cuando reciben luz directa del sol, por lo que para detectar el espacio disponible que se encuentra alrededor de un robot móvil es necesario implementar algoritmos que reduzcan el ruido en las imágenes y segmentar su información relevante [5].

Una manera de realizar la segmentación de una imagen es por medio de umbrales. Este tipo de segmentación es computacionalmente rápido, sin embargo, encontrar un umbral ideal puede resultar complicado, por lo que se implementan algoritmos que busquen dicho umbral de manera automática, como lo es el método de Otsu [1]. Este método determina un umbral global para la segmentación de la imagen en dos clases: el primer plano, el cual contiene el objeto de interés, que en este caso sería el área navegable del camino, y el segundo plano o fondo de la imagen, que contiene todo lo que está detrás del objeto de interés. Un ejemplo de los resultados de segmentar una imagen con este método puede verse en la figura 1.

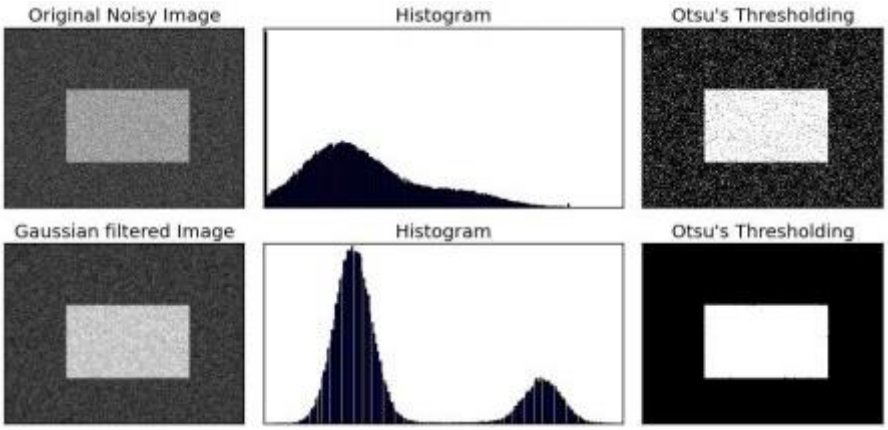

Figura 1. Resultados de la segmentación con Otsu y los efectos de filtrar la imagen. Obtenido de [6]

El método de Otsu se obtiene a partir del histograma de una imagen, el cual mide la frecuencia con la que se repite cada valor de intensidad dentro de una imagen en escala de grises [7], buscando obtener la mínima varianza intraclase posible, es decir, tener el mayor conjunto posible de pixeles pertenecientes a una misma clase juntos y a su vez tener la mayor dispersión posible entre los pixeles pertenecientes a clases diferentes, y por lo tanto tener máxima varianza interclase posible. La varianza intraclase y la varianza interclase son inversamente proporcionales, por lo que en el algoritmo solo nos interesaremos en calcular y minimizar la varianza intraclase, la cual está definida en la ecuación

$$\sigma_w^2(T) = q_1(T)\sigma_1^2(T) + q_2(T)\sigma_2^2(T)$$

donde:

$\sigma_w^2(T)$ es la varianza intraclase

$q_1(T)$ es la proporción de pixeles en segundo plano (background)

$\sigma_1^2(T)$ es la varianza en la intensidad en los pixeles en segundo plano

$q_2(T)$ es la proporción de pixeles en primer plano (foreground)

$\sigma_2^2(T)$ es la varianza en la intensidad en los pixeles en primer plano

T es el umbral.

Las proporciones de cada clase se definen en la ecuación (1). Donde H es el histograma de la imagen y N es el máximo número de intensidades que presenta.

$$q_1(T) = \sum_{i=1}^{T} H(i) \quad y \quad q_2(T) = \sum_{i=T+1}^{N} H(i) \qquad (1)$$

Para calcular la varianza, es necesario conocer la media ponderada de cada clase μ, la cual se obtiene según la ecuación (2).

$$\mu_1(T) = \sum_{i=1}^{T} \frac{iH(i)}{q_1(T)} \quad y \quad \mu_2(T) = \sum_{i=T+1}^{N} \frac{iH(i)}{q_2(T)} \qquad (2)$$

Finalmente, la varianza de cada clase $\sigma_1^2(\mathrm{T})$ y $\sigma_2^2(\mathrm{T})$ se define en las ecuaciones (3) y (4):

$$\sigma_1^2(\mathrm{T}) = \sum_{i=1}^{T} [i - \mu_1(T)]^2 \frac{H(i)}{q_1(T)} \qquad (3)$$

$$\sigma_2^2(\mathrm{T}) = \sum_{i=T+1}^{N} [i - \mu_2(T)]^2 \frac{H(i)}{q_2(T)} . \qquad (4)$$

Con el objetivo de detectar los límites del camino, lo cual incluye obstáculos, se utiliza el detector de bordes de Canny como entrada de la Transformada de Hough debido a su robusto desempeño [5]. El algoritmo de Canny calcula el gradiente de la imagen, descarta todos los pixeles que no estén en la dirección del gradiente y no sean un máximo local y después selecciona los pixeles que se encuentran dentro del borde mediante un umbral de histéresis con un valor máximo y un mínimo definido por el usuario [8]. El proceso para encontrar un máximo local se encuentra detallado en la figura 2, donde dependiendo de la dirección de la gradiente indicada en amarillo, el borde podría encontrarse en los pixeles de color rojo, por lo que el píxel central en cuestión se compara con los que estén en esa dirección, señalados de color negro.

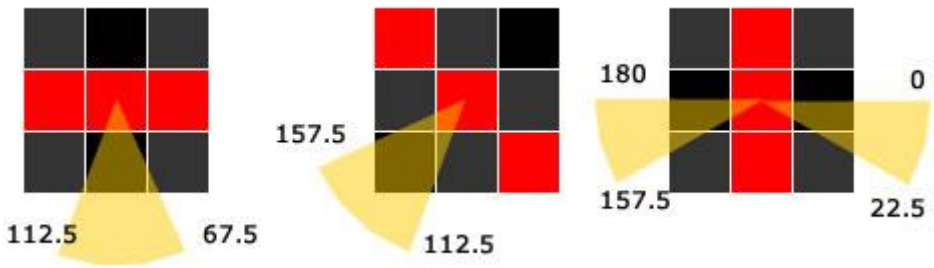

Figura 2. Posibles bordes dependiendo de la dirección del gradiente. Obtenido de [9].

Una vez realizado este procedimiento se obtienen varios puntos que potencialmente podrían ser parte de los bordes de la imagen, para determinar cuáles realmente lo son y unir a los que estén adyacentes, se utiliza el umbral de histéresis, donde si un punto sobrepasa el umbral alto, este se extiende hasta que se encuentre con el umbral bajo, como se muestra en la figura 3.

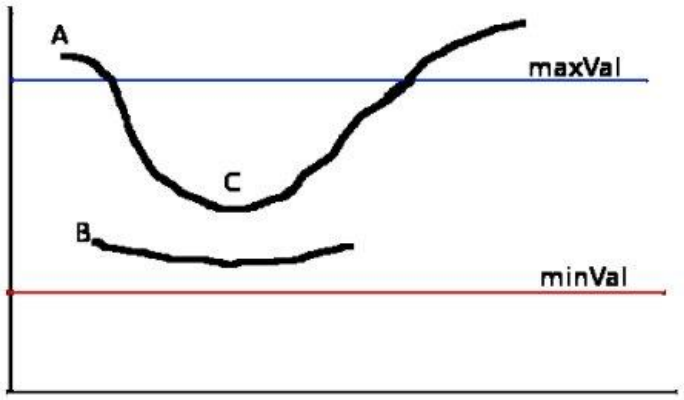

Figura 3. Borde detectado con umbral de histéresis. Obtenido de [8].

La Transformada de Hough es utilizada entonces para detectar líneas en los bordes mediante su representación en coordenadas polares y midiendo las intersecciones que se tienen con respecto a un punto [2]. Este procedimiento se pude apreciar en la figura 4. Este procedimiento es costoso computacionalmente, y para reducir el procesamiento necesario es común utilizar la PHT (*Probabilistic Hough Transform*) la cual es una estimación de la Transformada de Hough [10].

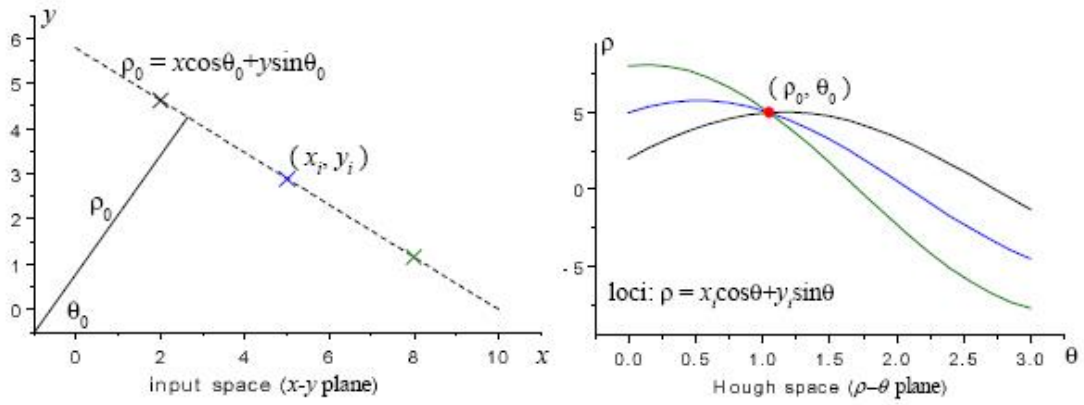

Figura 4. Representación de una línea en coordenadas cartesianas y en el espacio de Hough. Obtenido de [11].

En una imagen en escala de grises, el contraste se define como la diferencia de intensidad entre dos objetos distintos [12]. Para mejorar o hacer más notoria esta característica, una técnica comúnmente utilizada es la ecualización de su histograma, en la cual se distribuyen los valores de intensidad concentrados en una sola región del histograma a lo largo de todo su espectro. Los efectos de esta técnica se pueden observar en la figura 5.

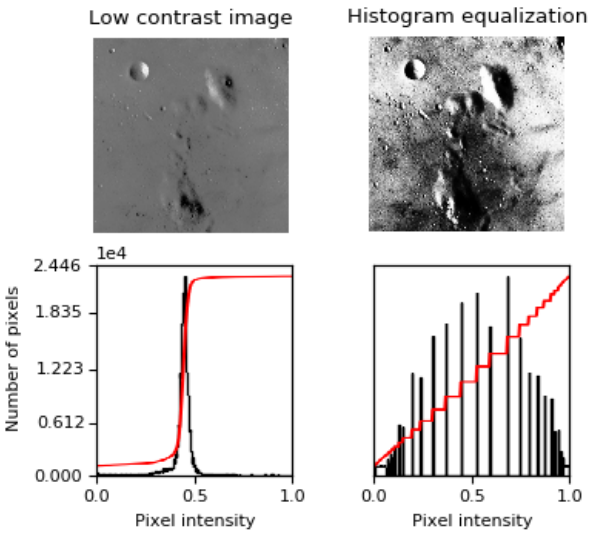

Figura 5. Ecualización de un histograma y sus respectivas imágenes. Obtenido de [12].

Un modelo de color es un modelo matemático abstracto que permite representar los colores de manera numérica. Un espacio de color es una forma específica de organizar los colores [13]. Un modelo de color puede tener distintas representaciones en espacios de color.

RGB (Red, Green, Blue) es un modelo de color aditivo donde los colores primarios pueden sumarse y formar una amplia variedad de colores. Existe un espacio de color con el mismo nombre utilizado para representar cada uno de estos colores en un canal distinto, el cual es útil para la generación de colores de manera sencilla, pero en procesamiento de imágenes presenta problemas a la hora de analizar su información debido a que los canales son dependientes de la iluminación, complicando la identificación de patrones de interés [13].

Una alternativa es el espacio de color HSV (del inglés *Hue*, *Saturation*, *Value* – Matiz, Saturación, Valor) que se muestra en la figura 6, el cual está basado en el mismo modelo

de color RGB. El canal H describe el color de la imagen en términos de su longitud de onda, el canal S diferencia al color contra un gris de la misma intensidad y el canal V define la iluminación de la imagen. De esta manera, al tener la información separada en canales distintos, es posible tener un mejor análisis de la imagen.

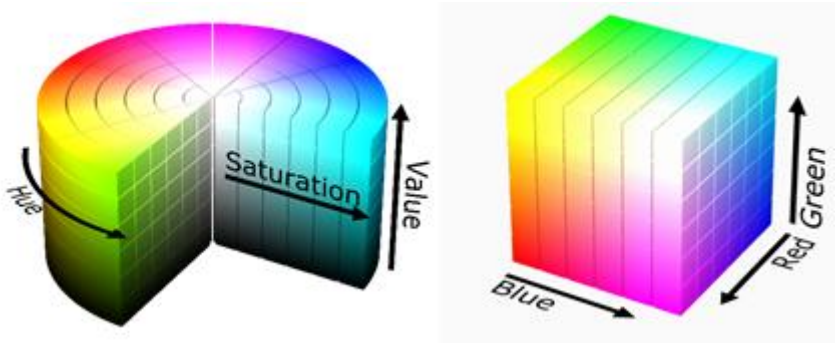

Figura 6. Comparación de los espacios de color. Obtenido de [13].

Descripción del proyecto

A continuación, se describe la metodología utilizada para la detección del área navegable por un vehículo a partir de una imagen obtenida por una cámara monocular.

La figura 7 describe de manera general el algoritmo que se ha realizado hasta el momento, donde se determina el área navegable considerando tres distintas fuentes: el camino detectado mediante Otsu, el camino detectado utilizando las líneas de Hough y la escala de grises de la imagen original con el histograma ecualizado. Todos los algoritmos fueron realizados haciendo uso de la librería OpenCV [8] especializada para visión por computacional.

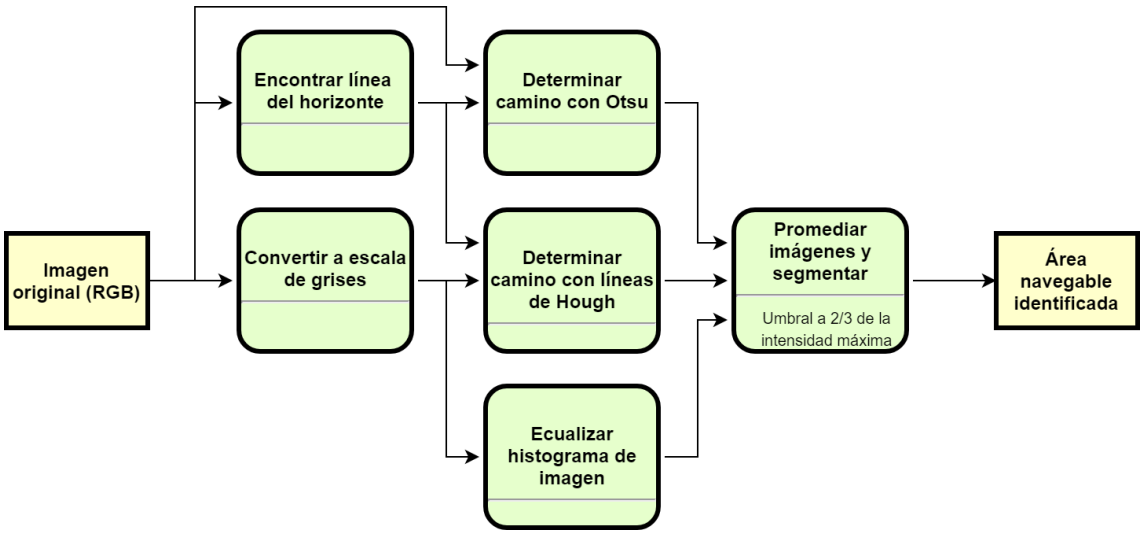

Figura 7. Esquema general del algoritmo implementado. Realizado en draw.io

Debido a que el método de Otsu busca un solo umbral para realizar una segmentación global binaria, es susceptible a errores si no existe una diferencia notoria entre el objeto de interés y el fondo de la imagen, por lo que resulta conveniente analizar una porción de la imagen más cercana al objeto de interés en vez de la imagen completa. Para propósitos de detección del camino en una imagen, la zona seleccionada es aquella más cercana al vehículo, en la parte inferior de la imagen.

Con el propósito de simplificar el procesamiento y analizar solamente la porción de la imagen correspondiente al camino, se implementa un algoritmo para determinar la

ubicación del horizonte basado principalmente en el método de Otsu [14]. Todo lo que esté por encima de este horizonte es considerado como parte del cielo y por lo tanto descartado, mostrándose los resultados de dicho procedimiento en la figura 8.

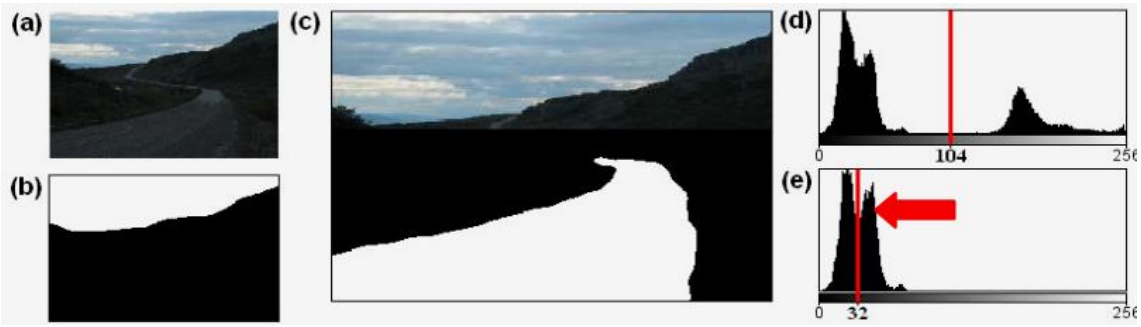

Figura 8. Resultado de Otsu con la imagen original a) y b); Resultado de Otsu considerando solo la parte del camino c); Histogramas de imagen antes y después de descartar el horizonte en d) y e). Obtenido de [5].

Una de las principales desventajas del método de Otsu es su sensibilidad al ruido de la imagen, por lo que antes de implementar este algoritmo se aplica una etapa de preprocesamiento, en la que suavizamos la imagen con un filtro gaussiano, que actúa como un filtro pasa bajas. Además, se cambia el espacio de color de la imagen de RGB a HSV y se selecciona solamente el canal H, el cual contiene la información que corresponde al color de la imagen, para su análisis.

Se inicia con la suposición de que el horizonte se encuentra en la parte superior de la imagen, por lo que se toma el 60% superior de la imagen y se divide verticalmente en 10 filas, para obtener los umbrales con el método de Otsu para cada una de ellas. Posteriormente se realiza la segmentación con todos los 10 umbrales en cada una de las sub-imágenes, y se mide la proporción de pixeles pertenecientes al cielo obtenidos en cada una de ellas. Por último, se analiza la diferencia porcentual la proporción de pixeles de cielo entre una fila y la anterior, y donde exista la mayor diferencia se encuentra nuestra línea del horizonte. Todo el proceso para obtener la línea del horizonte se muestra en la figura 9.

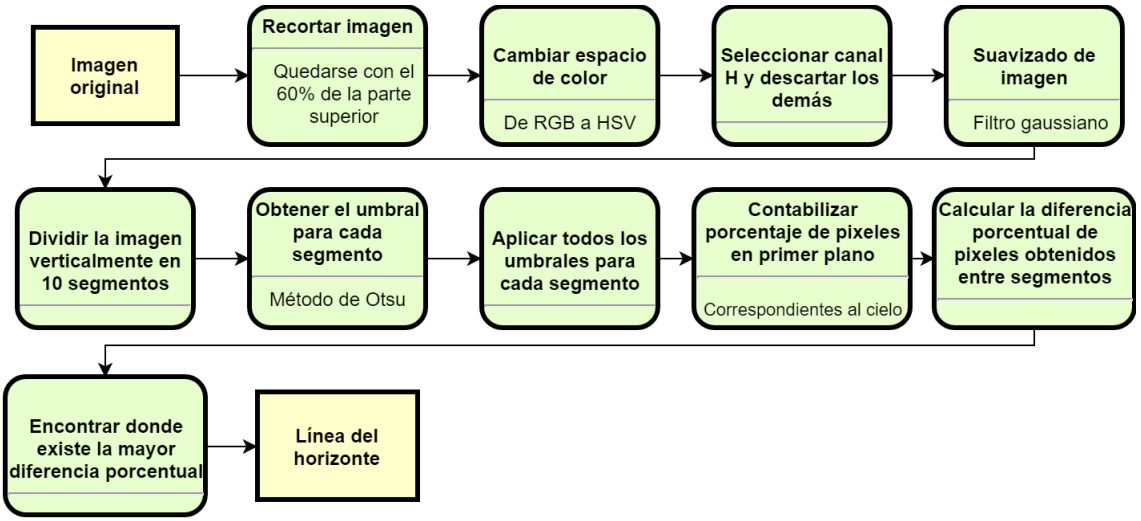

Figura 9. Diagrama a bloques del procedimiento para determinar la línea del horizonte. Realizado en draw.io

Una vez detectada la línea del horizonte, la parte superior a ella se descarta. Durante las pruebas realizadas se hizo evidente que el método de Otsu se veía afectado por las sombras de la imagen, para mitigar su efecto se decidió homogeneizar el canal V del espacio de color HSV, debido a que este contiene la información relevante a la iluminación de la imagen. Al tener el mismo valor de iluminación en todos los pixeles, el efecto que tenía la sombra en el resultado se volvió mucho menos notorio. Finalmente, se valida el porcentaje de área navegable en la parte inferior central de la imagen, ya que dependiendo su textura existe la posibilidad de obtener un mejor resultado negándola. El algoritmo completo para detectar el camino mediante Otsu se encuentra descrito por el diagrama en la figura 10.

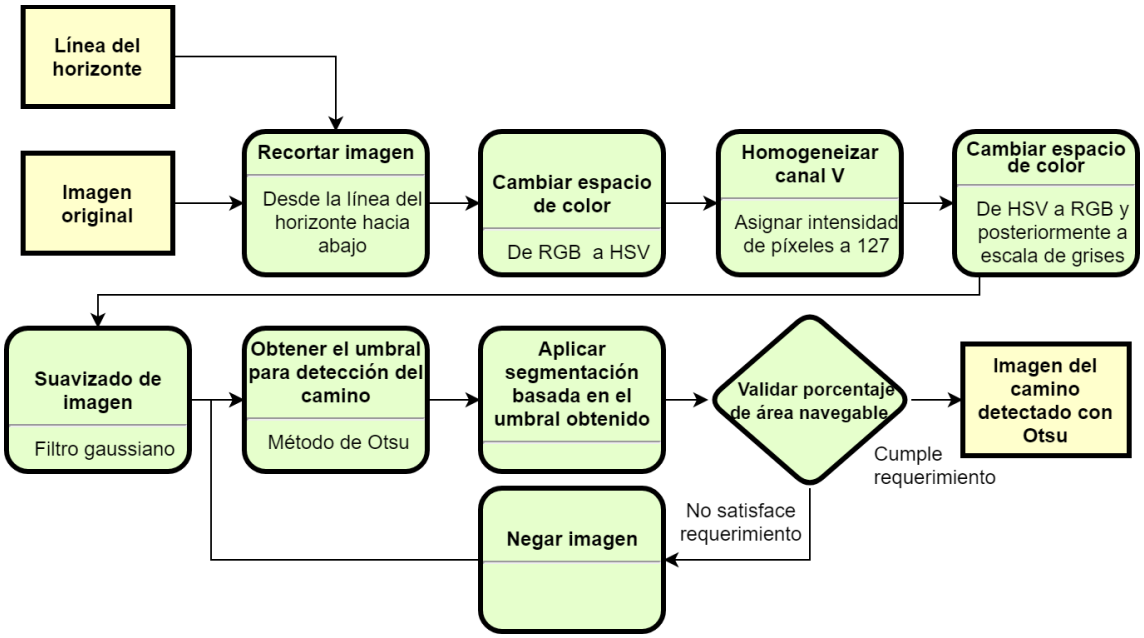

Figura 10. Diagrama a bloques de la detección del camino con Otsu. Realizado en draw.io

Esta metodología para encontrar la superficie navegable para el vehículo, como se menciona en [5], supone que la cámara está centrada en medio del camino y que este presenta cierta uniformidad, lo cual no sucede en los caminos de piedra que tiene el ITESO, por ejemplo. Por lo tanto, para identificar los límites del camino y detectar obstáculos, se toman en cuenta los bordes de la imagen, como suelen ser las banquetas, mediante la implementación de la Transformada de Hough para detectar líneas utilizando como entrada la salida del algoritmo de Canny. Este procedimiento se puede apreciar en la figura 11.

Figura 11 - a) Imagen original y su resultado de Otsu en b); c) Imagen negada y su resultado de Otsu en d); e) Detector de bordes de Canny; f) Transformada de Hough; g) Camino detectado con Hough. Obtenido de [5]

Una vez detectadas las líneas, se intentan formar contornos que determinen si una sección de la imagen es camino o no, sin embargo, muchas veces el contorno no se llega a cerrar por alguna discontinuidad en los bordes, por lo que a partir del tamaño de las

líneas y su pendiente se trata de identificar si la línea en cuestión pertenece a una banqueta, ya que como menciona [15] estas líneas tienden a extenderse hacia el fondo de la imagen con un ángulo ligeramente inclinado hacia adentro, y en caso de ser así, se forma un contorno extendiéndola hasta los laterales de la imagen. El algoritmo implementado se describe en la figura 12.

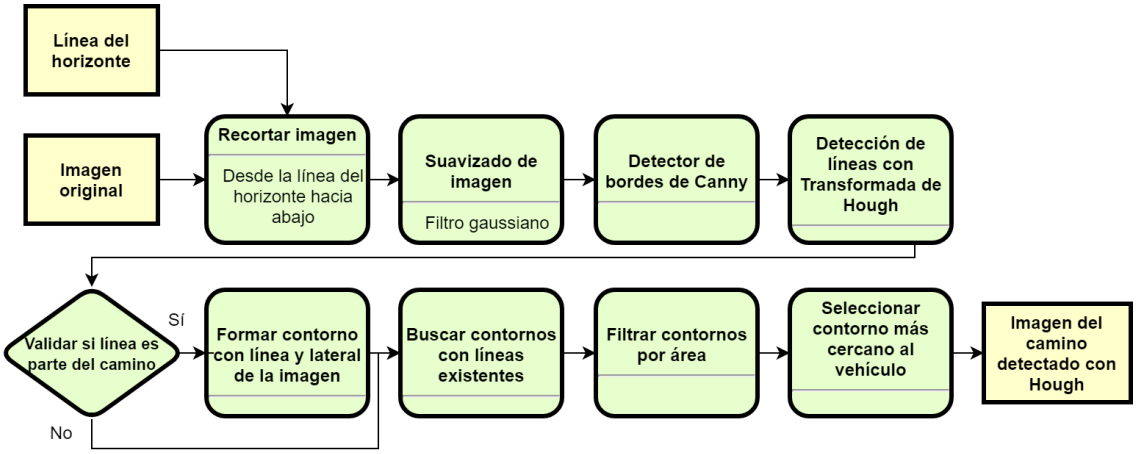

Figura 12. Diagrama a bloques de la detección del camino con Hough. Realizado en draw.io

La imagen del camino detectado se obtiene con la ecuación (5). Donde $FA_{w(x,y)}$ de la izquierda es el píxel en la posición (x, y) después de hacer un promedio con $FA_{w(x,y)}$ de la derecha que corresponde a la imagen original con su histograma ecualizado, $Os_{(x,y)}$ que corresponde a la salida del método de Otsu y $Hc_{(x,y)}$ que es la salida de la Transformada de Hough

$$FA_{w(x,y)} = \frac{FA_{w(x,y)} + Os_{(x,y)} + Hc_{(x,y)}}{3} \qquad (5)$$

Resultados obtenidos

En el caso del área navegable detectada mediante Otsu, el algoritmo resultante es robusto a perturbaciones como sombras o el gradiente de la imagen producido por las piedras en el camino, por ejemplo, mostrándose en la figura 13 la mejora obtenida al homogeneizar el canal V.

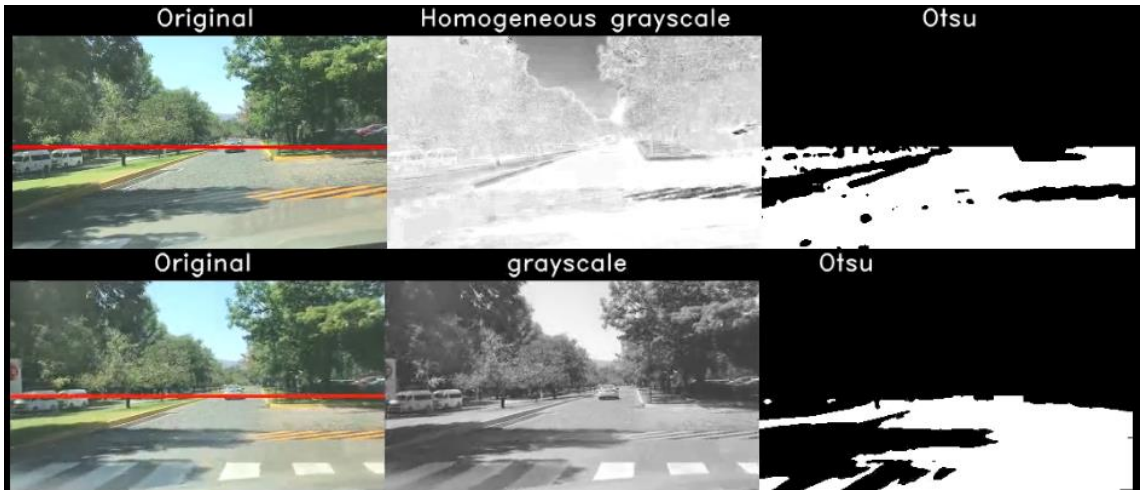

Figura 13. Comparación del resultado de Otsu con diferentes entradas. Realizado en OpenCV

Sin embargo, el hacer esta modificación en el espacio de color, hace que se pierdan ciertos detalles, como lo son los vehículos u otros obstáculos, de los cuales ahora solo se percibe una porción de ellos. Un ejemplo de esto se muestra en la figura 14.

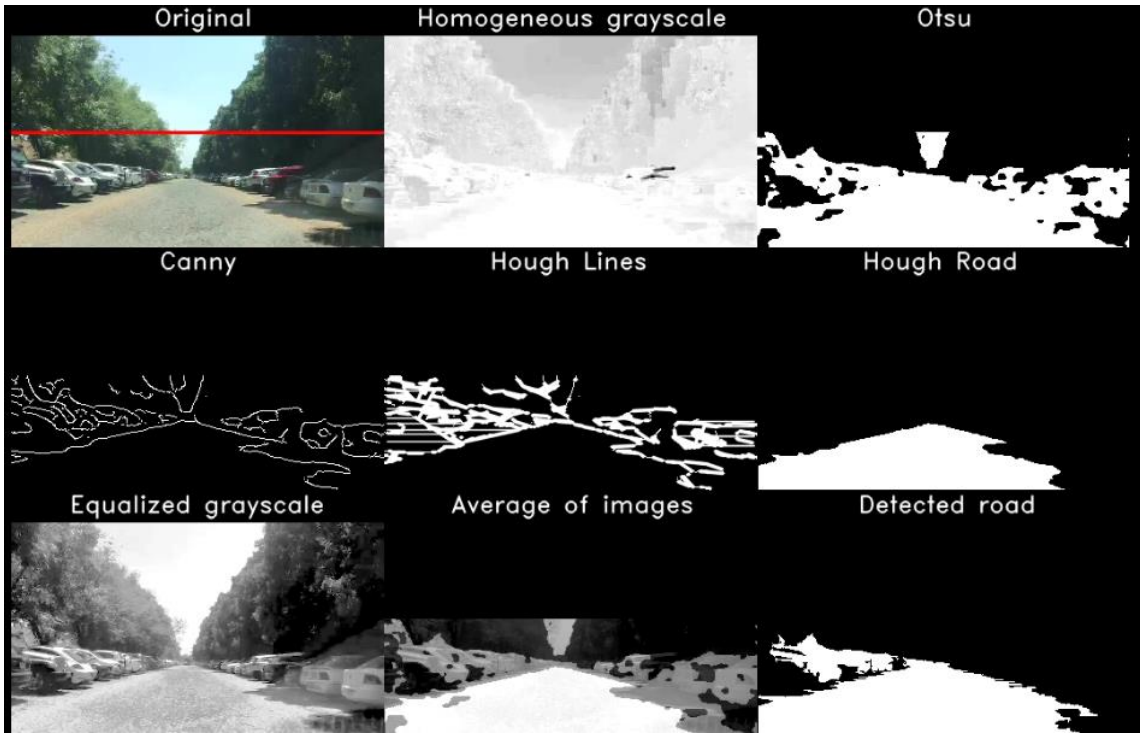

Figura 14. Ejemplo de detalles que se pierden al homogeneizar la iluminación y su corrección con las demás imágenes. Realizado con OpenCV.

En el caso de las líneas de Hough, se decidió que este algoritmo utilice como entrada la escala de grises de la imagen original, de manera que pudiera corregir los errores producidos en Otsu y viceversa, puesto que al utilizar esta imagen como entrada se sigue

siendo susceptible a sombras. En la detección del área navegable tanto de vehículos de manera rudimentaria, siguen existiendo momentos como el de las figuras 15 y 16 en los que no se alcanza a cerrar el contorno que describe a este elemento, quedando sin reconocer en algunos de los cuadros del video.

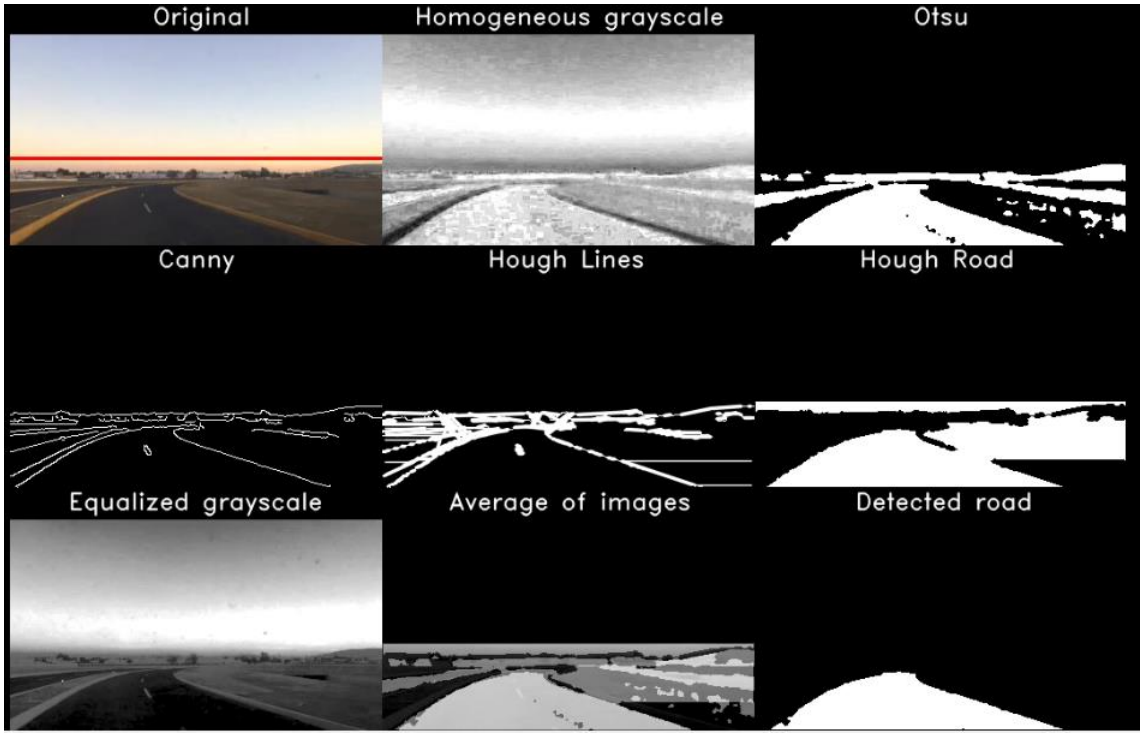

Figura 15. Ejemplo de un borde poco definido que no se alcanza a cerrar al momento de identificar contornos y su corrección con las demás imágenes. Realizado con OpenCV.

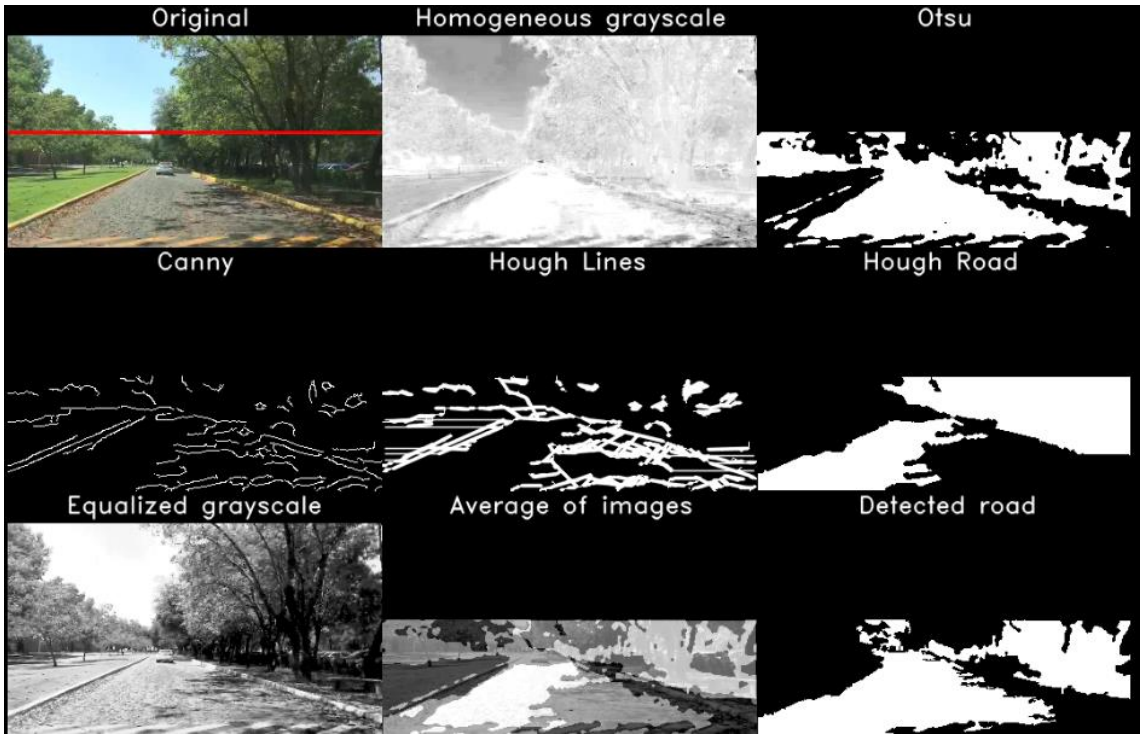

Figura 16. Susceptibilidad a sombras del camino detectado con Hough. Realizado en OpenCV.

En general, considerando el área navegable obtenida después de hacer el promedio de los caminos detectados con Otsu, Hough y la escala de grises de la imagen original ecualizada, se logra reconocer el camino adecuadamente gran parte del tiempo, en más del 95% de los cuadros del video, existiendo ciertas situaciones identificadas donde se puede llegar a perder el camino:

- El vehículo se encuentra dando una vuelta.
- Sombras que envuelven la totalidad de la imagen.
- Señales peatonales no identificadas.
- Bordes poco definidos.

Se hicieron pruebas en diversos entornos y puede afirmarse que su funcionamiento es bastante robusto, pues realizando únicamente la calibración de la selectividad del filtro utilizado para los bordes de Canny pudieron obtenerse resultados similares. Ejemplos de otros escenarios en las figuras 17 y 18.

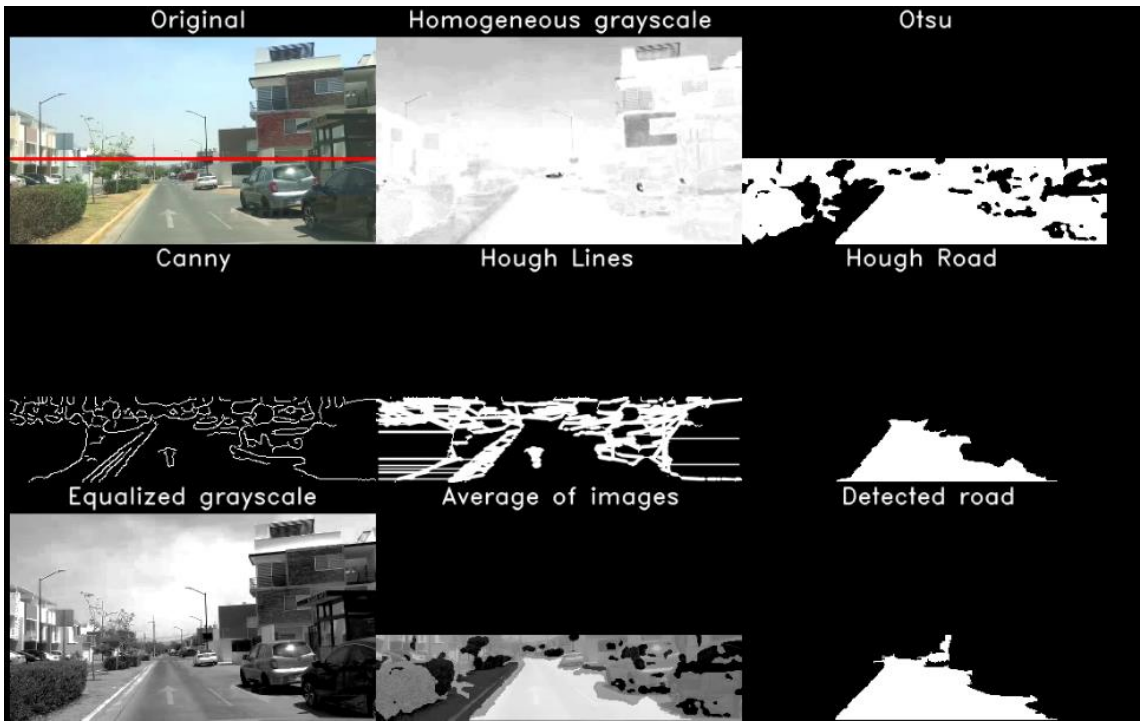

Figura 17. Pruebas de algoritmo en una calle con elementos peatonales bien definidos. Realizado con OpenCV.

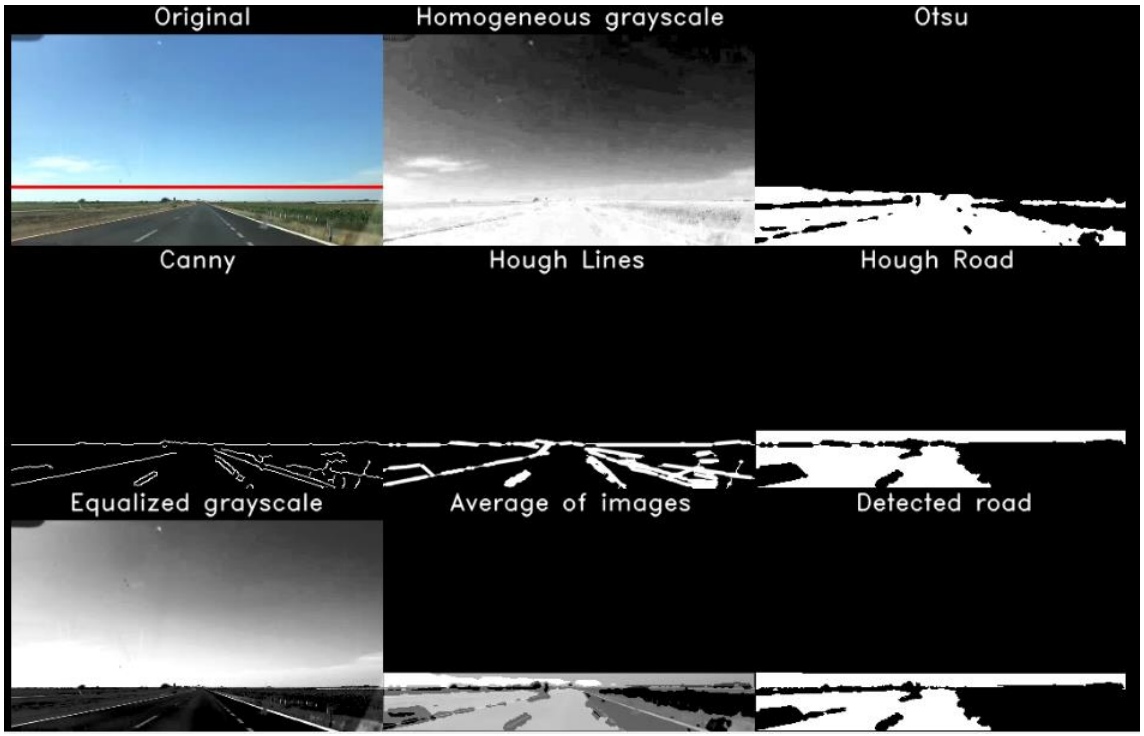

Figura 18. Pruebas del algoritmo funcionando en una carretera. Las líneas peatonales y los bordes poco definidos causaron que Hough no identificara correctamente los límites del camino. Realizado en OpenCV.

Trabajo a futuro

- Validar si vale la pena en cuanto a complejidad y costo computacional un algoritmo que remueva sombras sin tener que cambiar la composición de la imagen.
- Seguir mejorando el análisis de las líneas Hough para detectar con más certeza el camino y los vehículos, además de otros posibles patrones como señales peatonales, personas u otros tipos de obstáculos.
- Incluir en el análisis las componentes de color de la imagen. Pasar de una segmentación binaria a distintas segmentaciones según el patrón de interés.
- Considerar realizar segmentaciones de partes más pequeñas de la imagen, en las direcciones de los carriles, por ejemplo, similar a lo que se hace actualmente con el cielo.
- Combinar la información obtenida en la cámara con otros sensores.
- Implementar los algoritmos obtenidos hasta el momento en un sistema embebido. Validar su viabilidad en cuanto a tiempo de ejecución y complejidad de las instrucciones.

Referencias

[1] N. Otsu, «A threshold selection method from graylevel histogram,» *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 9, nº 1978, pp. 62-66, 1978.

[2] R. O. Duda y P. E. Hart, «Use of the hough transformation to detect lines and curves in pictures,» *IEEE Communications Magazine,* vol. 15, nº 1, pp. 11-15, 1972.

[3] G. Aviña-Cervantes, M. Devy y A. Marín-Hernandez, «Lane Extraction and Tracking for Robot Navigation in Agricultural Applications,» *Proceedings of the IEEE ICAR 2003*, 2003.

[4] K. Yamaguchi, A. Watanabe, T. Naito y Y. Ninomiya, «Road region estimation using a sequence of monocular images,» *2008 19th International Conference on Pattern Recognition*, pp. 1-4, 2008.

[5] A. M. Neto, A. Corrêa Victorino, I. Fantoni y J. V. Ferreira, «Real-Time Estimation of Drivable Image Area based on Monocular Vision,» *IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*, pp. 63-68, 2013.

[6] OpenCV, «OpenCV Tutorials. Image Thresholding,» 8 Marzo 2019. [En línea]. Available: https://docs.opencv.org/3.4.3/d7/d4d/tutorial_py_thresholding.html. [Último acceso: 13 Mayo 2019].

[7] W. Burger y M. J. Burge, «Digital Image Processing,» Springer, 2008, pp. 37-52.

[8] OpenCV, «OpenCV Tutorials. Canny Edge Detector,» 8 Mayo 2019. [En línea]. Available: https://docs.opencv.org/3.1.0/da/d22/tutorial_py_canny.html. [Último acceso: 12 Mayo 2019].

[9] U. Sinha, «AI Shack,» 2017. [En línea]. Available: http://aishack.in/tutorials/canny-edge-detector/. [Último acceso: 23 Marzo 2019].

[10] C. Galambos, J. Kittler y J. Matas, «Progressive probabilistic hough transform for line detection,» *Computer Vision and Pattern Recognition, IEEE Computer Society Conference,* vol. 1, pp. 554-560, 1999.

[11] L. Xin y O. Kazunori, «Hough transform algorithm for real-time pattern recognition using an artificial retina camera,» *Optics Express,* vol. 8, nº 9, pp. 503-508, 2001.

[12] R. Fisher, A. Walker, S. Perkins y E. Wolfart, Hypermedia Image Processing Reference, New York: John Wiley & Sons, 1996.

[13] O.-J. Skrede, «University of Oslo. Department of Informatics,» 8 Marzo 2017. [En línea]. Available: https://www.uio.no/studier/emner/matnat/ifi/INF2310/v17/undervisningsmateriale/slides_inf2310_s17_week08.pdf. [Último acceso: 12 Mayo 2019].

[14] A. M. Neto, A. Corrêa Victorino, I. Fantoni y J. V. Ferreira, «Robust Horizon Finding Algorithm for Real-Time Autonomous Navigation based on Monocular Vision,» *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 532-537, 2011.

[15] M. Hardwick, «Simple Lane Detection with OpenCV,» 20 Marzo 2017. [En línea]. Available: https://medium.com/@mrhwick/simple-lane-detection-with-opencv-bfeb6ae54ec0. [Último acceso: 12 Mayo 2019].