UNIVERSIDAD CATÓLICA

"NUESTRA SEÑORA DE LA ASUNCIÓN"

FACULTAD DE CIENCIAS Y TECNOLOGÍA

INGENIERÍA ELECTROMECÁNICA CON ORIENTACIÓN ELECTRÓNICA

Detección de carriles para la navegación de un vehículo autónomo utilizando visión computacional

Jesús María Franco Santacruz

Hernandarias, agosto de 2020

UNIVERSIDAD CATÓLICA

"NUESTRA SEÑORA DE LA ASUNCIÓN"

FACULTAD DE CIENCIAS Y TECNOLOGÍA

INGENIERÍA ELECTROMECÁNICA CON ORIENTACIÓN ELECTRÓNICA

Detección de carriles para la navegación de un vehículo autónomo utilizando visión computacional

Jesús María Franco Santacruz

Tutor: Ing. Walter Ramón Benítez Dávalos

Cotutor: Lic. en Ing. Electrónica Gregorio Ariel Guerrero Moral

Hernandarias

2020

Jesús María Franco Santacruz

Detección de carriles para la navegación de un vehículo autónomo utilizando visión computacional

Proyecto fin de carrera presentado como requisito parcial para optar al título de Ingeniero en la carrera de Ingeniería Electromecánica con Orientación Electrónica. Facultad de Ciencias y Tecnología, Universidad Católica "Nuestra Señora de la Asunción".

Hernandarias

2020

Franco Santacruz, Jesús María. (2020); Detección de carriles para la navegación de un vehículo autónomo utilizando visión computacional. Hernandarias, Universidad Católica. 150p.

Tutor: Ing. Walter Ramón Benítez Dávalos.
Cotutor: Lic. en Ing. Electrónica Gregorio Ariel Guerrero Moral

Defensa de Proyecto de Fin de Carrera

Palabras claves: ADAS. Detección de carriles. RANSAC. Transformada de Hough. Vehículos autónomos.

Dedicatoria

A mi madre Nidia, por haberme inculcado el valor de la educación y los valores, por distintas renuncias personales que ha tomado para que pueda recibir esta educación formal.

A mi padre Arcenio, por ser un ejemplo de vida de esfuerzo y honradez. Sé que en donde te encuentres estarás orgulloso de tu hijo.

Agradecimientos

A Dios, que ha sido tan generoso conmigo de brindarme la familia en la que crecí, por darme una vida de muchos privilegios que me permitieron llegar hasta este punto, por proveerme la sabiduría, y la fortaleza física y mental, para afrontar los desafíos que se han presentado.

Al profesor tutor, por su orientación, sabiduría, paciencia y comprensión en estos meses de trabajo.

Al profesor cotutor, por su orientación e invaluable experiencia puesta a disposición en estos meses de trabajo.

A mi madre, por ser aquella persona que me apoyó desde un principio, que independientemente de la situación me ha demostrado que el amor de una madre es incomparable y por su esfuerzo de años por el cual pude culminar esta etapa.

A mi novia Penelope, por su apoyo incondicional en todos estos años de relacionamiento, por ser el hombro en el cual pude apoyarme en los momentos de dificultad y por compartir los mayores momentos de felicidad juntos.

A mis familiares y amigos, por brindarme su cariño y apoyo, en especial a mi tío Juan Ángel y mi tía Aida.

A la Itaipú Binacional y la Universidad Católica Nuestra Señora de la Asunción Campus Alto Paraná, por darme la oportunidad de concluir esta carrera de grado.

A los profesores de esta casa de estudios, por transmitirme sus conocimientos con esmero y dedicación. En especial quiero agradecer a los ingenieros Juan Carlos Ocampos, Javier Narváez y Ladislao Aranda, por haber confiado en mi capacidad y demostrado su apoyo en todo momento.

A la National Instruments-Brazil, al Parque Tecnológico Itaipu (PTI-PY) y a las personas por detrás de estas, por haberme brindado la oportunidad de ser *LabVIEW Student Ambassador y* con ello haber tenido un crecimiento profesional acelerado y experiencias invaluables.

A todos los integrantes del Equipo Aguara'i del año 2018 y 2019, por haber confiado en mí, por aventurarse en uno de los desafíos más grandes que tuvimos en nuestras vidas y por lograr nuestro objetivo trabajando en equipo.

"¿Quieres que te dé una fórmula para tener éxito? La verdad es que es muy simple: duplica tu porcentaje de fracasos."

Thomas J. Watson, presidente y CEO de IBM 1914–1956

Resumen

La detección del carril es un problema desafiante y juega un papel fundamental, tanto en los vehículos autónomos como en los sistemas avanzados de asistencia a la conducción (ADAS). Para presentar una solución al problema, el presente proyecto de grado consistió en el estudio, evaluación e implementación de los algoritmos de detección de carriles para la navegación autónoma de un vehículo.

Se han propuesto dos algoritmos basados en procesamiento de imágenes. Ambas propuestas consisten en: modelar linealmente los carriles y extraer las características de los carriles a través de un análisis del gradiente de intensidades. Los algoritmos utilizados para la posterior comprensión de estas características fueron, la transformada probabilística progresiva de Hough para la detección de líneas y el RANSAC (*RANdom SAmple Consensus*) para una estimación robusta del modelo.

Ambos algoritmos fueron evaluados de forma *offline* en dos datasets, TuSimple y Hernandarias (elaborado para el presente proyecto). Finalmente, se han implementado ambos algoritmos como medio de percepción del entorno para mantener al vehículo en su respectivo carril, en el simulador CARLA.

Palabras clave: ADAS. Detección de carriles. RANSAC. Transformada de Hough. Vehículos autónomos.

Abstract

Lane detection is a challenging problem and plays a critical role in both self-driving cars and advanced driver assistance systems (ADAS). To present a solution to the problem, this degree project consisted of the study, evaluation, and implementation of lane detection algorithms for the autonomous navigation of a vehicle.

Two algorithms based on image processing have been proposed. Both proposals consist of linearly modeling the lanes and extracting the characteristics of the lanes through an analysis of the intensity gradient. The algorithms used for the subsequent understanding of these characteristics were the progressive probabilistic Hough transform (PPHT) for line detection and the RANdom SAmple Consensus (RANSAC) for a robust estimation of the model.

Both algorithms were evaluated offline in two datasets, TuSimple and Hernandarias (elaborated for the present project). Finally, both algorithms have been implemented as perception of the environment to keep the vehicle within the current ego (own) lane boundaries, with the CARLA simulator.

Keywords: ADAS. Hough transform. Lane detection. RANSAC. Self-driving cars.

Índice

Índice de figuras

Índice de tablas

Lista de abreviaturas

A2G……... Autonomous Arandu Group

ACC…….…...……...……. Adaptive Cruise Control

ADAS……. Advanced Driver Assistance Systems

ADS……...…... Automated Driving Systems

ALD……... Advanced Lane Detector

API…….……... Application Programming Interface

BGRA…….…… Blue, Green, Red, Alpha

BSD……. Blind Spot Detection

CNN……..… Convolutional Neural Network

CV…….. Computer Vision

DARPA……. Defense Advanced Research Projects Agency

DSC……..... Dice Similarity Coefficient

Euro NCAP…….…. European New Car Assessment Programme

FHA…….. Federal Highway Administration

FNR……... False Negative Rate

FoV……...……... Field of View

FPGA……..……..... Field Programmable Gate Array

FPR……. False Positive Rate

GNSS……. Global Navigation Satellite System

GPS……. Global Positioning System

HIL…….……. Hardware in the Loop

HSV…….…….. Hue, Saturation, Value

IMU…… Inertial Measurement Unit

IP…… Internet Protocol

IPM……. Inverse Perspective Mapping

IR…….……. Infrared

LCA…… Lane Change Assistance

LDWS……... Lane Departure Warning System

LiDAR……. Light Detection and Ranging

LKAS……...…… Lane Keeping Assist System

LSTM……...……... Long Short-Term Memory

MAE…… Mean Absolute Error

MPC…….. Model Predictive Control

MPC2…….……. Second Generation Multi Purpose Camera

NHTSA…… National Highway Traffic Safety Administration

OpenCV…… Open Source Computer Vision

PID……..... Proportional Integral Derivative

PPHT…….….... Progressive Probabilistic Hough Transform

RADAR…….. Radio Detection and Ranging

RANSAC……..……. RANdom SAmple Consensus

RGB……... Red, Green, Blue

RHT……... Randomized Hough transform

RMSE……. Root Mean Square Error

RNN…… Recurrent Neural Network

ROC……..…….... Receiver Operating Characteristic

ROI……...…… Region of Interest

ROS…… Robot Operating System

SAE……...…… Society of Automotive Engineers

SHT…….……... Standard Hough Transform

SVM……... Support Vector Machine

TCP…….…….. Transmission Control Protocol

Introducción

La identificación de carriles en la carretera es una tarea común y aparentemente sencilla, lo realizan todas las personas para garantizar que sus vehículos se encuentren dentro de los límites permitidos y así minimizar las posibilidades de colisiones con otros automóviles. No obstante, la mayoría de los accidentes son causados por fallas humanas, debido a esto se han desarrollado algoritmos para mejorar la seguridad en la conducción.

Estos algoritmos tales como, sistemas de advertencia de abandono de carril (LDWS, por sus siglas en inglés), el control de crucero adaptativo (ACC, por sus siglas en inglés), entre otros, mayoritariamente son la base de los sistemas avanzados de asistencia a la conducción (ADAS, por sus siglas en inglés) y en los vehículos autónomos, donde la detección de carriles juega un papel fundamental en asistir al conductor (Xing et al., 2018).

Desde el punto de vista histórico, una de las primeras concepciones y pruebas de un vehículo autónomo fue dada en el año 1960, cuando General Motors probó un vehículo con los algoritmos de cambio y permanencia en el carril. La introducción al mercado automovilístico de un sistema de asistencia de permanencia en el carril (LKAS, por sus siglas en inglés) fue dada por la empresa Honda en el año 2002, que estaba basada en una cámara y algoritmos de visión computacional para la detección de carriles (Özgüner, Acarman, & Redmill, 2011).

Finalmente, la aceptación de estas tecnologías tanto por el público y la industria, y el incentivo de investigación en este tema, fue dada gracias a las grandes competiciones del 2003, 2005 y 2007

de vehículos autónomos con fines militares promovidas por la *Defense Advanced Research Projects Agency* (DARPA) de los Estados Unidos, con lo cual se evidenció que vehículos pueden conducirse de forma autónoma tanto en entornos urbanos y rurales (Buehler, Iagnemma, & Singh, 2009).

Las consecuencias de lo anterior, tanto en los sistemas ADAS y vehículos autónomos, se pueden observar en el presente y serán desde el punto de vista de los algoritmos de detección de carriles, estudiados, detallados, de forma exhaustiva dentro de este trabajo.

Este proyecto tiene como objetivo detectar carriles para la navegación de un vehículo autónomo, pasando por las etapas de estudio, evaluación e implementación de los algoritmos de visión computacional para lograr dicho fin.

Organización del documento

MARCO TEÓRICO

Capítulo 1. Vehículo Autónomo y Sistemas ADAS. En este capítulo, se realiza una breve introducción respecto a los vehículos autónomos, sobre la arquitectura utilizada comúnmente y la clasificación de estos según el nivel de autonomía que presentan. Además, se introduce el concepto de los sistemas ADAS, ejemplos actuales de la industria y las reglamentaciones en las cuales estos sistemas son puestos a prueba.

Capítulo 2. VISIÓN COMPUTACIONAL. En este capítulo, se explican algunos algoritmos claves de las etapas de la visión computacional, como también sus aplicaciones y un contexto histórico breve de la misma.

Capítulo 3. SISTEMAS DE DETECCIÓN DE CARRILES. En este capítulo, se expone una división de estos sistemas en dos enfoques principales y se ejemplifica esta división a través de algoritmos de detección de carriles pertenecientes a cada enfoque. Se enseña también el flujo de trabajo utilizado en la industria para el desarrollo de estos algoritmos y las métricas de desempeño con las cuales se evalúan.

Capítulo 4. SIMULADORES. En este capítulo, se presentan los simuladores AirSim y CARLA con sus respectivas funcionalidades y características. Finalmente se determina el que será utilizado en este proyecto.

MARCO METODOLÓGICO

Capítulo 5. DISEÑO METODOLÓGICO. En este capítulo, se explica la metodología utilizada para el análisis de la problemática propuesta en este trabajo final de grado.

Capítulo 6. DISEÑO E IMPLEMENTACIÓN. En este capítulo, se exponen la arquitectura de los algoritmos utilizados, la explicación detallada de cada etapa del algoritmo propuesto (desarrollado en el lenguaje de programación Python) y la configuración de la simulación.

Capítulo 7. PRUEBAS Y RESULTADOS. En este capítulo, se presentan los resultados cuantitativos y comentarios, acerca de las evaluaciones realizadas en el dataset TuSimple y en el dataset Hernandarias, como también los resultados de la implementación realizada en el simulador CARLA.

CONSIDERACIONES FINALES

CONCLUSIÓN. Se realiza una conclusión general y específica de lo desarrollado en este trabajo final de grado.

TRABAJOS FUTUROS. Se presentan sugerencias de mejoras a los algoritmos propuestos e ideas para trabajos futuros.

APÉNDICE

Apéndice A. ESTADÍSTICA BÁSICA. En el apéndice, se explica los conceptos básicos de medidas de centralización y de probabilidades.

Apéndice B. CONVOLUCIÓN EN IMÁGENES. En el apéndice, se explica la base matemática de la convolución aplicada a imágenes, ejemplos concretos de su aplicación y se presenta una noción básica del efecto de los valores asociados a la matriz de convolución.

Planteamiento del problema

Para poder mantener a un vehículo ya sea autónomo o que consta de un sistema ADAS en su respectivo carril, es de gran importancia detectar de forma robusta y exacta los carriles. Esta es una tarea compleja que según Xing et al. (2018) puede ser desmenuzada en cuatro aspectos.

Primeramente, se deben extraer las características del carril, como ejemplo, las marcas pintadas en el mismo. El siguiente aspecto es el modelado del carril, en el cual se fija o se estima a partir de las mediciones un modelo ya sea lineal, de segundo o uno de mayor orden a las características extraídas anteriormente. El tercer aspecto consiste en diseñar una detección robusta que tenga la capacidad de predecir las siguientes posiciones de las marcas del carril, y por último se encuentra el coste computacional de ejecutar los pasos anteriores, donde el tiempo de ejecución entre cada iteración debe satisfacer los requisitos de una ejecución en tiempo real.

Cada una de estas partes representan grandes desafíos en el desarrollo de estos algoritmos, puesto que principalmente los dos primeros aspectos deben ser flexibles en su ejecución debido a la variabilidad de la iluminación, a condiciones climáticas adversas, marcas del carril sin buena pintura, marcas de carril oclusas, entre otras razones. El segundo problema está en la suposición de un modelo, las rutas no necesariamente presentan una geometría estándar, por tanto, errores serán generados cuando un carril no concuerda con el modelo supuesto.

En cambio, con respecto al modelo como se ha dicho anteriormente puede no ser supuesto, sino que estimado, pero con la consecuencia de la necesidad de fusionar cámaras con LiDARs y un

coste computacional elevado, debido a una cantidad de datos mayor que procesar (Urmson et al., 2009).

Ante estos problemas se redactan las siguientes preguntas de investigación.

Pregunta general

- ¿Cómo un vehículo autónomo se mantiene en el carril que le corresponde?

Preguntas específicas

- ¿Cuáles son algunos de los algoritmos de visión computacional usados en la detección de carriles?
- ¿Cómo simular la detección de carriles con un algoritmo de visión computacional en un ambiente virtual con un lenguaje de programación?
- ¿Cómo constatar el funcionamiento del algoritmo de visión computacional en un ambiente de un vehículo autónomo?

Objetivo general

- Detectar carriles para la navegación de un vehículo autónomo utilizando visión computacional.

Objetivos específicos

- Estudiar los algoritmos de visión computacional para la detección de carriles mediante una revisión bibliográfica del estado del arte.
- Implementar y evaluar un algoritmo de visión computacional para la detección de carriles a través de un lenguaje de programación.
- Implementar un algoritmo de visión computacional para la detección de carriles en el ambiente de un vehículo autónomo por intermedio de un simulador.

Justificación

La industria automovilística con empresas como Waymo de Google, Cruise de General Motors, Argo AI de Volkswagen AG y Ford Motor Co., se encuentra caminando a pasos agigantados hacia la autonomía en la conducción de vehículos, como también con Tesla, visualizada como un éxito (Bloomberg L.P., 2020). Esto representa una gran oportunidad para incursionar en dicho campo de tal manera a agregar conocimiento del tema no sólo a la región, sino que también a la comunidad educativa a la cual pertenezco.

Desde otra óptica, la detección de carriles es una tarea principal de comprensión del entorno para los vehículos autónomos o los sistemas ADAS, permitiendo que el vehículo tome decisiones de permanecer en su carril o abandonarlo. Para cumplir con la tarea propuesta la solución está directamente relacionada al campo de la visión computacional, el cual es un campo de gran aplicabilidad no sólo a este tema de interés, sino que a variados problemas de la ingeniería.

Por último, este trabajo final de grado está directamente vinculado a la línea de investigación abierta por el proyecto Aguara'i, en el cual he participado en los años 2018 y 2019, y pretende dar seguimiento al mismo agregando nuevas características esenciales que todo vehículo autónomo requiere.

Marco teórico

Capítulo 1

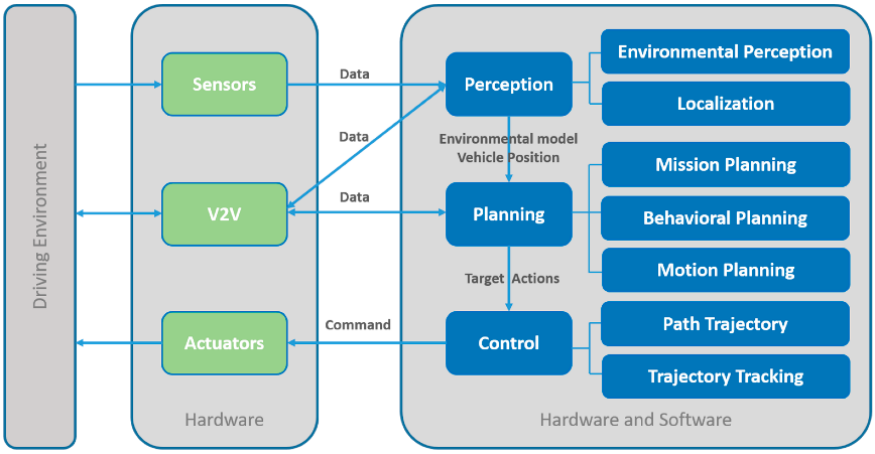
Vehículo autónomo y sistemas ADAS

1.1 Vehículo autónomo

Un vehículo autónomo es aquel que puede manejarse a sí mismo en distintos escenarios, puede navegar por distintos tipos de rutas y entornos, con muy poca o nula interacción de un ser humano (Fagnant & Kockelman, 2015).

De esta manera podemos comprender por manejarse por sí mismo o navegar, como la tarea de que el vehículo autónomo se desplace de un punto A, a un punto B. Este desplazamiento es controlado por la unión de distintos subsistemas que componen un vehículo autónomo, especialmente el subsistema de percepción, lo cual se puede observar en la Figura 1 a seguir, que representa un sistema típico de un vehículo autónomo en el cual se puede evidenciar sus distintas partes y sus conexiones.

Figura 1. Sistema típico de un vehículo autónomo.



Fuente: Rosique, Navarro, Fernández & Padilla, (2019).

1.1.1 Percepción.

Según Chen, Seff, Kornhauser y Xiao (2015) la percepción como su nombre lo indica es la comprensión del entorno, aplicado a un vehículo autónomo, esto se entiende por reconocer distintos objetos relevantes ya sean carriles, señales de tránsito, otros vehículos, pedestres, etc.

Para que la percepción cumpla con su fin esta es dependiente de la información de su entorno. La manera de adquirir esta información es a través de los sensores, donde con el objetivo de diseñar un sistema de percepción más robusto, la información a ser analizada proviene de distintos sensores detallados a continuación.

1.1.2 Sensores.

Los sensores más utilizados para la percepción en la mayoría de los estudios son tres: radares, LiDAR y cámaras.

Según Rosique et al. (2019) los tres sensores citados tienen las siguientes características que serán detalladas a continuación.

1.1.2.1 Radar.

Su principio de funcionamiento consiste en la emisión de una onda electromagnética con longitud de onda en el orden de milímetros, la reflexión de la onda por un objeto a frente del radar y la medición correspondiente del tiempo de vuelo entre la emisión y recepción del eco.

Con esta característica es posible medir la distancia del sensor a un objeto, la velocidad y su dirección. Sus principales usos son: el asistente de cambio de carril (LCA, por sus siglas en inglés), detección de puntos ciegos (BSD, por sus siglas en inglés), entre otros.

1.1.2.2 LiDAR.

Existen en la actualidad distintos tipos de LiDAR los cuales de acuerdo a cómo se obtiene la información del entorno los podemos clasificar de dos o tres dimensiones, así también se los puede clasificar de acuerdo a su construcción, por lo cual pueden ser rotativos o de estado sólido, este último el mayormente utilizado en los vehículos autónomos actuales.

Su principio de funcionamiento consiste en la emisión de un pulso de luz en el orden de los 905 nm a 1550 nm, es decir en la parte infrarroja del espectro electromagnético. De forma parecida al radar lo que se mide es el tiempo de vuelo de este pulso de luz entre la emisión y la recepción.

De esta manera puede ser utilizado para la creación de un mapa de tres dimensiones del entorno con gran precisión a través de la nube de puntos obtenida, permitiendo el control de crucero adaptativo (ACC, por sus siglas en inglés), detección e identificación de objetos, entre otros.

1.1.2.3 Cámara.

Es el sensor comúnmente más utilizado en los sistemas de percepción, pueden ser clasificados en dos tipos según el rango de longitud de onda en la cual trabajan. Los visibles son aquellos en el que el sensor percibe las ondas en el rango de 400 nm a 780 nm, es decir se comportan como el ojo humano, percibiendo la luz visible. El otro tipo son los infrarrojos que trabajan con longitudes de onda infrarrojas, es decir de 780 nm a 1 mm, debido a que trabajan en este espectro sufre menos interferencias de luz.

Separando ambos tipos de cámaras por sus puntos fuertes sus principales aplicaciones son: las visibles que pueden ser utilizadas para la detección e identificación de señales de tránsito, el LCA, la detección de objetos en el entorno entre otras. Por otro lado, las infrarrojas presentan un mejor comportamiento en situaciones de picos de iluminación y en la detección de objetos calientes como pedestres, animales y otros vehículos.

Más detalles de las cámaras como su principio de funcionamiento, el formato de la información recibida, serán abordados en el siguiente capítulo con una revisión más exhaustiva.

1.1.3 Niveles de autonomía de un vehículo autónomo.

En la actualidad existen dos enfoques en los cuales se plantea la siguiente idea, que es la mejora continua de los sistemas ADAS para que en un determinado momento se llegue al nivel en el cual el vehículo sea totalmente autónomo, mientras que otras compañías apuestan por partir de la autonomía total desde el inicio de su proyecto. Como ejemplo de ambos enfoques se puede hablar de Tesla y Waymo respectivamente (Galván, s.f.).

Con el fin de estandarizar y poder comparar el nivel de autonomía de forma clara que tiene un cierto vehículo, entre otros objetivos la Sociedad de Ingenieros Automotrices (SAE, por sus siglas en inglés) ha elaborado el estándar J3016 (SAE International, 2018).

Este estándar de la industria indica que existen seis niveles de autonomía, partiendo del nivel 0, en la cual el conductor debe realizar todas las tareas de conducción y prestar atención en todo momento al entorno, hasta llegar al nivel 5, en el cual la presencia del conductor no es necesaria y el vehículo puede conducirse a sí mismo en distintos entornos, condiciones climáticas, etc. A continuación, se muestra la Figura 2, en la cual se grafica resumidamente para un mejor entendimiento todos los seis niveles.

Figura 2. Niveles de autonomía de un vehículo

Fuente: National Highway Traffic Safety Administration [NHTSA] , s.f. (a).

Según este estándar, los sistemas ADAS están limitados hasta el nivel dos de autonomía, luego ya se tienen los sistemas automáticos de conducción (ADS, por sus siglas en inglés). Cabe destacar que la detección de carriles es una tarea trascendente para todos los niveles de autonomía desde el nivel uno hasta el cinco, pudiendo entonces entenderse que es tanto aplicable a un ADS como a un ADAS.

1.2 Sistemas ADAS

Los sistemas ADAS tienen el objetivo de hacer que la conducción sea más confortable y segura para el conductor, permitiéndole al sistema según sea el caso tomar algunas decisiones primarias como la dirección, el acelerar, frenar y advertirle al conductor sobre peligros (Vaa, Penttinen, & Spyropoulou, 2007).

Con el dominio de estas decisiones primarias se han diseñado distintos sistemas. Estos sistemas pueden clasificarse de dos maneras, sistemas basados en información y basados en manipulación, la diferencia consiste en que, los basados en información únicamente proveen información de interés y advertencias al conductor, en cambio los basados en manipulación toman el control del vehículo en casos de emergencia o cuando sea solicitado por el conductor (Kala, 2016).

Según el autor citado en el párrafo precedente algunas funcionalidades de los sistemas basados en información pueden ser, el replanteamiento dinámico de la ruta a seguir para llegar a un destino elegido, anticipar tráfico y congestiones, obtener información acerca de los vehículos que se encuentran alrededor, como también alertar al conductor cuando este no esté con plena atención en la ruta ya sea por fatiga u otros motivos.

Los sistemas basados en manipulación a diferencia del anterior pueden tomar el control, por ejemplo, el sistema ACC permite que el conductor pueda controlar la dirección del vehículo, pero dejar al sistema el control de la velocidad. Otros sistemas que pueden citarse son, el de frenado de emergencia, la asistencia para el adelantamiento y el estacionamiento automático.

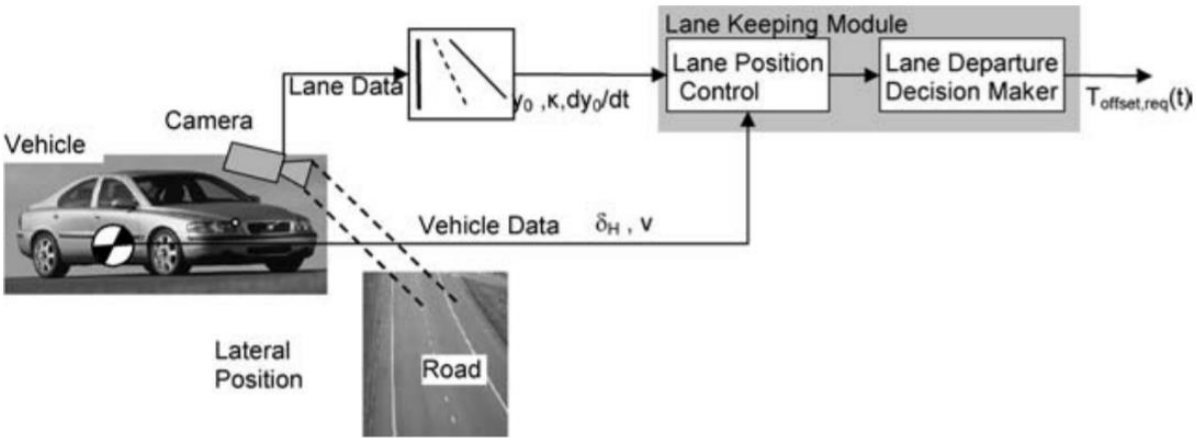
De todos estos sistemas pertenecientes a los sistemas ADAS, se expone a seguir únicamente uno, puesto que se han filtrado los sistemas de acuerdo con su congruencia con este trabajo final de grado.

1.2.1 Sistema de ayuda de permanencia en el carril.

El sistema de ayuda de permanencia en el carril (LKAS, por sus siglas en inglés) es aquel que le permite al vehículo como su nombre lo indica permanecer en el carril ayudando al conductor con el control lateral del vehículo. Este sistema detecta cuando el automóvil se aleja del centro del carril y puede automáticamente sin ninguna acción del conductor ajustar la dirección del vehículo para volver al centro del carril o emitir un aviso al conductor (Pohl, Birk, & Westervall, 2007).

Para que este sistema funcione son necesarias algunas mediciones y estimaciones. La primera de ellas es la estimación de la posición de las marcas del carril en frente del automóvil, la curvatura del carril, la tasa de variación de la posición de las líneas con respecto al tiempo, la medición del ángulo de giro y de la velocidad longitudinal simbolizadas en la Figura 3 como $y_0, \kappa, \frac{dy_0}{dt}, \delta_H, v$ respectivamente.

Figura 3. Módulo de control lateral del vehículo



Fuente: Pohl et al. (2007).

Como se observa en la Figura 3 el módulo de permanencia en el carril consta de 2 módulos: el control de posición en el carril, que calcula de acuerdo con la información recibida el ángulo de giro de referencia y el torque con el fin de mantener al vehículo en el carril, mientras que el módulo de tomada de decisión de abandono de carril decide cuando un abandono de carril ocurre o no.

1.3 Sistemas ADAS comerciales

A lo largo de los años han sido desarrollados por la industria distintos sistemas ADAS por distintas empresas internacionales. Se ha recabado de IndustryARC (s.f.) la lista de las mayores empresas desarrolladoras de estos sistemas de los cuales se han seleccionado tres de ellas y a continuación se detallarán tres ejemplos de un sistema de Alerta de Salida de Carril (LDWS, por sus siglas en inglés) elaborado por las mismas con sus respectivas características y limitaciones.

1.3.1 Serie Mobileye 6.

Las principales características, según Mobileye (s.f.) en su manual del usuario provisto para la serie 6 de Mobileye para el LDWS, son:

- El sistema emite una alerta visual y sonora cuando el vehículo se desvía inadvertidamente del carril. Por inadvertido se comprende que el conductor no ha señalizado utilizando la señal de giro, lo cual al usarlo desactiva el sistema.
- El sistema es un apoyo al conductor no un sistema automático de conducción, por tanto, el conductor debe seguir al mando del vehículo con su propio juicio de valores y su atención al conducir.

- El sistema está activo a partir de velocidades superiores a 65 km/hora.
- El sistema está compuesto por una cámara, un módulo embebido para el procesamiento de imágenes denominado SeeQ® y un display para la visualización del estado del sistema.

A pesar de ser un sistema robusto diseñado por la industria, tiene ciertas limitaciones detalladas a continuación:

- El LDWS no emitirá alertas cuando los carriles no están marcados o claramente marcados, tampoco emitirá al realizarse un giro pronunciado.
- El sistema no es 100 % exacto.
- Las condiciones climáticas pueden afectar negativamente al sistema.

A continuación, en la Figura 4 se muestra el sistema con sus distintas partes.

Figura 4. Serie 6 Mobileye

Fuente: Rosco Collision Avoidance, (s.f.).

1.3.2 Bosch.

Esta empresa proveedora de tecnología está presente en el mercado de los LDWS a través de la cámara multipropósito de segunda generación (MPC2, por sus siglas en inglés). Sus principales características y funcionalidades según Robert Bosch GmbH (2013), se detallan a continuación:

- El MPC2 proveído es del tipo integración, es decir, se vende como una solución para que manufactureras de vehículos puedan agregar a sus vehículos distintos sistemas ADAS, como ejemplo de estudio el LDWS.
- El algoritmo de detección de carriles detecta y clasifica todas las marcas de carril más comunes hasta una distancia de aproximadamente 60 metros y hasta 100 metros en condiciones óptimas de visibilidad.
- El algoritmo es capaz de determinar la desviación lateral y el ángulo de giro del vehículo en el carril con extrema exactitud
- El sistema electrónico provisto dentro de la cámara es una unidad escalable de procesamiento, el cual provee un arreglo de compuertas programables en campo (FPGA, por sus siglas en inglés) y un microprocesador de doble núcleo, lo cual le permite el procesamiento paralelo de imágenes, consiguiendo con esto analizar las imágenes en un corto espacio de tiempo.
- Si el sistema detecta que el conductor se aleja inadvertidamente del centro del carril a velocidades iguales o mayores a 60 km/hora y va a abandonar el carril, el sistema emite una señal visual, audible y/o sensorial.

Figura 5. Cámara multipropósito de segunda generación - MPC2

Fuente: Robert Bosch GmbH (s.f.).

1.3.3 Tesla.

Tesla se ha consagrado como una de las empresas más innovadoras del mercado automovilístico, a través de sus vehículos eléctricos, por sobre todo por su sistema automático de conducción, denominado *Autopilot*. Sin embargo, ha implementado en su software el sistema de ayuda en el carril con o sin el *Autopilot* activado, lo que le permite al conductor activar/desactivar el sistema de prevención de abandono de carril cuando lo considere necesario (Tesla, 2020).

Este sistema según lo indica en el manual de sus vehículos *Model* S, X y 3 presenta las siguientes características y limitaciones:

- El sistema puede comportarse como alerta o asistencia, es decir, cuando ocurre un abandono inadvertido del carril puede alertar al conductor con una vibración en el volante y una alerta visual o en suma a la vibración y la alerta visual, el vehículo tendrá la capacidad de volver hacia el centro de su carril.
- El sistema se activa en el siguiente rango de velocidades 64 km/h a 145 km/h.

- Este sistema puede emitir falsos positivos o falsos negativos en las siguientes condiciones: mala visibilidad debido a lluvias fuertes, niebla o nieve, tampoco en casos de alto brillo incidiendo en la visión de las cámaras, como el atardecer o la luz de otros vehículos, puede no detectar las líneas de carril cuando no estén bien marcadas o cambian abruptamente.

1.4 Reglamentaciones de los sistemas ADAS

Para que un sistema ADAS puede ser colocado en servicio dentro de un vehículo para las masas es necesario que este sistema pase por distintas etapas de validación y pruebas. Existen dos organizaciones, una estadounidense y europea que promueven pruebas de validación de los sistemas a través de puntajes, de los cuales se ahondará a continuación.

1.4.1 National Highway Traffic Safety Administration.

La Administración Nacional de Seguridad del Tráfico en Carreteras (NHTSA, por sus siglas en inglés) es la entidad responsable de asegurar la integridad física de las personas en las carreteras de los Estados Unidos. Esto lo hace a través de normas elaboradas por la misma, que indica los requerimientos mínimos de rendimiento de los vehículos para reducir muertes, lesiones y pérdidas económicas por consecuencia de accidentes de tránsito (NHTSA, s.f. (b)).

La NHTSA ha elaborado la prueba de confirmación del LDWS, en la cual se indica los requerimientos y pruebas necesarias, de los cuales se citarán algunos, por los cuales debe pasar el sistema (NHTSA, 2013).

Algunos requerimientos y pruebas son:

- Las pruebas del LDWS deben ser llevadas a cabo a una velocidad constante de 72 km/hora. Los abandonos de carril deben ser a la izquierda y derecha. Por último, las pruebas se deben realizar en tres diferentes tipos de marcas de carril, líneas blancas continuas, líneas amarillas discontinuas y tachas reflectivas discontinuas.
- Cada condición de prueba deberá ser probada hasta que se alcance cinco pruebas válidas, si se llegan a realizar más que cinco pruebas validas, serán tomadas en consideración las primeras cinco para la aceptación o negación del LDWS.
- El desempeño del LDWS es evaluado considerando la proximidad del vehículo respecto al borde de la línea del carril en el momento de la alerta emitida por el sistema.

Cabe destacar que la NHTSA en su página oficial recomienda a los conductores la compra de vehículos que cuentan con el LDWS, sin embargo, aún no cuentan con ninguna especificación técnica de desempeño para los LKAS, por tanto, no emite un juicio de valor sobre la compra de un vehículo que cuente con este sistema, pero acota que el sistema puede ayudar a una conducción más segura (NHTSA, s.f. (c)).

1.4.2 European New Car Assessment Programme.

Según Euro NCAP (s.f.), el Programa Europeo de Evaluación de Automóviles Nuevos (Euro NCAP, por sus siglas en inglés) es una organización europea compuesta por siete países, organizaciones automovilísticas y de consumidores de todos los países de Europa. Euro NCAP

realiza evaluaciones independientes de vehículos populares vendidos en Europa y se ha convertido en una organización que fomenta mejoras significativas de seguridad en el diseño de vehículos nuevos.

Con el objetivo de establecer un procedimiento de pruebas para los sistemas de ayuda en el carril, han desarrollado el protocolo de pruebas de los sistemas de ayuda en el carril. En este se establecen todos los requerimientos que debe contar el vehículo y específica el procedimiento de pruebas para los LKAS Y LDWS (Euro NCAP, 2019). A continuación, serán citados algunos procedimientos con el afán de mostrar de manera superficial como se llevan a cabo las pruebas de los sistemas citados:

- Las pruebas son llevadas a cabo en una superficie seca y uniforme, no en condiciones de lluvia y ni de visibilidad menor a 1 km como mínimo. La superficie debe ser del tipo pavimentado y no puede contener ninguna irregularidad.
- Las pruebas requieren ser realizadas en dos tipos de marcas de carril, continuas y discontinuas con un ancho de 0,10 a 0,15 metros. Los bordes del carril deben ser pasto y/o grava o cualquier otro sustituto aprobado.
- Las pruebas del sistema LKAS y LDWS, serán realizadas en dos escenarios bajo las siguientes condiciones, con marcas de carril continuas y discontinuas, donde se aumentará la velocidad lateral del vehículo desde 0,2 m/s a 0,5 m/s en intervalos de 0,1 m/s para que pueda abandonar el carril. Las pruebas se realizarán tanto para el lado izquierdo y derecho.

- Sólo se realizarán pruebas del LDWS en el caso que el mismo pueda funcionar de forma separada del LKAS.
- El procedimiento de prueba será acelerar el vehículo bajo prueba hasta alcanzar la velocidad de 72 km/hora y luego aplicar un cambio de dirección con el objetivo de alcanzar la velocidad lateral requerida con un mínimo sobrepaso.
- Finalmente, el fin de la prueba del LDWS termina cuando comienza la advertencia. El fin de la prueba del LKAS culmina cuando el sistema falla y no mantiene al vehículo en su carril dentro de la distancia permitida de abandono de carril o cuando el sistema interviene y mantiene al vehículo en su carril, dentro de la distancia permitida de abandono de carril, causando que el mismo vuelva al centro del carril.

Capítulo 2

Visión computacional

2.1 Aplicaciones

Uno de los fines de la visión computacional es servir a la industria con dos objetivos principales, lograr una mayor interacción entre las máquinas y el entorno que las rodea, y conseguir un control de calidad total de los productos fabricados (Alegre, Pajares, & de la Escalera, 2016).

A comienzos de los años 2000 en el campo de la visión computacional empiezan a utilizarse los algoritmos de aprendizaje automático, como la máquina de vectores de soporte (SVM, por sus siglas en inglés) y los comienzos del uso de las redes neuronales. Esto fue seguido por un punto de inflexión en el año 2012, en el dataset *ImageNet* construido por la universidad de Stanford y Princeton, cuando el algoritmo de redes neuronales convolucionales *AlexNet*, ha demostrado su eficacia en la detección de objetos, al llegar incluso a un porcentaje de error menor a un ser humano en esta tarea a lo largo de los años (Stanford University School of Engineering, 2017).

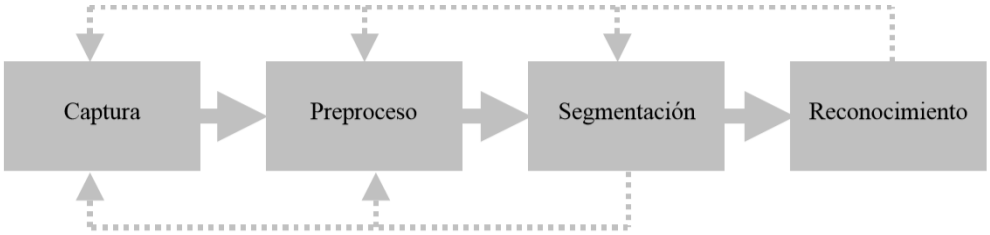
Según la cita anterior, los grandes avances tecnológicos ya sea por el lado del hardware y software, posibilitaron que la visión computacional pudiese ser extendida a más campos tales como: detección automática de rostros que aparecen en una fotografía, reconocimiento automático de las matrículas de los vehículos, procesamiento de lenguaje natural, como también aplicaciones en el ámbito de la seguridad y conducción autónoma, entre otras.

2.2 Definiciones básicas

Según Sánchez Salmerón y Ricolfe Viala (2016) la Visión Computacional (CV, por sus siglas en inglés) puede definirse de la siguiente manera, consiste básicamente en la deducción automática de la estructura y propiedades de un mundo tridimensional, posiblemente dinámico, a partir de una o varias imágenes bidimensionales de ese mundo. Por estructuras y propiedades se pueden comprender por los colores, formas geométricas, textura, etc.

La visión computacional consta de varias etapas visualizadas en la Figura 6. Cada una de estas etapas serán detalladas en las siguientes secciones. Existe una secuencialidad entre cada una de estas etapas, sin embargo, al fallar alguna de estas etapas es normal volver a una etapa anterior o a la primera etapa de captura cuando llegue a fallar por ejemplo el reconocimiento (Vélez Serrano, Moreno Díaz, Sánchez Calle, & Sánchez Marín, 2003).

Figura 6. Etapas de la visión computacional



Fuente: Vélez Serrano et al. (2003).

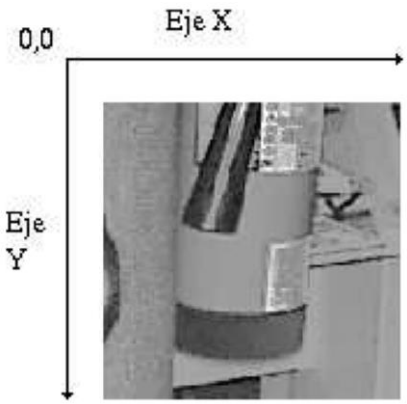
2.2.1 Imagen digital.

Como información principal tenemos la imagen, que nada más es que una representación bidimensional de una escena del mundo tridimensional. Esta es formada a través de la recepción

de una señal electromagnética que se refracta y refleja en la escena para luego llegar al área del sensor de la cámara. A partir de este momento en el sensor se produce la transducción a una señal eléctrica, que luego es digitalizada, formateada, transmitida y almacenada a través de distintos dispositivos electrónicos (Sánchez Salmerón & Ricolfe Viala, 2016).

A la imagen obtenida por el sensor se la puede analizar desde el punto de vista de una función bidimensional que asocia a un valor de posición en un sistema de coordenada a otro valor que corresponde a la intensidad o brillantez en cada punto de la imagen. Para el análisis de una imagen existe la convención de ejes que se muestra en la Figura 7 (Sucar & Gómez, s.f.).

Figura 7. Ejes de coordenadas XY en una imagen



Fuente: Sucar & Gómez (s.f.).

El formato en el cual se utilizan las imágenes con fines de procesamiento es en el digital. Para que se llegue a este formato se lleva a cabo el proceso de digitalización de la imagen que le imprime a la imagen dos características esenciales debido al muestreo y a la cuantización (Vélez Serrano et al., 2003).

El objetivo del muestreo es convertir la imagen continúa provista por el sensor I_C a una imagen digital, es decir discretizar la función bidimensional continua a posiciones discretas. Como resultado se obtiene una imagen digital I_D de $N \times M$ posiciones discretas que pueden ser representadas matricialmente como:

$$I_D(x,y) = \begin{pmatrix} I_D(0,0) & I_D(0,1) & I_D(0,M-1) \\ I_D(1,0) & \cdots & I_D(1,M-1) \\ \vdots & \ddots & \vdots \\ I_D(N-1,0) & \cdots & I_D(N-1,M-1) \end{pmatrix} \tag{2.1}$$

A cada elemento de esta matriz se lo denomina píxel (del inglés, *picture element*).

Por último, resta la cuantización del valor de la función, es decir la discretización de los posibles valores de cada píxel. Con el objetivo de almacenar y realizar cálculos sobre estos píxeles, estos son cuantizados en potencias de 2, por tanto, suelen usarse los siguientes niveles 2, 4, 16, 256 posibles, determinados por la cantidad de bits q. Con esta operación lo que se conseguirá es discretizar los valores de iluminación o brillantez de cada píxel, por ejemplo, en el caso de 256 niveles, yendo del negro que sería el 0 pasando por los niveles de gris hasta alcanzar el 255 que es el color blanco.

De forma matemática se puede expresar este proceso de cambio de I_D que pertenece a $\mathbb{R}$ a I_{DC} (discreta cuantizada) que pertenece a $\mathbb{N}$, de la siguiente manera:

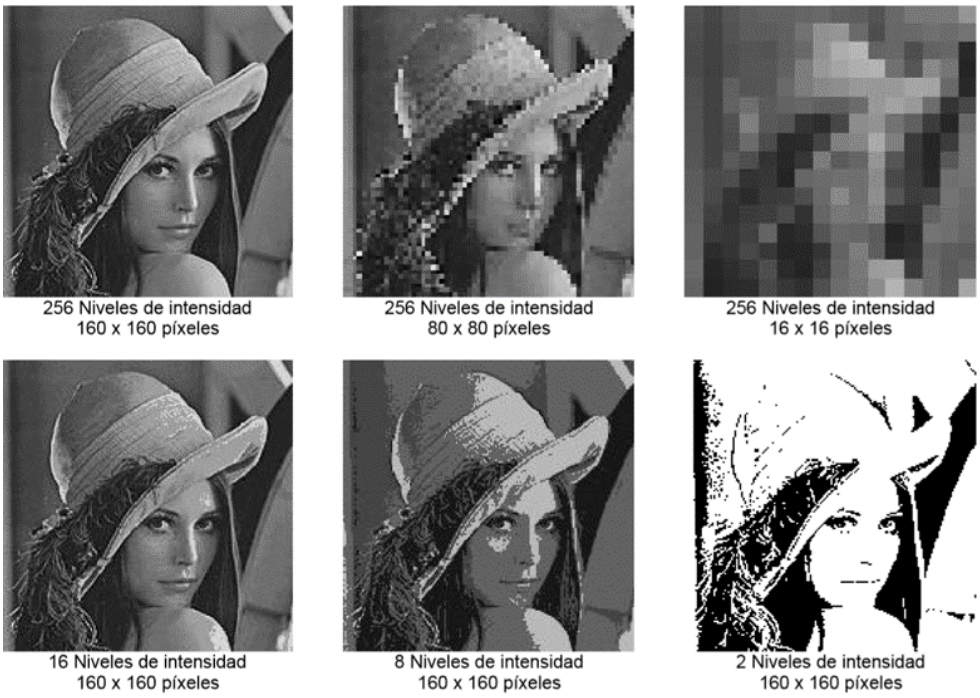
$$I_{DC}(x,y) \in \mathbb{N} \tag{2.2}$$

Donde,

$$x, y \ \in \mathbb{N}\ ;\ 0 \leq \mathrm{x} \leq \mathrm{N}-1\ ;\ 0 \leq \mathrm{y} \leq \mathrm{M}-1$$

$$0 \leq I_{DC}(x, y) \leq 2^q - 1$$

Figura 8. Muestreo y cuantización



Fuente: Vélez Serrano et al. (2003).

En la Figura 8 se puede observar las dos características de la imagen debido al muestreo y la cuantización. Como citado anteriormente el muestreo indica la cantidad de píxeles que tendrá nuestra imagen, esto es posible observar en la primera fila de la Figura 8, cuando a medida que va disminuyendo la cantidad de píxeles en el muestreo, menos información detallada se obtendrá de la escena, es decir, un píxel ocupará una región más grande de la escena. En la segunda fila se observa el efecto de la cuantización, en la cual se percibe el efecto de la disminución de niveles posibles de representación de la intensidad o brillantez de la escena.

Por último, una imagen multiespectral se denota por una función vectorial f de componentes $(f_1, f_2, \dots, f_n)$ donde cada componente es la intensidad de la imagen a diferentes longitudes de onda. Como ejemplo e introducción para el siguiente tópico una imagen a color normalmente es representada con la intensidad de tres longitudes de onda diferentes medidas de la escena (Sucar & Gómez, s.f.).

$$f(x, y) = [f_{rojo}(x, y), f_{azul}(x, y), f_{verde}(x, y),] \tag{2.3}$$

2.2.2 Espacios de color.

Antes de definir qué es un espacio de color, se debe comprender la manera en cómo un ser humano percibe los colores. El ser humano contiene en su retina tres clases de sensores (conos) que tienen más sensibilidad a ciertas longitudes de onda del espectro electromagnético visible. Los cuales son, los sensores tipo α que tienen una mayor sensibilidad a 480 nm (azul), los tipos β a 540 nm (verde) y los últimos el tipo γ a 540 nm (rojo). A partir de esta información provista por las tres clases y luego de combinadas, es como el ser humano puede percibir distintos tipos de colores. Por ende, a estos tres colores citados se los denomina colores primarios (Sucar & Gómez, s.f.).

Los espacios de color se pueden subdividir en dos modelos: los modelos sensoriales que están más enfocados a las cámaras o monitores de televisión y los modelos perceptuales que tienen un enfoque orientado al procesamiento de imágenes y visión computacional. De ambos modelos existen varios espacios de color, sin embargo, sólo se citarán dos que son el RGB y el HSV pertenecientes a cada uno de los modelos.

2.2.2.1 Espacio RGB.

El espacio RGB (*Red, Green, Blue*, por sus siglas en inglés) pertenece a los modelos sensoriales y se basa en la adición de los colores primarios de la luz (rojo, verde y azul) con el fin de cubrir una gran cantidad de colores. Esta adición es realizada en la forma normalizada de las luminancias R, G, B captadas por el sensor, ecuaciones (2.4), (2.5) y (2.6) Las ecuaciones que describen este proceso son dadas a continuación (Martín, García, & Armingol, 2016).

$$R = \int E(\lambda) S_R(\lambda) d\lambda \tag{2.4}$$

$$G = \int E(\lambda) S_G(\lambda) d\lambda \tag{2.5}$$

$$B = \int E(\lambda) S_B(\lambda) d\lambda \tag{2.6}$$

Donde $E(\lambda)$ es la intensidad del espectro de luz incidente en el sensor y $S_R(\lambda)$, $S_G(\lambda)$, $S_B(\lambda)$ son las sensibilidades del sensor a cada longitud de onda, que se puede observarlos desde el punto de vista de filtros de colores.

Normalizando estos valores se obtienen las siguientes ecuaciones,

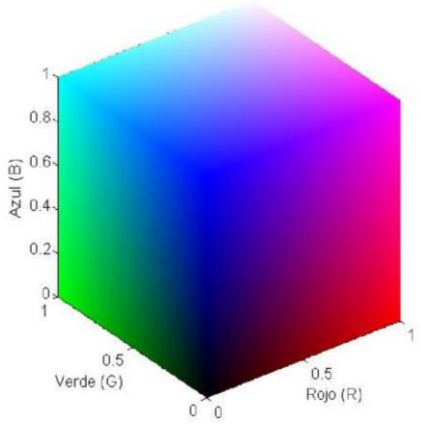
$$r = \frac{R}{R + G + B} \tag{2.7}$$

$$g = \frac{G}{R + G + B} \tag{2.8}$$

$$b = \frac{B}{R + G + B} \tag{2.9}$$

Los cuales permiten construir una representación tridimensional normalizada denominada coordenadas cromáticas, que permiten visualizar a un cierto color como un elemento dentro del espacio determinado por estas coordenadas, Figura 9.

Figura 9. Espacio cromático RGB



Fuente: Martín et al. (2016).

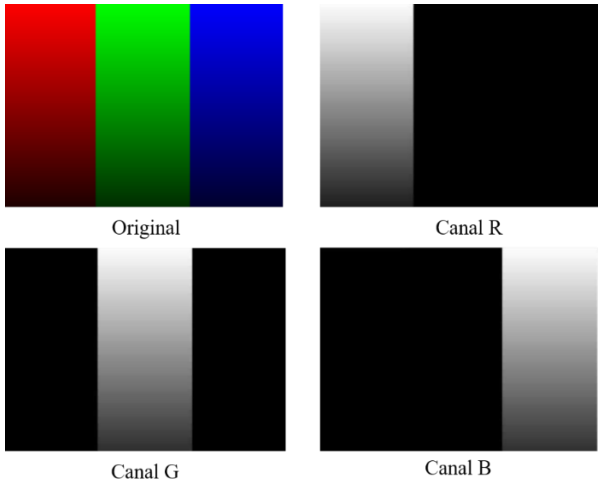
De lo escrito anteriormente y de las ecuaciones (2.7), (2.8) y (2.9), se puede observar que un cierto color x con luminancia L puede ser determinado bajo la adición o mezcla de los tres colores primarios.

$$x = r + g + b \tag{2.10}$$

Para una mayor comprensión y unión con los conceptos pasados, al espacio RGB se lo puede observar desde el punto de vista de tres partes diferentes que se denominarán canales. Cada canal representa la cantidad de cada color primario existente en la imagen, en otras palabras, que tanta luz incidente recibida por el sensor se encuentra en el rango de longitud de onda de cada color primario. Por último, del proceso de cuantización explicado anteriormente se vio que el valor de

cada píxel es decir su intensidad de luz era discretizada bajo convención en 256 niveles desde el 0 que representa al negro, hasta el 255 que representa el blanco, en palabras simples al tener el nivel 1, indica que es nula la presencia de la longitud de onda medida, en cambio si se tiene el nivel 256 representa una alta incidencia de la longitud de onda medida. Esto puede observarse en la siguiente Figura 10 con la imagen original y su separación en los tres canales.

Figura 10. Canales RGB y niveles de intensidad



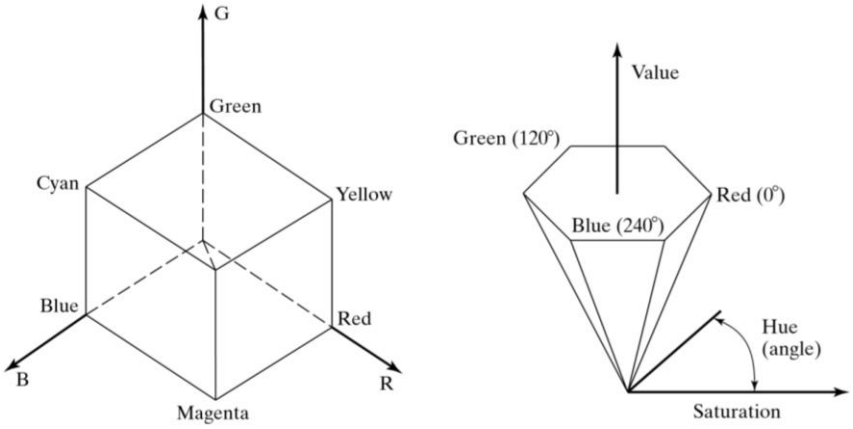
Fuente: Sinha (s.f.).

2.2.2.2 Espacio HSV.

El espacio HSV (*Hue, Saturation, Value*, por sus siglas en inglés) forma parte de los modelos perceptuales debido a que tienen cierta similitud con la percepción humana. Sus tres canales son el (*Hue*, H) que indica el color en términos de tono, es decir es rojo, amarillo, su segundo canal es la (*Saturation*, S) donde su valor indica la "cantidad" de color, por ejemplo a través del canal S se puede diferenciar entre un verde pálido y uno puro, y su último canal es el (*Value*, V) que da una idea de que tan brillante es un cierto color (Sinha, s.f.).

Este espacio de color, a diferencia del RGB que se representa por un cubo, es representado por una pirámide hexagonal invertida Figura 11, donde el canal H es el ángulo formado entre el eje de referencia, el color rojo, y el punto del color en cuestión, el canal S es la distancia del punto representativo del color en cuestión con relación al eje vertical de la pirámide hexagonal invertida y el canal V es el eje vertical. Cabe destacar que este espacio es el resultado de una transformación no lineal del espacio RGB, gráficamente se observa esto en la Figura 11 al "deformar" el cubo del espacio RGB. Las ecuaciones que describen estas transformaciones se encuentran en Hall (1989) según citado en (Szeliski, 2010, pág. 90).

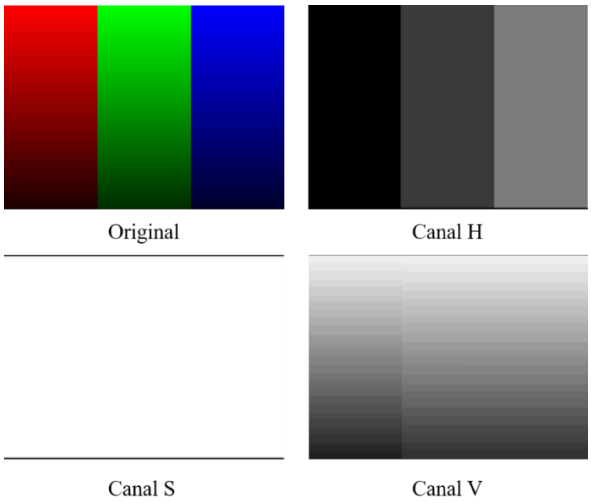
Figura 11. Espacio de color RGB y HSV



Fuente: Forsyth & Ponce (2012).

Una de sus características principales es la independencia del canal H a la variación de iluminación, por tanto, una posible aplicación es en la identificación de colores en lugares con una variación de iluminación. Lo anterior se puede evidenciar en la Figura 12 donde cada color tiene su nivel de gris bien especificado independientemente de la variación de iluminación de la imagen visualizado en el canal H.

Figura 12. Espacio HSV y niveles de intensidad



Fuente: (Sinha (s.f.).

2.2.3 Histograma.

El histograma de una imagen se representa a través de un gráfico, donde en el eje de abscisas se encuentran los niveles de cuantización de los píxeles de la imagen y en el eje de ordenadas el número de píxeles o frecuencia de aparición para cada uno de los niveles de cuantización. Recordando que, si la cuantización es 8 bits, el 0 representa al negro, los siguientes números enteros a los diferentes niveles de gris, hasta llegar al 255 que representa al blanco (Vélez Serrano et al., 2003).

Como se observa en la primera imagen de la Figura 13, su histograma indica poca o nula cantidad de píxeles con un nivel de intensidad cercano al blanco. También se debe notar que, a pesar de ser una representación de la imagen, el histograma no dice nada respecto a la disposición espacial de los píxeles, por tanto, a partir de un histograma es imposible reconstruir una imagen, y hasta es posible que imágenes diferentes tengan el mismo histograma.

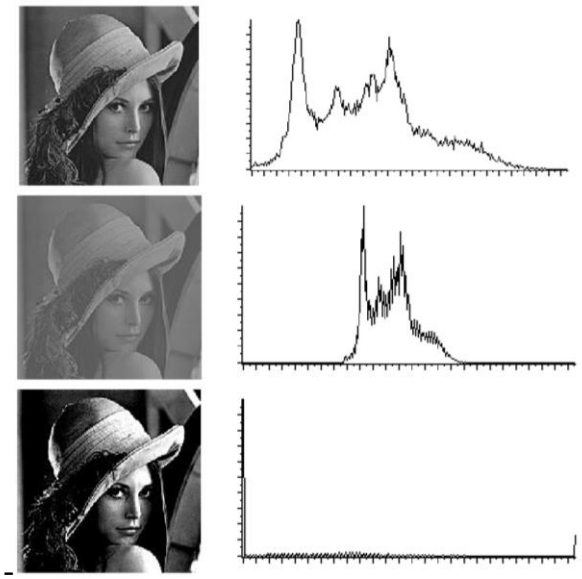
Una de las motivaciones de tener el histograma de una imagen, es poder visualizar el contraste de una imagen que es la diferencia existente en la imagen entre los niveles más claros y oscuros. Para tareas de reconocimiento es de vital importancia que las imágenes tengan un alto nivel de contraste, pero sin llegar a estar saturadas, es decir, no tener picos en los extremos del eje de abscisas del histograma.

En la segunda y tercera imagen de la Figura 13 se pueden observar los efectos de cambios de contraste en una imagen, siendo la segunda una disminución del contraste, por ende, una mayor concentración de píxeles en la región media del histograma. En cambio, en la tercera imagen un aumento del contraste que se visualiza en el histograma con una gran cantidad de píxeles en los extremos, inclusive ya saturados.

Según Prince (2012), matemáticamente un histograma puede ser definido de acuerdo a la ecuación (2.11), donde k indica los niveles de intensidad, p_{ij} la intensidad de cada píxel, y la función $\delta[\cdot]$ da como resultado uno, si el argumento es cero y cero, cuando el argumento es distinto de cero.

$$h_k = \sum_{i=1}^{N}\sum_{j=1}^{M} \delta[p_{ij} - k] \quad k = 0, 1, 2, \dots, 255 \tag{2.11}$$

Figura 13. Histograma de una imagen y cambio de contraste



Fuente: Vélez Serrano et al. (2003).

2.3 Preprocesamiento y segmentación de imágenes

El preprocesamiento consiste en la aplicación de filtros, el realce de ciertas características ya sean bordes, contraste, entre otros, para con ello eliminar partes que no aportan información relevante a nuestro objeto en la imagen o son fuentes de ruido. Posterior a esto se llega a la etapa de segmentación, que tiene como finalidad agrupar conjuntos de elementos para poder comprender la escena, separando nuestro objeto de estudio de lo restante en la escena (Vélez Serrano et al., 2003).

2.3.1 Conversión a escala de grises.

Existen diversos algoritmos para la conversión de una imagen de color a escala de grises y es de uso masivo en la mayoría de las aplicaciones debido a que simplifican los algoritmos de visión

computacional y reducen los requerimientos computacionales. En la mayoría de los usos por no detallar el algoritmo utilizado se cree que la elección del algoritmo no es trascendente, sin embargo, como es citado en artículo científico de los autores Kanan y Cottrell (2012) para el reconocimiento de objetos y rostros, *Gleam* es el mejor algoritmo y para reconocimiento de texturas, *Luminance'* y *Luminance* son buenas elecciones.

Los 3 algoritmos citados son regidos por funciones $\mathcal{G}$ que toman una imagen a color en el espacio $\mathbb{R}^{n\times m\times 3}$ y lo convierten a $\mathbb{R}^{n\times m}$, es decir tres canales bidimensionales que son llevados a una sola representación bidimensional. Todos los valores se asumen que están entre el 0 y el 1. Los valores R, G, B se asumen lineales, es decir sin ninguna corrección gamma aplicadas a los mismos, en contrapartida los valores con corrección gamma se simbolizan R', G', B' y las transformaciones son realizadas a cada píxel.

Las funciones de conversión para cada uno de los algoritmos son:

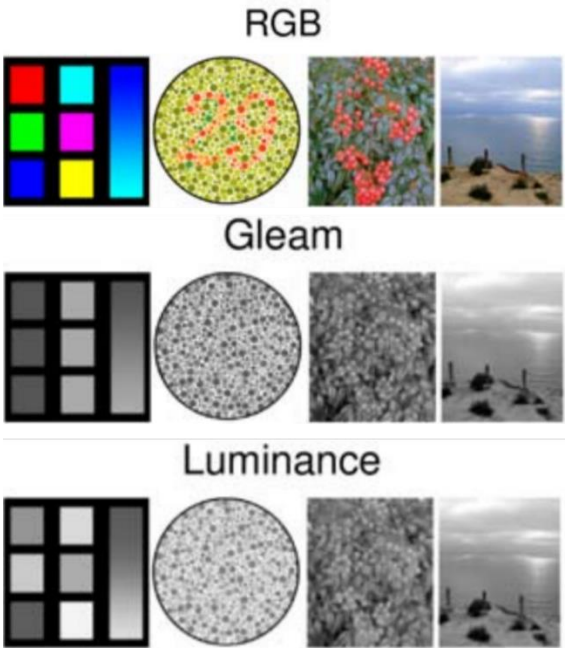
$$\mathcal{G}_{Gleam} = \frac{1}{3}(R' + G' + B') \tag{2.12}$$

$$\mathcal{G}_{Luminance} = 0.3R + 0.59G + 0.11B \tag{2.13}$$

$$\mathcal{G}_{Luminance'} = 0.3R' + 0.59G' + 0.11B' \tag{2.14}$$

La función de corrección gamma estándar utilizada en la mayor parte de las aplicaciones es la siguiente $\Gamma(t) = t' = t^{1/2.2}$. Algunos resultados de la conversión a escala de grises son mostrados en la Figura 14.

Figura 14. Algoritmos de conversión a escala de grises



Fuente: Kanan & Cottrell (2012).

2.3.2 Filtrado.

Según Sucar y Gómez (s.f.), el filtrado como una etapa más del preprocesamiento de la imagen, a través de los distintos filtros paso bajo, paso alto, entre otros, permitirán la atenuación o el aumento de ciertas características como ruido o bordes. La aplicación del filtro a una imagen f consiste en la transformación de la imagen a través de una función T a una nueva imagen g. Matemáticamente se puede escribir:

$$g(x, y) = T[f(x, y)] \tag{2.15}$$

Considerando que la transformación sea lineal, según la teoría de sistemas: el paso de una señal por un sistema lineal tiene como consecuencia que su salida es la convolución de la transformación del sistema con la señal de entrada:

$$g(x,y) = h(x,y) * f(x,y) \tag{2.16}$$

Por el teorema de la convolución podemos escribir la ecuación anterior como:

$$G(u,v) = H(u,v)F(u,v) \tag{2.17}$$

Debido a estas dos ecuaciones se puede concluir que existen dos formas de filtrar una imagen, una de ellas en el dominio espacial a través de la convolución y la segunda en el dominio de la frecuencia a través de la multiplicación de dos transformaciones de Fourier, la primera de espacio a frecuencia y la segunda de frecuencia a espacio. En las siguientes páginas se hablará de los filtros en el dominio del espacio, un filtro paso bajo y otro paso alto. Esta reducción es tomada debido a que fueron los utilizados en el desarrollo del trabajo.

Los filtros en el dominio del espacio son aquellos que son aplicados directamente sobre los píxeles. La aplicación directa sobre los píxeles es realizada a través de un *kernel* o máscara de convolución, en el caso de que sean filtros lineales. Un *kernel* no es más que una matriz que podría considerarse como una pequeña imagen, que en sus elementos define los pesos para un píxel de referencia $f(i,j)$ de la imagen a ser filtrada y sus píxeles vecinos, los cuales luego de multiplicados y sumados, dan el nuevo valor del píxel de referencia. Lo anterior indica que la aplicación de un

kernel sobre una imagen es igual a la convolución de ambas matrices, de la imagen a ser filtrada y el *kernel* (Espinosa Aranda, Fernández Carrobles, & Vállez, 2016).

Matemáticamente esta convolución según la cita anterior se puede expresar para un caso particular de un *kernel* de tamaño 3×3 que será aplicado sobre una imagen f para obtener como resultado una imagen g, como sigue:

$$g(i,j) = \sum_{m=-1,n=-1}^{m=1,n=1} f(i+m, j+n) \times K(m+1, n+1) \tag{2.18}$$

Nótese que no es especificado los valores de los elementos del *kernel*, es decir los pesos que serán aplicados al píxel de referencia y a sus píxeles aledaños. Es en este momento donde el valor de estos elementos definirá de qué tipo de filtro se trata. Un ejemplo de un filtro paso bajo es el filtro de media y su *kernel* correspondiente es:

$$K = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{2.19}$$

De esto se puede comprender que el filtro de media realiza la media aritmética de todos los píxeles vecinos al píxel de referencia, multiplicando el valor de cada píxel por la unidad y luego dividiendo por la cantidad de píxeles tomados por el *kernel*. De manera intuitiva esto generará un suavizado de la imagen.

Para una mayor comprensión de la convolución entre un *kernel* y una imagen, ver apéndice B. Se recomienda la lectura del apéndice anterior y del apéndice A para un mejor entendimiento de las secciones a continuación.

2.3.2.1 Filtro gaussiano.

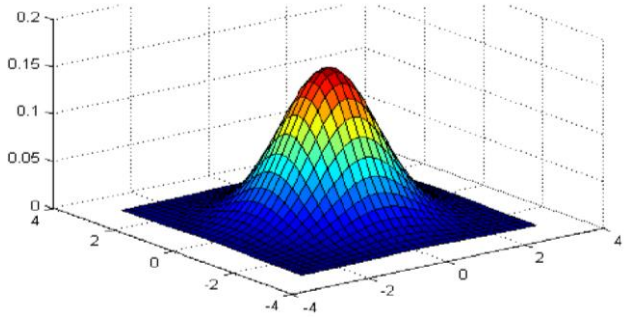
El filtro gaussiano pertenece a los filtros paso bajo. Es uno de los que mejores resultados consigue en la reducción de ruido o suavizado de la imagen. Este filtro proporciona como resultado una mejor imagen sin altas variaciones de frecuencia, que no son de interés para la etapa de detección de bordes (Espinosa Aranda et al., 2016).

Su función de transformación al considerar una distribución gaussiana bivariante donde su media y varianza son las mismas se puede escribir como:

$$T(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.20}$$

La representación gráfica de una distribución Gaussiana con $\mu = 0$ y $\sigma = 1$ se puede observar en la Figura 15 y a seguir su *kernel* correspondiente de tamaño 3×3.

Figura 15. Distribución gaussiana con μ=0 y σ=1

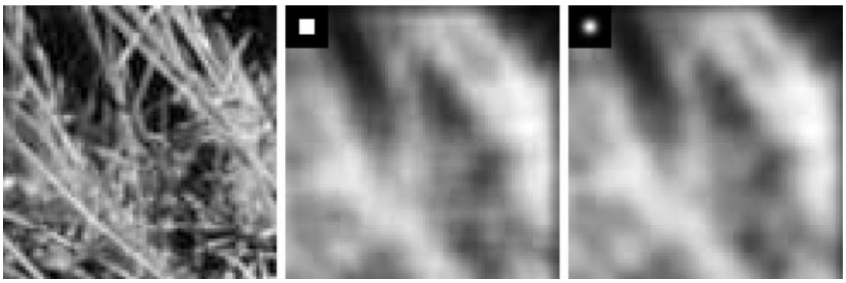


$$K = \begin{pmatrix} 0.153 & 0.156 & 0.153 \\ 0.156 & 0.159 & 0.156 \\ 0.153 & 0.156 & 0.153 \end{pmatrix}$$

Fuente: Espinosa Aranda et al. (2016).

Visualmente la aplicación de un filtro paso bajo de media y gaussiano a una imagen puede observarse en la Figura 16. La imagen de la izquierda es la imagen original, la del centro el resultado de aplicar el filtro de media y a la derecha luego de aplicar el filtro gaussiano. Según se observa en las imágenes filtradas, ambas presentan un grado de suavizado parecido, sin embargo, como se puede observar el filtro de media produce en la imagen barras verticales y horizontales a menudo conocido como efecto *Ringing* o efecto Timbre (Forsyth & Ponce, 2012).

Figura 16. Filtro paso bajo, media y gaussiano



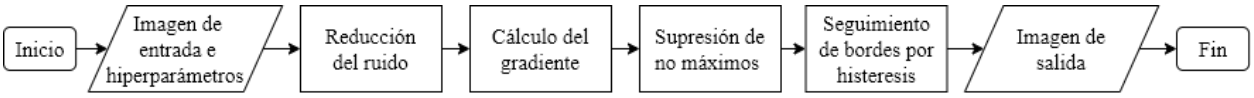
Fuente: Forsyth y Ponce (2012).

2.3.2.2 Filtro de Canny.

El filtro de Canny forma parte de los filtros tipo paso alto, especializado en el realce de bordes, siendo uno de los mejores frente a otros detectores de bordes como los filtros Sobel, Robert, Prewitt y el Laplaciano (Maini & Aggarwal, 2009).

Por detección de bordes se comprende el proceso por el cual se identifica y localiza discontinuidades agudas de la imagen, es decir, cambios abruptos en la intensidad de píxeles vecinos. Para la detección de bordes el algoritmo de Canny se divide en cuatro etapas, que se muestran a seguir en el diagrama de flujo de la Figura 17.

Figura 17. Diagrama de flujo del algoritmo de Canny



Fuente: Maini y Aggarwal (2009).

Como entrada del algoritmo se tienen: la imagen de entrada en escala de grises y los hiperparámetros que serán detallados a continuación. La reducción del ruido consiste en remover la mayor parte del ruido existente en la imagen, debido a que un ruido por lo general representará una discontinuidad aguda, que se puede ser considerado como un posible borde siendo esto un error. Esto es aplicado a través de un filtro paso bajo gaussiano.

El cálculo del gradiente consiste en hallar la magnitud y dirección del gradiente de intensidades en cada punto de la imagen que pueda ser encontrado. Este procedimiento se lleva a cabo a través del operador bidimensional de Sobel que usa dos máscaras convolucionales de 3×3, una para

para estimar el gradiente en la dirección x (columnas) y la otra para estimar el gradiente en la dirección y (filas). Ambas máscaras y la magnitud del gradiente se calculan como sigue:

$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} G_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \tag{2.21}$$

$$G = \sqrt{{G_x}^2 + {G_y}^2} \tag{2.22}$$

La ecuación (2.22) puede ser aproximada por la siguiente ecuación permitiendo así una menor carga computacional en su cálculo.

$$G = |G_x| + |G_y| \tag{2.23}$$

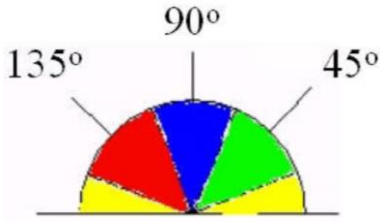
La dirección del gradiente en la dirección se calcula a través de la función inversa tangente,

$$\theta = tan^{-1}(\frac{G_y}{G_x}) \tag{2.24}$$

Cabe resaltar que existe una discontinuidad en el momento en el cual $G_x = 0$. Cuando sucede esto el ángulo resultante es 90° si $G_y \neq 0$ y 0° si $G_y = 0$.

Una vez calculada la dirección del gradiente (borde) es necesario poder analizarlo en congruencia con el arreglo de píxeles de la imagen. Por tanto, para ello se han de establecer cuatro posibles direcciones $0°, 45°, 90°$ y $135°$. Todas las otras direcciones que estén dentro del rango de $67.5° - 112.5°$ por ejemplo serán establecidas a $90°$ y así para los demás tres ángulos que restan. Esto puede ser observado de manera intuitiva por las regiones de color en la Figura 18.

Figura 18. Rangos de dirección de bordes



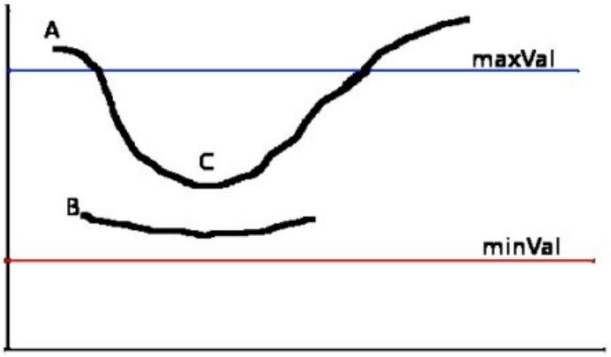
Fuente: Maini y Aggarwal (2009).

Al tener las orientaciones de los gradientes, se pasa a la supresión de no máximos que consiste en la supresión de los píxeles que no constituyen un máximo local alrededor de cada píxel de la imagen en la dirección del gradiente, es decir establecer el valor de intensidad de esos píxeles a 0, lo cual producirá “bordes finos”.

Finalmente, la histéresis hace uso de dos valores umbrales que son los hiperparámetros de entrada dichos anteriormente, donde por hiperparámetros se comprende aquellos valores definidos por el que utiliza el algoritmo. Ambos hiperparámetros son denominados *maxVal* y *minVal,* con un valor alto y bajo respectivamente. Con estos dos umbrales se decidirá cuáles píxeles forman un borde y cuáles no. Para ello se seguirá la siguiente regla ejemplificada a través de la Figura 19.

Considerando los tres puntos de la imagen A, B y C (que representan píxeles), puesto que el punto A tiene un valor de intensidad de su gradiente mayor al umbral *maxVal* automáticamente es considerado como un borde. El punto C está por debajo del umbral *minVal,* pero como está unido al borde al cual pertenece el punto A, por ende, también será considerado como un píxel del borde y por último el punto B o cualquier otro píxel que no esté unido a un píxel considerado como un borde o por debajo del minVal se descartará como un posible borde.

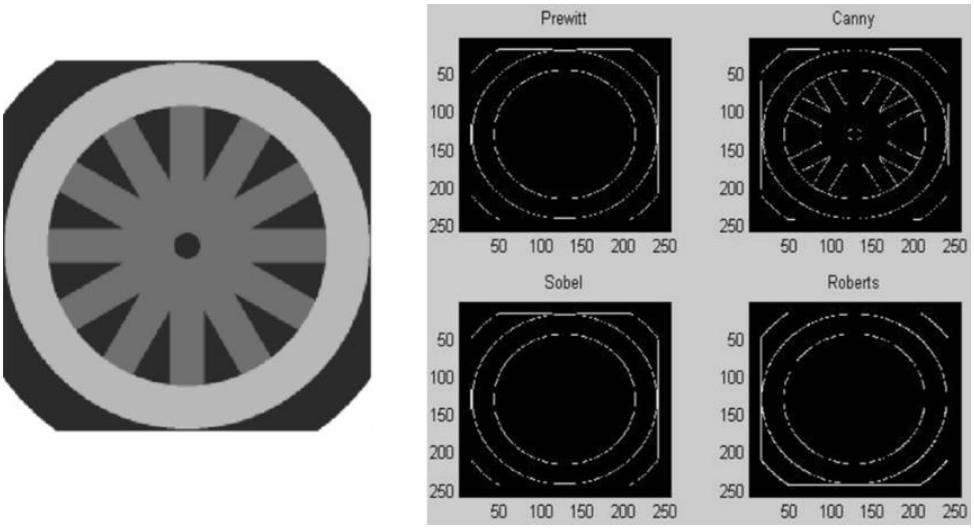
Figura 19. Umbrales de Canny



Fuente: Open Source Computer Vision [OpenCV] (s.f.).

De esta manera el filtro de Canny presenta los mejores resultados a la hora de detección de bordes. A continuación, en la Figura 20 se muestra el resultado de la comparativa con otros detectores de bordes y la segmentación resultante.

Figura 20. Comparativa entre detectores de bordes



Fuente: Maini y Aggarwal (2009).

2.3.3 Umbralización por el método de Otsu.

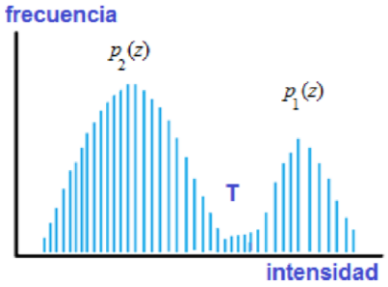
Según Guijarro, Herrera y Montalvo (2016), en la segmentación se desea separar a un objeto de interés del entorno, para ello con técnicas clásicas de visión computacional esto se lleva a cabo a través de la umbralización, es decir un píxel de intensidad $f(x,y)$ será considerado como parte del objeto si $f(x,y) > T$, caso contrario será considerado como parte del entorno, donde T es el umbral que divide estos dos conjuntos. De este proceso se obtiene una imagen binaria, que formalmente se define como sigue.

$$g(x,y) = \begin{cases} 0 \; si \; f(x,y) > T \\ 1 \; si \; f(x,y) \leq T \end{cases} \tag{2.25}$$

Luego, los píxeles asignados con el valor cero son el objeto y los píxeles asignados con el valor uno son el entorno.

El problema de la umbralización reside en la determinación del umbral T de forma óptima, para que con este se pueda segmentar efectivamente la imagen. Para ello Nobuyuki Otsu ha desarrollado una manera automática, no supervisada y no paramétrica de establecer el valor de este umbral (Otsu, 1979).

Figura 21. Histograma visto como una suma de dos funciones de densidad de probabilidad



Fuente: Guijarro et al. (2016).

A seguir se expone el desarrollo matemático y la conjetura de Otsu por tras de la determinación de este umbral. Lo percibido por Otsu fue que el histograma de intensidades de una imagen a segmentar normalmente presenta una semejanza a la suma de dos funciones de densidad de probabilidad gaussianas, como se puede observar en la Figura 21, donde cada lóbulo representa al objeto y al entorno. En consecuencia, para encontrar el umbral óptimo se ha de maximizar la varianza entre ambos lóbulos, donde estas varianzas son funciones de T, en otras palabras, se desea encontrar el valor de T, tal que los valores de ambas funciones estén lo más dispersas posible una de la otra.

Considerando un histograma normalizado bimodal y visto como una distribución de probabilidades, determinada por la variable aleatoria z, que indica los niveles de intensidad de una imagen con L niveles de intensidad, se tiene,

$$p_z = \frac{n_z}{N} \qquad p_z \geq 0, \sum_{z=1}^{L} p_z = 1 \tag{2.26}$$

Donde n_z es la cantidad de píxeles de la imagen con un nivel de intensidad z y N es la cantidad total de píxeles.

Suponiendo que se puede separar la imagen en dos clases, divididas por un nivel de intensidad T, luego la probabilidad de ocurrencia de cada clase, su media y la media de toda la imagen son dados por,

$$\omega_0 = \sum_{z=1}^{T} p_z = \omega(T) \tag{2.27}$$

$$\omega_1 = \sum_{z=T+1}^{L} p_z = 1 - \omega(T) \tag{2.28}$$

$$\mu_0 = \sum_{z=1}^{T} z \frac{p_z}{\omega_0} \tag{2.29}$$

$$\mu_1 = \sum_{z=T+1}^{L} z \frac{p_z}{\omega_1} \tag{2.30}$$

$$\mu = \sum_{z=1}^{L} z p_z \tag{2.31}$$

La varianza entre clases está dada por,

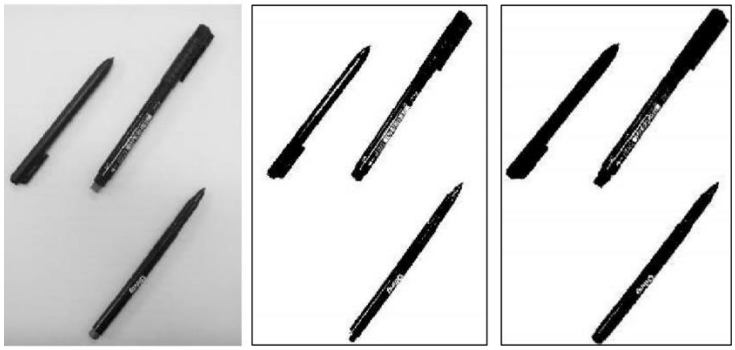
$$\sigma_B^2 = \omega_0(\mu_0 - \mu)^2 + \omega_1(\mu_1 - \mu)^2 \tag{2.32}$$

$$\sigma_B^2 = \omega_0\omega_1(\mu_1 - \mu_0)^2 \tag{2.33}$$

Por tanto, maximizando la ecuación (2.33), se determina el umbral T óptimo que permite segmentar a la imagen en dos clases, objeto y entorno. Los resultados de la aplicación de esta

técnica se pueden observar en la Figura 22. Se observa la imagen original en escala de grises a la izquierda, el resultado de la determinación manual de un umbral $T = 0{,}3$ en el centro y a la derecha el resultado del uso de la técnica de Otsu con un umbral óptimo $T = 0{,}505$. Si se tienen 256 niveles de intensidad, entonces el umbral de Otsu en términos de esta escala de intensidades será 129.

Figura 22. Comparación de umbrales y sus resultados



Fuente: Guijarro et al. (2016).

2.3.4 Región de interés.

A menudo para realizar el análisis de una imagen se considera solamente una parte de la imagen debido a que lo restante no aporta información necesaria a nuestro objetivo de segmentación, reconocimiento o clasificación. Es por ello por lo que a esta región se la llama Región de Interés (ROI, por sus siglas en inglés) la cual es una región por filtrar o procesar de alguna manera. Existen diversas regiones geométricas posibles del ROI, por ejemplo, pueden ser un círculo, cuadrado, un polígono, elipses o áreas con un cierto contorno (The MathWorks, Inc, s.f. (a)).

Un uso común de un ROI es para la creación de una imagen binaria tipo máscara, es decir todos los píxeles que se encuentren dentro de la región del ROI son establecidos a uno en el sentido de

una multiplicación *AND* de la máscara con la imagen original y todos los restantes a cero literalmente porque sus valores de intensidad serán cambiados a cero. Con esto sólo se mantendrán los valores de los píxeles que estén situados bajo el ROI.

Por último, el ROI puede ser estático o dinámico, es decir, esta región puede ser definida por diseño y no variar entre iteraciones del algoritmo de procesamiento, o presentar un comportamiento dinámico donde a cada iteración dependiendo del análisis de la imagen anterior, se defina el mejor ROI (Xing et al., 2018).

2.4 Reconocimiento o clasificación

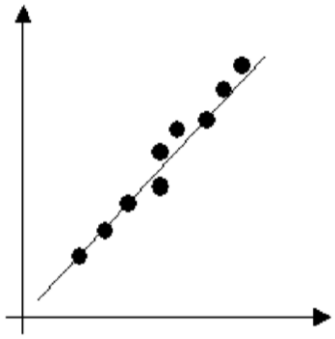
La etapa de reconocimiento o clasificación tiene como objetivo la distinción de los objetos que han sido segmentados. A partir de este momento de acuerdo con las características de nuestro objeto final ya sea esta una persona, vehículo, marcas del carril, entre otros, se podrá distinguir de los objetos segmentados por un cierto análisis cuales corresponden a nuestro objeto final (Vélez Serrano et al., 2003).

2.4.1 Transformada de Hough.

La transformada de Hough conocida como SHT (del inglés *Standard Hough Transform*), forma parte del reconocimiento o clasificación, debido a que a través del análisis de los píxeles segmentados que forman parte de un borde, este algoritmo puede representarlos con distintas curvas, como circunferencias, elipses, líneas, entre otras, siempre y cuando la curva puede ser representada en forma paramétrica (Sucar & Gómez, s.f.).

Se explicará el caso de una línea recta debido a que el proceso es semejante para las otras curvas y es el que fue utilizado en el presente trabajo. Luego del proceso de segmentación y obtención de bordes, estos constituyen un conjunto de puntos $P = \{(x_i, y_i)| \, i = 1, 2, \dots, n\}$ que pueden pertenecer o no a una línea recta. El fin es encontrar la línea recta que mejor modele a este conjunto de puntos como se observa en la Figura 23.

Figura 23. Detección de líneas



Fuente: Sucar y Gómez (s.f.).

Cada punto del conjunto pertenece a todas las líneas que pasen por dicho punto, en otras palabras, todos los pares (ρ, θ) que verifiquen la siguiente ecuación paramétrica de la recta.

$$\rho = x_i cos\theta + y_i sen\theta \tag{2.34}$$

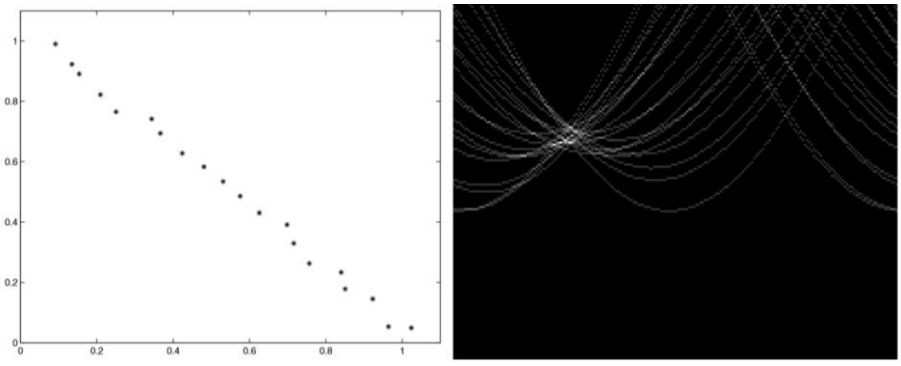
Donde ρ es la distancia perpendicular de la línea al origen y θ es el ángulo formado por esta línea perpendicular y el eje horizontal.

De lo dicho anteriormente, se puede concluir que dos puntos diferentes del conjunto que generen el mismo par (ρ, θ), pertenecen a la misma recta. Entonces cuanto más pares iguales producidos por puntos del conjunto se tenga, con más certeza se modelará la línea recta.

A los efectos de llevar a cabo lo anterior, se construye una matriz bidimensional llamada Acumulador. En esta matriz inicialmente sus elementos son iniciados en cero, sus filas y columnas indican los valores discretizados de ρ y θ, respectivamente. El tamaño de la matriz viene dado por la precisión requerida en los valores de ρ y θ y cada elemento del Acumulador irá aumentando su valor llamado "votos" a medida que un punto del conjunto genere el par (ρ, θ) que define la posición de ese elemento.

El elemento de esta matriz que presente el mayor número de votos dictará el mejor (ρ, θ) que modele con una línea recta al conjunto de puntos P. En aplicaciones prácticas se pueden modelar más de una línea recta, definiendo un valor umbral de votos.

Figura 24. Conjunto de puntos y su espacio de Hough



Fuente: Forsyth y Ponce (2012).

De la Figura 24 se pueden observar las distintas sinusoides generadas por cada punto en la imagen de la izquierda y la intersección mayoritaria de estas sinusoides en un cierto rango de (ρ, θ).

El algoritmo anterior explicado incluso para casos más simples conlleva un tiempo de cómputo muy elevado. Con el objetivo de mejorar el algoritmo Kiryati, Eldar y Bruckstein (1991) han propuesto seleccionar de una manera aleatoria del conjunto de puntos *P* sólo un subconjunto de estos. Conocida como RHT (del inglés *Randomized Hough transform*).

Los experimentos llevados a cabo por los autores seleccionando entre el 5% al 20% de los elementos del conjunto de puntos *P*, han dado como resultado que la transformada probabilística de Hough (RHT) obtiene con un coste computacional considerablemente menor, un desempeño no diferenciable que la SHT. Su única desventaja es que requiere de un conocimiento a priori de los puntos pertenecientes a la línea, lo cual es dificultoso en la práctica.

Matas, Galambos y Kittler (2000) han propuesto una nueva mejora de la transformada de Hough denominada, transformada probabilística progresiva de Hough (PPHT, por sus siglas en inglés) la cual asume que líneas con fuerte soporte de puntos, es decir, en segmentos de líneas largos no es necesario que todos los puntos bajo la línea voten, es decir, basta con algunos puntos para superar el umbral establecido para los elementos del Acumulador.

En la Figura 25 se observa el diagrama de flujo de este algoritmo, donde inicialmente se le introduce tres hiperparámetros y el conjunto de puntos *P*, los hiperparámetros son: el umbral de

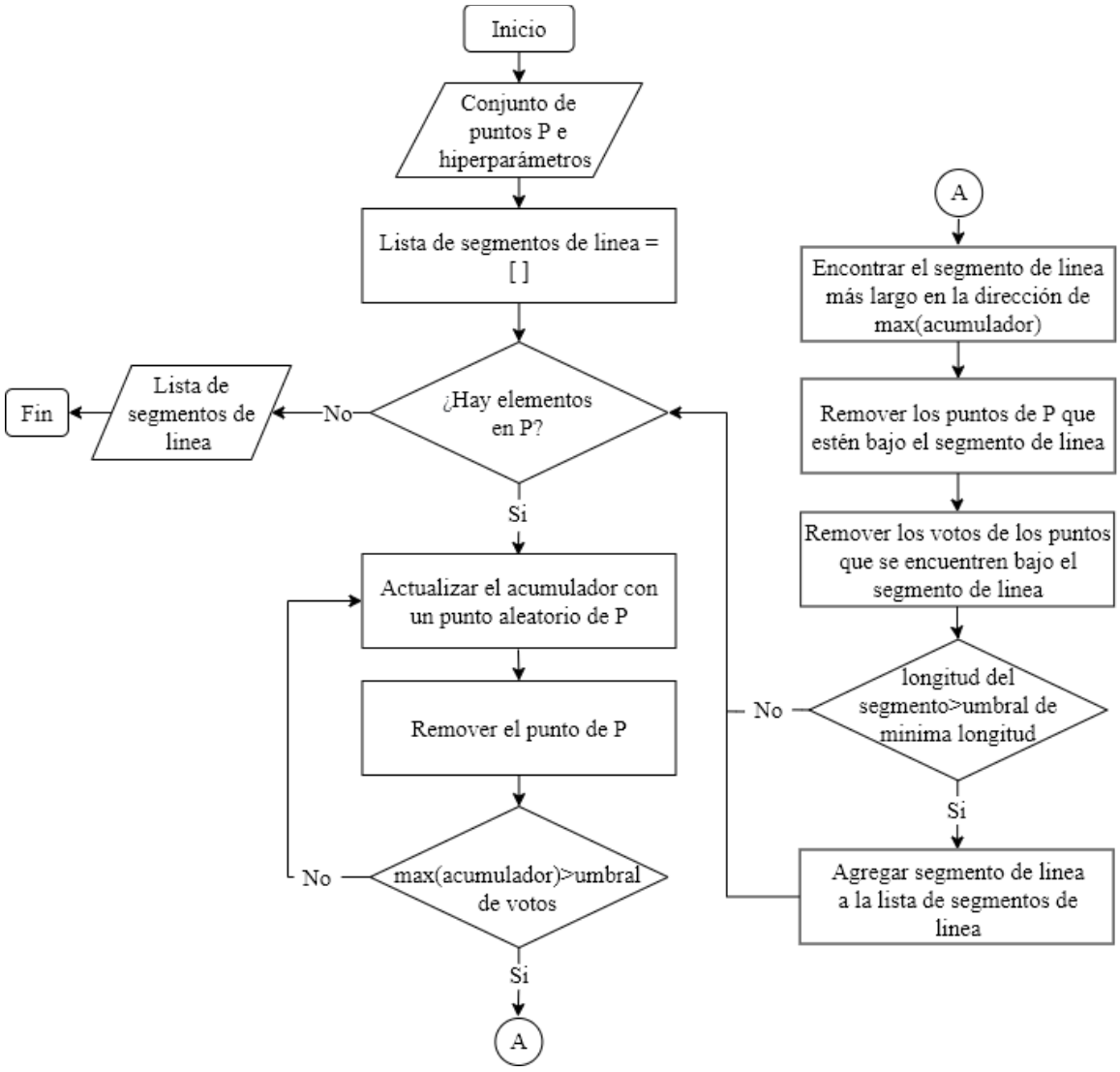
cantidad de votos, la mínima longitud del segmento de línea y la máxima separación entre dos puntos para que sean considerados como pertenecientes al mismo segmento. Como salida se tendrá una lista de segmentos de línea que han cumplido todas las condiciones mencionadas a seguir.

Si existen puntos en P, entonces se calcula el par (ρ, θ) para un punto aleatoriamente seleccionado y se actualiza el acumulador. Seguidamente se retira este punto del conjunto y se verifica si el máximo valor de todos los elementos del acumulador es mayor al umbral de votos, dependiendo de que la condición se satisfaga o no, se volverá a seleccionar un punto aleatoriamente del conjunto o se proseguirá a la siguiente etapa.

A partir de este par con la mayor cantidad de votos, en la dirección indicada por θ se debe encontrar el segmento de línea más largo que sea continuo o exhiba una brecha que no exceda el umbral de máxima separación entre dos puntos. De esta manera teniendo en cuenta este segmento, para disminuir la cantidad de cálculos, se retiran todos los puntos bajo este segmento de línea y sus correspondientes votos.

Por último, se verifica si la longitud de este segmento es mayor al umbral de longitud mínima, si fuese el caso, se agrega a la lista de segmentos de línea, y se repite todo el proceso.

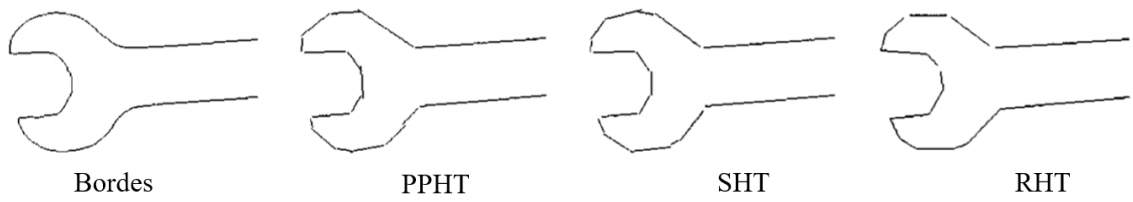
Figura 25. Diagrama de flujo del algoritmo PPHT



Fuente: Matas et al. (2000).

El resultado de aplicar los tres enfoques citados de la transformada de Hough a una imagen se exhibe en la Figura 26. Cabe resaltar que el más eficiente en cuanto a desempeño de modelado de rectas y tiempo de computación es el PPHT.

Figura 26. Variaciones del algoritmo de Hough



Fuente: Matas et al. (2000).

2.4.2 RANSAC.

Otra alternativa para el ajuste de los puntos del borde a una cierta curva es el consenso de la muestra tomada al azar (RANSAC, por sus siglas en inglés). RANSAC posibilita el ajuste robusto de un conjunto de puntos a un cierto modelo, incluso si este conjunto de puntos contiene porcentajes mayores al 50% de puntos atípicos o espurios, llamados en inglés *outliers*, es decir datos considerados como ruido para nuestro modelo (Trespaderne, de la Fuente, & Gómez García Bermejo, 2016).

Según han escrito Forsyth y Ponce (2012) en su libro, la idea por detrás del RANSAC es que, al considerar por ejemplo el modelado de una recta, se han de tomar dos puntos de manera aleatoria del conjunto. Con estos dos puntos se construye un modelo de recta y se verifica si todos los demás puntos del conjunto están dentro de un rango cercano a esta línea. En el caso que se dé lo anterior, claramente se ha encontrado un modelo promisorio para el conjunto de puntos. Esta estimación puede ser mejorada aún más si se toman todos los puntos cercanos, denominados en inglés *inliers*, y se aplica una regresión lineal con un ajuste por mínimos cuadrados a estos puntos.

El algoritmo RANSAC generalizado es detallado a seguir, en conjunto con su diagrama de flujo visualizado en la Figura 27. Para implementar este algoritmo es necesario determinar sus cuatro hiperparámetros: $n, k, t\ y\ d$. A continuación, se dará una explicación para determinarlos.

El número de muestras n, es la cantidad mínima de puntos que permite modelar a la curva deseada. En el caso de una recta se necesitan dos puntos, para una circunferencia tres puntos, y así sucesivamente.

El número de iteraciones requerido k está determinado por la siguiente ecuación,

$$k = \frac{\log(z)}{\log(1 - \omega^n)} \tag{2.35}$$

Donde z es la probabilidad de encontrar un modelo erróneo, normalmente se maneja un valor del 1% y ω es el porcentaje de *inliers* en el conjunto de puntos. A modo de ejemplo, para el caso de modelar con una recta a un conjunto de datos con $\omega \approx 80\%$, se obtiene en cinco iteraciones un modelo con 99% de exactitud.

El umbral t normalmente es definido en el proceso de modelado, puesto que indica para cual distancia de un punto al modelo se lo considera un *inlier* al mismo. A menudo se prueban ciertos valores y se ve qué sucede o se observa un conjunto de datos y visualmente se determina las desviaciones máximas que pueden suceder. Se debe ser cauteloso en su elección puesto que un valor pequeño hará que el modelo posea muy pocos *inliers* y con un valor muy grande el modelo se vea afectado por *outliers*.

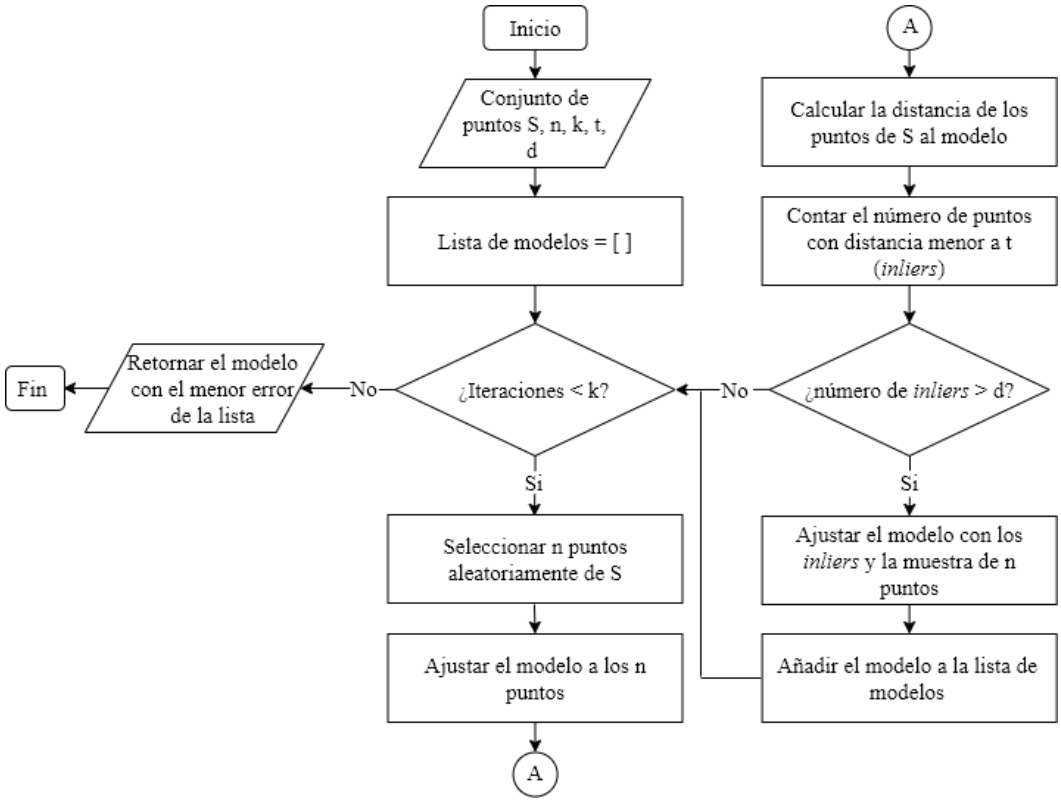
El último parámetro por determinar es la cantidad de *inliers* d que un modelo debe tener para considerarlo válido. Considerando la probabilidad y de que un *outlier* esté dentro de la región de *inliers* determinada por el umbral t anterior, y que se desea elegir un número d de puntos, tal que la probabilidad y^d de que todos los puntos cercanos al modelo son *outliers* sea pequeña, se puede decir menor al 5% y nótese que $y \leq (1 - \omega)$, porque 1-ω es la probabilidad de encontrar *outliers* incluyendo aquellos que están alejados del modelo. De lo anterior se puede tomar d, tal que $(1 - \omega)^d$ sea pequeña.

Al tener definido estos cuatro hiperparámetros y el conjunto de puntos S a modelar. El primer paso del algoritmo consiste en seleccionar aleatoriamente n puntos del conjunto, con estos puntos se calculan los parámetros del modelo, por ejemplo, en el caso de una recta serían los parámetros m y b. Luego para cada punto de S que no corresponda a los n puntos seleccionados, se calcula la distancia del punto al modelo, con esta distancia se puede definir si un punto en cuestión respecto al modelo es un *inlier* o *outlier*, con la condición de que su distancia sea menor a t.

En el caso que el número de *inliers* encontrados en relación al modelo sea mayor a la mínima cantidad de *inliers* para considerar un modelo válido, se ajusta nuevamente el modelo considerando los *inliers* y los n puntos, finalmente se agrega este modelo a la lista de modelos, en cambio si no es mayor, se descarta el modelo y el proceso comienza nuevamente.

Al finalizar el número de iteraciones determinados por k, se elige el modelo de la lista de modelos con la mayor cantidad de *inliers* y con el menor error, según el criterio de error utilizado para el modelo.

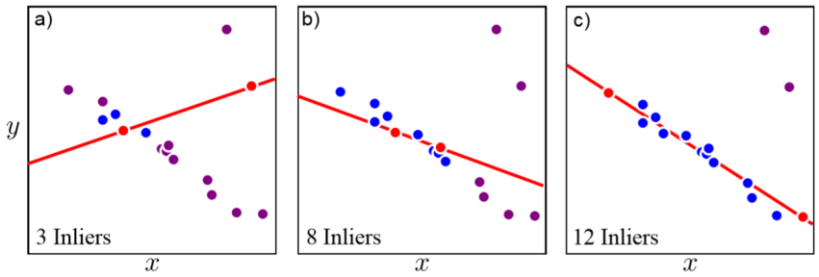
Figura 27. Diagrama de flujo del algoritmo RANSAC



Fuente: Forsyth y Ponce (2012).

En la Figura 28, se puede observar el procedimiento del algoritmo RANSAC para el caso de una recta y el porqué de su robustez frente a los *outliers*.

Figura 28. Procedimiento – RANSAC



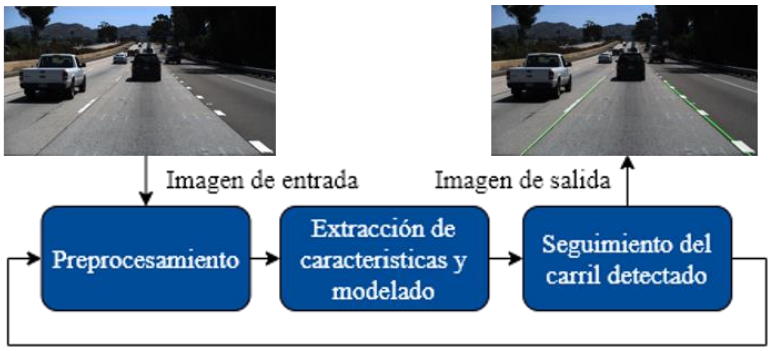
Fuente: Prince (2012).

Capítulo 3

Sistemas de detección de carriles

Según Xing et al. (2018), los sistemas de detección de carriles basados en visión normalmente pueden ser divididos en tres partes principales, preprocesamiento de imágenes, detección de carril y seguimiento de carril. Estas partes se unen y forman el sistema observado en la Figura 29, la cual indica también un lazo de retroalimentación, que por ejemplo podría ser utilizado para lo mencionado anteriormente, obtención de un ROI dinámico.

Figura 29. Sistema de detección de carril usual



Fuente: Xing et al. (2018).

El algoritmo de detección de carril a su vez puede ser dividido en dos principales enfoques, los basados en procesamiento de imágenes y un novedoso enfoque basado en aprendizaje automático.

Los basados en procesamiento de imágenes, consisten normalmente en la detección a través del análisis de las características de marcas del carril, tales como: textura, colores y bordes. Donde a continuación se asumen que los carriles pueden ser descritos con un modelo específico, ya sea

lineal, de segundo orden o distintos tipos de splines y en algunos casos se implementa la predicción y seguimiento para aumentar la robustez del sistema.

Por último, el novedoso enfoque de aplicar el aprendizaje automático se basa principalmente en utilizar redes neuronales convolucionales (CNN, por sus siglas en inglés), algoritmos evolutivos, entre otros. En la literatura algunos de los usos de estos algoritmos en la detección de carriles, fue hecha a través de clasificadores, y regresores para la extracción de características y atributos geométricos de las marcas del carril.

3.1 Detección basada en procesamiento de imágenes

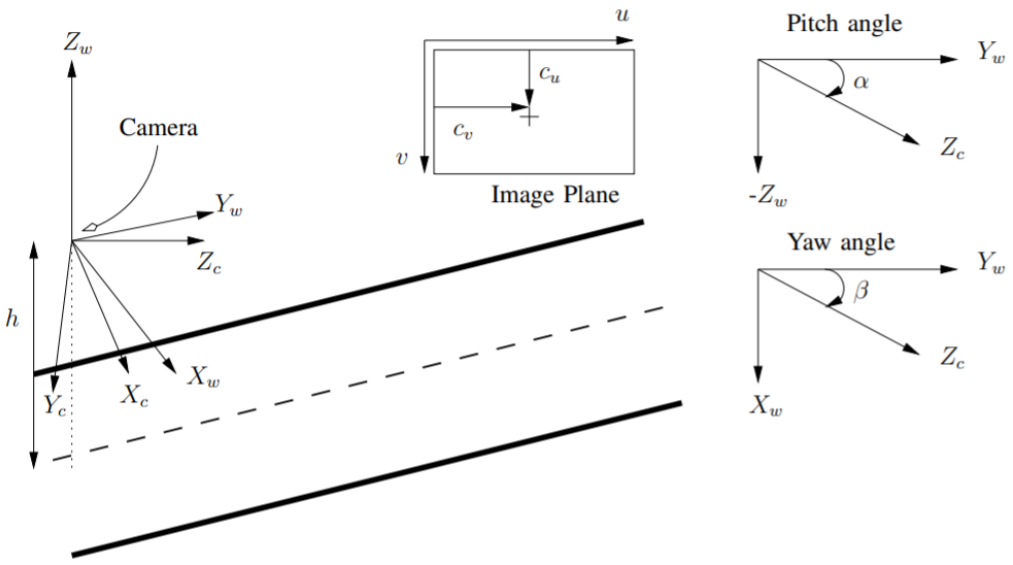
3.1.1 Detección en tiempo real de las marcas de carril en calles urbanas.

El algoritmo propuesto por Aly (2008) para la detección de carriles consta de cinco etapas. Es un algoritmo robusto que se ejecuta en tiempo real y hace uso de algunos de los métodos explicados anteriormente.

La primera etapa consiste en la obtención de una vista de pájaro del carril (IPM, por sus siglas en inglés), es decir una vista superior como si se observara el carril desde lo alto. Primeramente, se asume un carril plano al realizar esta transformación y segundo se deben conocer los parámetros intrínsecos tales como, las distancias focales horizontal y vertical $\{f_u, f_v\}$, los centros ópticos $\{c_u, c_v\}$ y los extrínsecos de la cámara como el ángulo pitch, yaw y la altura que se encuentra la misma del carril.

Para realizar esta transformación debe definirse tres coordenadas, un marco global $\{X_w, Y_w, Z_w\}$ que estará centrado en el centro óptico de la cámara, un marco de la cámara $\{X_c, Y_c, Z_c\}$ y un marco de la imagen $\{u, v\}$. La representación de estas coordenadas es mostrada en la Figura 30.

Figura 30. Relación entre sistemas de coordenadas



Fuente: Aly (2008).

La proyección de un punto cualquiera del marco de la imagen $P_i = \{u, v, 1,1\}$ al plano del carril puede ser encontrado realizando la transformación homogénea a seguir,

$$P_g = {}^{g}_{i}T P_i \tag{3.1}$$

$$
{}^{g}_{i}T = h \begin{pmatrix} -\frac{1}{f_u}c_2 & \frac{1}{f_v}s_1 s_2 & \frac{1}{f_u}c_u c_2 - \frac{1}{f_v}c_v s_1 s_2 - c_1 s_2 & 0 \\ \frac{1}{f_u}s_2 & \frac{1}{f_v}s_1 c_1 & -\frac{1}{f_u}c_u s_2 - \frac{1}{f_v}c_v s_1 c_2 - c_1 c_2 & 0 \\ 0 & \frac{1}{f_v}c_1 & -\frac{1}{f_v}c_v c_1 + s_1 & 0 \\ 0 & -\frac{1}{hf_v}c_1 & \frac{1}{hf_v}c_v c_1 - \frac{1}{h}s_1 & 0 \end{pmatrix}
$$

Donde $c_1 = cos\alpha$, $c_2 = cos\beta$, $s_1 = sin\alpha$ y $s_2 = sin\beta$

La aplicación de la transformación anterior al ROI delimitado por el rectángulo rojo en la imagen a la izquierda de la Figura 31, da como resultado el IPM observado a la derecha. Cabe resaltar la importancia de esta transformación, debido a que con esto se evita que líneas paralelas en el plano del carril se encuentren en un punto denominado (*vanishing point*) por efecto de la perspectiva, sino que se mantienen también paralelas en el IPM, facilitando así la detección de carriles.

Figura 31. IPM aplicado a una imagen

Fuente: Aly (2008).

Como segunda etapa del proceso se realiza el filtrado de la imagen y la aplicación del umbral al resultado. Para llevar a cabo se hace uso del filtrado gaussiano orientado según la dirección vertical y horizontal, sus ecuaciones correspondientes son:

$$f_v(y) = e^{-\frac{1}{2\sigma_y{}^2}y^2} \tag{3.2}$$

$$f_u(x) = \frac{1}{\sigma_x^2} e^{-\frac{x^2}{2\sigma_x{}^2}\left(1-\frac{x^2}{\sigma_x{}^2}\right)} \tag{3.3}$$

El valor de σ_x es ajustado de acuerdo con la longitud del segmento de marca del carril esperado, a su vez σ_y se define por el ancho estimado de estas. Al finalizar el filtrado es aplicado un umbral a los valores de los píxeles en la imagen con el fin de mantener sólo los píxeles con intensidad mayor a un cierto umbral. A diferencia de otros procesos de umbralización el resultado no es una imagen binaria, puesto que se mantienen las intensidades de los píxeles que corresponden a los bordes.

La detección de líneas en la imagen resultante de la etapa anterior constituye la tercera etapa. Se aplican dos algoritmos, el primero es una versión simplificada de la transformada de Hough, para obtener las sumas de las intensidades de los píxeles de cada columna de la imagen. Luego esta suma es suavizada por un filtro gaussiano, donde los máximos locales constituyen las posiciones de las líneas en la imagen. Para obtener una exactitud a nivel de subpíxel de esta posición se ajusta una parábola al máximo local y sus datos contiguos.

Para mejorar aún más la detección alrededor de la línea encontrada por el proceso anterior, se toma un área rectangular y se aplica el modelado de recta por RANSAC a los puntos dentro del rectángulo.

Para poder detectar los carriles curvos en esta cuarta etapa, se modelan los carriles por splines cúbicas de Bézier. El modelado es llevado a cabo por el algoritmo de RANSAC de la misma manera que el paso anterior utilizando los píxeles pertenecientes a un área rectangular alrededor de la línea detectada por el RANSAC de la tercera etapa.

En la quinta y última etapa se aplican tres post-procesamientos, que tienen como objetivo de localizar aún mejor las splines y extenderlas en toda la imagen.

Para mejorar la localización de las splines se muestrean puntos de la spline y a través de estos en la dirección normal a la tangente se trazan segmentos de línea. Sobre estos segmentos de línea se realiza un histograma de niveles de intensidad de los puntos bajo este segmento, que luego de una convolución con un *kernel* de un filtro gaussiano se hallan los máximos locales que deberían corresponder a una mejor localización de los puntos de la spline. Una verificación adicional es hecha, al considerar que el nuevo punto debe estar dentro de un rango esperado de distancia para que se lo considere válido. Al final se hace un reajuste de la spline con estos nuevos puntos.

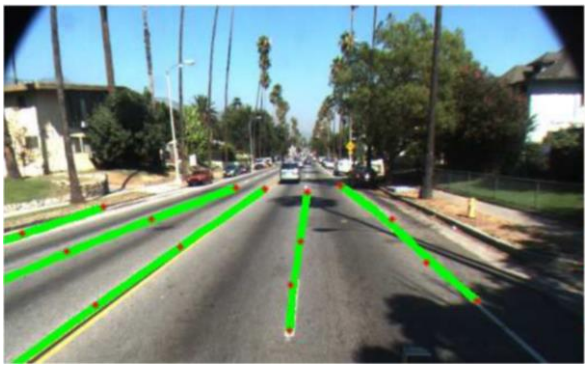
Para extender la spline en ambas direcciones se realiza el mismo proceso explicado en el párrafo anterior, sólo que esta vez los puntos son tomados en la dirección donde probablemente estaría siguiendo la curva. Siempre y cuando los puntos constituyan un máximo local con intensidad

mayor a un umbral y no exista un cambio abrupto en la dirección de la spline, entonces seguirá el proceso de extensión.

El último post-procesamiento es el de las verificaciones geométricas donde se verifican si las splines antes de aplicar los dos pasos previos no son muy cortas ni con una curvatura muy grande, en el caso de que esto se dé, son reemplazadas por las líneas modeladas por RANSAC. Como también se verifica si las splines son prácticamente verticales en el IPM, si no son rechazadas.

En la Figura 32 se muestra el resultado de aplicar estas cinco etapas del algoritmo.

Figura 32. Detección de carriles, algoritmo de Mohamed Aly



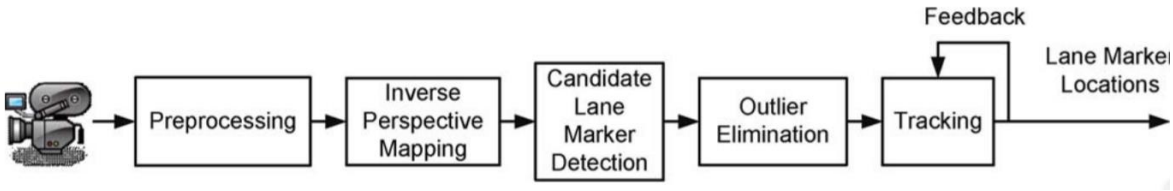
Fuente: Aly (2008).

3.1.2 Un novedoso sistema de detección de carriles con una generación de posiciones reales de las mismas (*Ground Truth*).

Un nuevo enfoque propuesto por Borkar, Hayes y Smith (2012) añade un componente a lo ya descrito hasta el momento que es el Filtro de Kalman, con el objetivo de suavizar la detección de carriles y seguimiento de estos. El sistema propuesto nombrado Detector Avanzado de Carriles 2.0

(ALD 2.0, por sus siglas en inglés) consta de las etapas mostradas en la Figura 33. Cada etapa del sistema es detallada a continuación.

Figura 33. Sistema de detección de carriles con seguimiento



Fuente: Borkar et al. (2012).

La primera etapa del preprocesamiento consiste en la transformación de la imagen de entrada a escala de grises. Esta transformación es realizada con la media aritmética de los tres canales del espacio RGB, es decir, la transformación $Gleam$ sin la aplicación de la corrección gamma. La segunda etapa es la aplicación del difuminado temporal que tendrá el efecto sobre las marcas del carril, puesto que al promediar n fotogramas de una captura continua de imágenes, las marcas de carril discontinuas, luego del promediado formarán una línea. La ecuación que describe este proceso se da a seguir.

$$Difuminado\ temporal = \sum_{i=0}^{N} \frac{I(n-i)}{N} \tag{3.4}$$

Donde $I(n)$ es el fotograma actual, y N es el número de fotogramas a ser promediados.

La segunda etapa del sistema consiste en la obtención del IPM de la imagen. Con esto se logra que las marcas del carril no varíen su ancho a medida que se encuentren más lejanas, sino que

mantienen un ancho constante y sean paralelas con una distancia aproximadamente fija entre las mismas.

Con el objetivo de segmentar la imagen a través de los bordes en ella, se aplica la umbralización adaptativa. Como resultado de aplicar este algoritmo, se tendrá una imagen binaria, donde los bordes tendrán el nivel de intensidad 255 y todo el resto 0. A diferencia de una umbralización global, el método propuesto define el umbral para el píxel analizado de acuerdo a los píxeles vecinos al mismo.

Para la detección de posibles marcas de carril se aplica la transformada de Hough a la imagen binaria, pero con una baja resolución, es decir, en vez de usar una resolución en términos de (ρ, θ) de $\Delta = 0.25, 0.5$ ... se usa un $\Delta = 2.5$. Con esto se logra una disminución del uso de memoria a causa del acumulador y tiempo de cómputo. De este cálculo se toman las diez líneas con la mayor cantidad de votos en cada mitad de la imagen, separando así las marcas de la izquierda y derecha.

Cada una de estas líneas es muestreada de la manera parecida como en el sistema de detección descrito en la sección anterior para estimar el centro de la marca del carril. A diferencia del anterior enfoque, la curva del histograma es comparada con plantillas a través del método coincidencia de plantillas (*Template Matching*). Estas plantillas son elaboradas en base a las normas de Administración Federal de Autopistas (FHA, por sus siglas en inglés), de los cuales son tomadas tres: las marcas finas de 15,24 cm, gruesas de 25,4 cm y dobles. Considerando el difuminado temporal estas plantillas son elaboradas con *kernel*s gaussianos con sus respectivas desviaciones estándar.

El centro de la posible marca del carril se estima en base a cada muestra, donde se elige el píxel con la mayor correlación cruzada normalizada entre el histograma y las tres plantillas, como también esta debe superar un cierto umbral establecido en el entrenamiento de distintos ejemplos de positivos y negativos con el cual se obtuvo la mayor exactitud de la curva característica operativa del receptor (ROC curve, por sus siglas en inglés).

Luego de la estimación del centro de cada marca del carril, se aplica el algoritmo de RANSAC al conjunto de puntos, con la finalidad de eliminar los *outliers*. Al tener el conjunto de *inliers* se aplica la regresión lineal con un ajuste por mínimos cuadrados para ajustar a una recta y con esto se obtiene el mejor modelo de las líneas del carril. Las líneas resultantes nuevamente son parametrizadas en términos ρ y θ para la siguiente etapa.

Finalmente, para el suavizado y el seguimiento de las marcas del carril es utilizado el filtro de Kalman, que estima los valores de ρ y θ en base a las mediciones. El vector de estado $\boldsymbol{x}(n)$ y de observación $\boldsymbol{y}(n)$ son definidos como sigue,

$$\boldsymbol{x}(n) = \boldsymbol{y}(n) = [\rho(n)\ \ \dot{\rho}(n)\ \ \theta(n)\ \ \dot{\theta}(n)]^T \tag{3.5}$$

La matriz de transición de estado es,

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.6}$$

La matriz C en la ecuación de medición del filtro de Kalman es la matriz identidad. El ruido se asume blanco, y cada proceso no correlacionado con los otros. Las matrices de covarianza son constantes y diagonales. En caso de no obtener una medición, se establece la matriz C como nula, forzando así que el filtro de Kalman dependa únicamente de su predicción, aunque si esto sucede un cierto tiempo de segundos se deshabilita el filtro de Kalman para evitar predicciones incorrectas, cuando no estén presentes marcas de carril.

En la Figura 34 se observa el resultado de la aplicación de este sistema en una autopista en horario nocturno.

Figura 34. Detección de carriles, ALD 2.0

Fuente: Borkar et al. (2012).

3.2 Sistemas de aprendizaje automático.

Los avances en el poder de cómputo de la década pasada han hecho posible la utilización de nuevos métodos para detectar carriles. Algunos investigadores han aprovechado esto y redirigido

sus esfuerzos en detectar las marcas del carril usando novedosos métodos, como el aprendizaje automático (Xing et al., 2018).

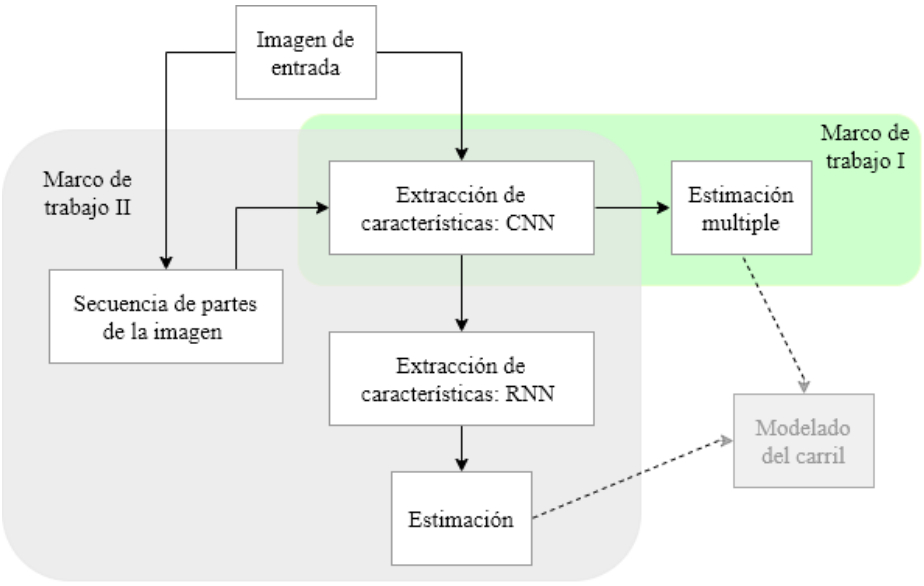
A diferencia de los métodos basados en procesamiento de imágenes, los sistemas de aprendizaje automático proveen un sistema más exacto. Dos de los puntos en contra aún por resolver de estos métodos son: la necesidad de más poder computacional y datos para el entrenamiento del modelo.

3.2.1 Red neuronal profunda para predicción estructural y detección de carriles en escenas de tránsito.

Li, Mei, Prokhorov y Tao (2016) han propuesto el uso de una CNN y una red neuronal recurrente (RNN, por sus siglas en inglés) para la detección de carriles. La CNN realiza dos tareas diferentes, la primera consiste en un clasificador que detecta si existe o no una marca del carril en un cierto ROI y la segunda consiste en un regresor que estima la orientación y posición de la marca del carril con respecto al ROI, siempre y cuando el clasificador arroje un resultado positivo. Para poder detectar la presencia o no de marcas del carril en una secuencia de imágenes y poder inferirlas se hace uso de la memoria interna de una RNN.

Cabe resaltar que este enfoque está enfocado en la detección de bajo nivel, es decir, si en una cierta región de la imagen se encuentra o no una marca del carril. A diferencia de otros trabajos que realizan también la detección de bajo nivel, pero seguidamente ajustan estas marcas a un cierto modelo lineal, de segundo orden y así sucesivamente. El flujo de trabajo de este sistema propuesto se puede observar en la Figura 35.

Figura 35. Sistema de detección de carriles usando redes neuronales convolucionales



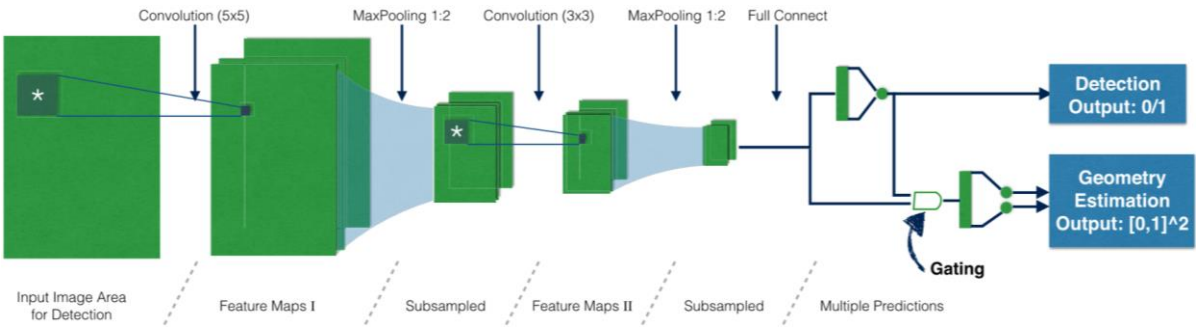
Fuente: Li et al. (2016).

A seguir se dan detalles del diseño de la CNN y la RNN con sus respectivas ecuaciones y figuras.

3.2.1.1 Red neuronal convolucional multitarea.

Esta CNN como se ha dicho anteriormente cumple la tarea de clasificador y regresor. Su diseño se basa en la CNN *LeNet* propuesta por (LeCun, Bottou, Bengio, & Haffner, 1998). En la Figura 36 se puede observar su estructura completa.

Figura 36. CNN multitarea



Fuente: Li et al. (2016).

La imagen de entrada es primeramente transformada aplicando el IPM y seguidamente un ROI de la misma se utiliza como entrada a la red. La extracción de características es realizada a través de dos capas de filtros convolucionales y dos capas de disminución de la resolución de las imágenes.

La primera capa acepta un ROI de 28x20 píxeles, el cual se convoluciona con una serie de filtros de tamaño 5x5, lo cual producirá el primer conjunto de mapa de características. Este mapa de características es reducido a través de la disminución de su resolución espacial, aplicando el muestreo por la técnica de *max-pooling* de los píxeles bajo una matriz de 2x2. Nuevamente el proceso es realizado, pero a diferencia de la anterior, los filtros de la capa convolucional son de tamaño 3x3.

Ahora que las características ya han sido extraídas, están son compartidas entre dos redes neuronales pre-alimentadas, las cuales son el clasificador y el regresor. Ambas redes consisten en una capa oculta de neuronas completamente conectadas a todas las salidas de la capa anterior que luego forman dos modelos lineales para las tareas de estimación.

Las características geométricas estimadas por el regresor son la orientación y la posición del segmento de línea correspondiente a una marca del carril. La orientación es medida en relación con la dirección horizontal del ROI y la posición es determinada por una distancia del centro del ROI al segmento de línea de marca del carril y esta será negativa o positiva dependiendo si está situada por encima o por abajo del centro del ROI.

Las estimaciones de clasificación y regresión se calculan de la siguiente manera,

$$p = \sigma\left(\sum_i \phi_i z_i^C + b^C\right) \tag{3.7}$$

$$(d, \theta)^T = \sum_i \boldsymbol{\psi}_i z_i^R + \boldsymbol{b}^R \tag{3.8}$$

Donde:

p indica la probabilidad de que una parte de la imagen sea o no una marca del carril.

d, θ representan a la distancia y la orientación de cada segmento de línea.

C y R indican clasificación y regresión.

b^C, $\boldsymbol{b}^R$ son el bias de cada modelo lineal.

$\phi_i, \boldsymbol{\psi}_i$ son los coeficientes de cada neurona.

z_i^C, z_i^R son las neuronas en las posiciones i.

σ es una función de activación.

Nótese que letras en negrita indican vectores de 2-D.

Para poder evaluar el desempeño de la clasificación y la regresión, sus funciones de error o de pérdida se definen en base a la probabilidad logarítmica negativa y error cuadrático. Sus ecuaciones se muestran a continuación.

$$L^C = -logp^g(1-p)^{1-g} \tag{3.9}$$

$$L^R = [(d-\hat{d})^2 + (\theta-\hat{\theta})^2]g \tag{3.10}$$

Donde

$g \epsilon \{0,1\}$ y es la etiqueta que indica realmente si existe o no una marca del carril.

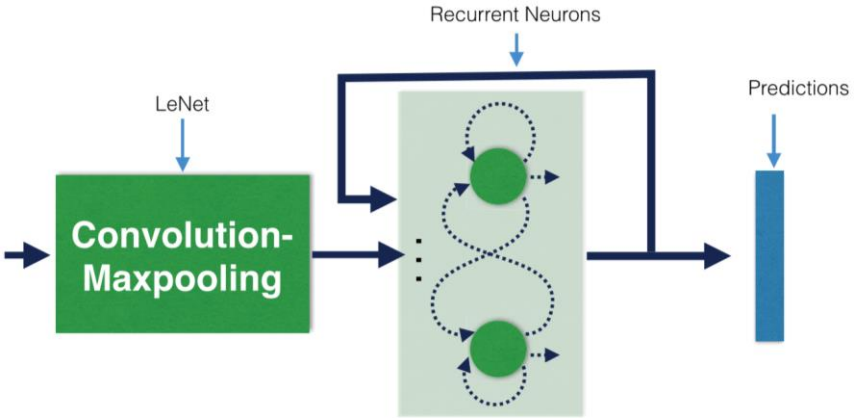
$\hat{d}, \hat{\theta}$ indican la distancia y orientación real de cada marca del carril.

Por último, cabe resaltar el término g en la función de error del regresor y el funcionamiento de la compuerta que tiene como interruptor la predicción del clasificador. La relevancia del término g es debido a que cuando el clasificador no detecta ninguna marca del carril es innecesario el cálculo de error. Siguiendo el mismo razonamiento durante el proceso de propagación hacia atrás la compuerta es controlada por el estado de g.

3.2.1.2 Red neuronal recurrente.

Para poder comprender la estructura global de las marcas del carril a través de detecciones locales en la imagen se hace uso de la memoria interna de una RNN que con una secuencia de imágenes puede inferir esta estructura global del carril incluso cuando las marcas del carril sólo son parcialmente observadas. La estructura de esta red se detalla a seguir.

Figura 37. Estructura de la RNN

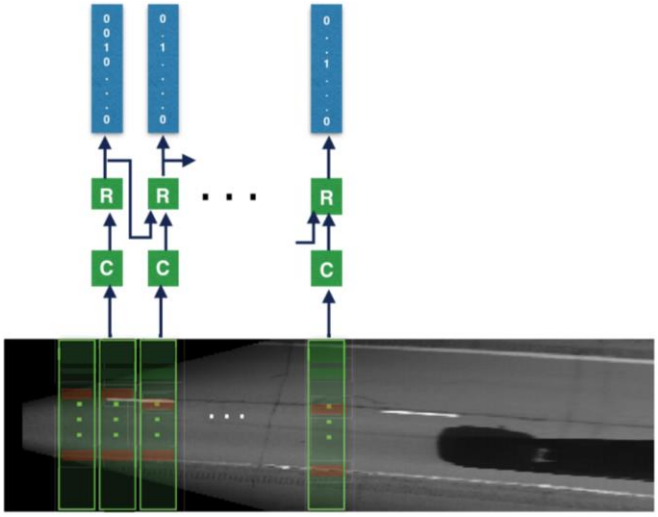


Fuente: Li et al. (2016).

Como se observa en la Figura 37 esta red cuenta con tres componentes principales: una etapa de extracción de características llevadas a cabo por la aplicación de una convolución seguida de un proceso de reducción de la resolución espacial con un sondeo máximo de los píxeles, luego estos datos fluyen a través de una capa oculta de neuronas recurrentes y por último una capa de clasificadores toman los estados de las neuronas recurrentes y producen sus salidas respectivas.

Primeramente, la imagen de entrada ya aplicado el IPM, es dividida en ROIs de 10x80 píxeles que son inseridos consecutivamente a la red. En la Figura 38 se puede observarlos por los rectángulos verdes perpendiculares al carril. Esta región nuevamente es subdividida en pequeñas regiones rectangulares de 10x5 píxeles las cuales efectivamente serán clasificadas según contengan o no una marca del carril. En la Figura 38, “C” significa la etapa de convolución y sondeo máximo, “R” significa la etapa de neuronas recurrentes y los dígitos unos y ceros la clasificación según exista o no una marca del carril.

Figura 38. RNN aplicado a una imagen



Fuente: Li et al. (2016).

A diferencia de la CNN anterior esta cuenta con una sola capa de convolución con sus filtros respectivos de tamaño 5x5, y una sola aplicación de la reducción de resolución espacial a través del sondeo máximo igual de una matriz de tamaño 2x2. La última diferencia consiste en que esta capa no cuenta con una función de activación debido a que la siguiente capa de neuronas recurrentes cuenta con complejas neuronas con funciones de activación no lineales.

La capa de neuronas recurrentes se lleva a cabo a través de la arquitectura de memoria a corto y largo plazo (LSTM, por sus siglas en inglés). Esta arquitectura permitirá distinguir cuales son las regiones que necesitan almacenar, luego la salida será establecida por las compuertas y sus respectivos pesos.

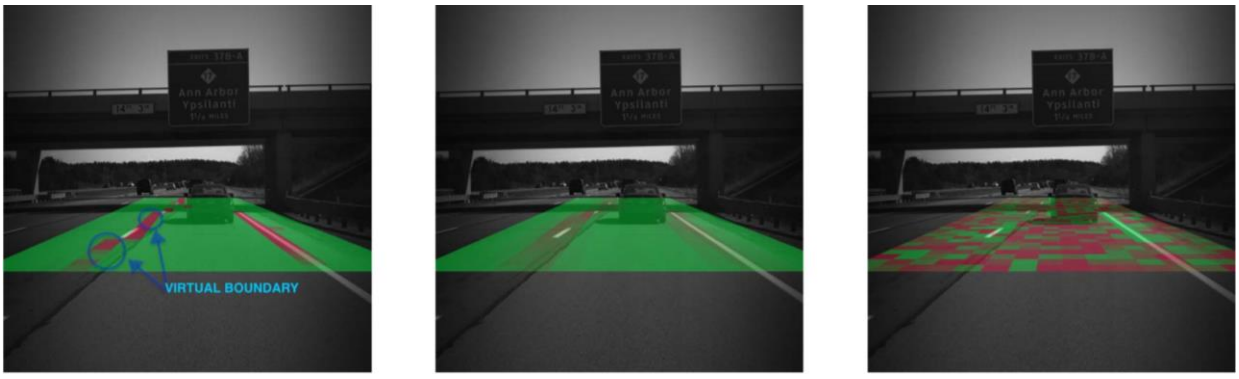
En el diseño del LSTM se emplearon sesenta y cuatro celdas, donde cada una contiene cuatro neuronas correspondientes a la memoria interna y las compuertas de entrada, salida y de olvido.

Para una mayor comprensión del tema con las respectivas ecuaciones de cada compuerta se puede recurrir al libro escrito por (Graves, 2012).

Por último, la última capa es un clasificador basado en una red neuronal estándar prealimentada que determinara si los estados de las LSTM corresponden o no a una marca del carril.

Los resultados comparativos visuales de la aplicación de una RNN, CNN y SVM en la detección de marcas del carril se pueden observar en la Figura 39 respectivamente de izquierda a derecha, donde los colores rojos indican que existe una alta probabilidad de que bajo una cierta región se encuentra una marca del carril y lo contrario el color verde. Cabe destacar que la CNN de esta prueba fue obtenida reemplazando las neuronas recurrentes por una red prealimentada estándar.

Figura 39. Comparativa entre una RNN, CNN Y SVM

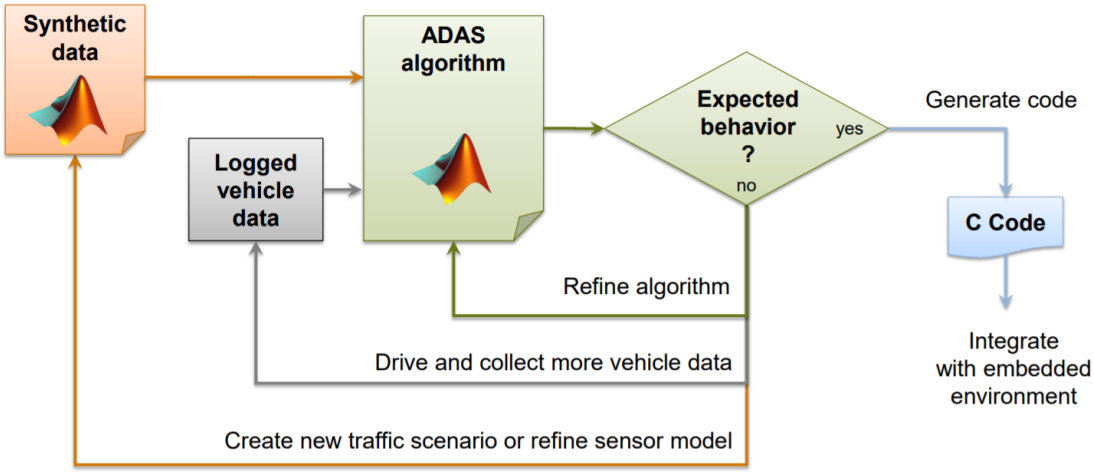


Fuente: Li et al. (2016).

3.3 Flujo de trabajo utilizado en la industria

Para poder ejemplificar uno de los flujos de trabajo utilizados en la industria para el desarrollo de un algoritmo dentro de los sistemas ADAS, que en un caso particular podría ser detectar carriles para un sistema LKAS o LDWS, en la Figura 40 se muestra el flujo de trabajo propuesto en (The MathWorks, Inc., 2016).

Figura 40. Flujo de trabajo del desarrollo de un algoritmo para ADAS



Fuente: The MathWorks, Inc. (2016).

Para entender el contexto de la Figura 40 en el trabajo se puede observar que, los algoritmos de visión computacional explicados hasta el momento estarían situados dentro de la caja ADAS *algorithm*, en la cual también estarían algoritmos de fusión de sensores en el caso por ejemplo de tener una cámara y un radar, como también los algoritmos de tomada de decisión.

El algoritmo de ADAS en desarrollo sería proporcionado por dos distintos tipos de datos, uno sintético, es decir, creado artificialmente y otro almacenado mientras el vehículo se encuentra en

movimiento. Si su comportamiento cumple con los requerimientos de diseño, entonces el siguiente paso sería generar un código en C e integrarlo a un sistema embebido, sino el proceso se repetiría mejorando así el algoritmo hasta alcanzar los requerimientos deseados.

3.4 Criterios de evaluación de los sistemas de detección de carriles

Según Xing et al. (2018) existen tres características básicas que un sistema de detección de carril debe de cumplir, los cuales son, exactitud, robustez y eficiencia. En otras palabras, el sistema tendrá que cumplir con los requerimientos de ejecución segura en tiempo de real, bajo error en la estimación de la posición de las líneas del carril y que esto sea ejecutado con un bajo coste computacional.

La evaluación de estos sistemas se ha dividido en dos: *offline* y *online*. La evaluación *offline* consiste en distintas métricas que son medidas de los resultados de la detección en imágenes o secuencia de imágenes. Esto normalmente es llevado a cabo en conjunto de datos privados o públicos tales como KITTI, TuSimple, entre otros. En cambio, la evaluación *online* es llevada cabo en tiempo real con el objetivo de analizar el grado de coincidencia de la detección del sistema con otro sistema con distintos sensores tales como el sistema de posicionamiento global (GPS, por sus siglas en inglés), LiDAR y modelos de los carriles de mucha exactitud.

Para la evaluación *offline* se han propuesto distintas métricas. De acuerdo con Kumar y Simon (2015) las más usuales son las siguientes: *Precision*, *Recall*, *F-score*, *Accuracy*, curvas *Receiver Operating Characteristic* (ROC, por sus siglas en inglés) y *Dice Similarity Coefficient* (DSC, por sus siglas en inglés). Lo medido en estas métricas es el resultado de la comparación entre las

detecciones y el conjunto de datos verdaderos del entorno (*Ground Truth*), con el fin de determinar cuando existe un verdadero positivo (VP), un falso positivo (FP), un verdadero negativo (VN), o un falso negativo (FN).

Un verdadero positivo se da cuando existe una marca del carril y el algoritmo la detecta. Un falso positivo ocurre cuando el algoritmo detecta una marca de carril donde no la hay. Un falso negativo se da cuando existe una marca del carril, pero el algoritmo no lo detecta y por último un verdadero negativo ocurre cuando el algoritmo no detecta la marca del carril y esta no existe.

En este trabajo se utilizará como métrica de los algoritmos propuestos la exactitud (*Accuracy*), que intuitivamente indica que tan bien se están detectando las marcas del carril excluyendo los verdaderos negativos. Su respectiva ecuación se muestra a continuación.

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN} \tag{3.11}$$

También se usarán las proporciones de falsos negativos y falsos positivos (FNR Y FPR, por sus siglas en inglés).

$$FNR = \frac{FN}{VP + FN} \tag{3.12}$$

$$FPR = \frac{FP}{FP + VN} \tag{3.13}$$

Para obtener una excelente detección de carriles los valores del FNR y el FPR deben estar cercanos lo más posible al 0% y el *Accuracy* cercano al 100%.

Cabe resaltar que estas métricas aún no representan un marco de referencia o un estándar donde todo el sistema de detección de carril es evaluado y con estas poder comparar entre distintos sistemas.

Capítulo 4

Simuladores

La simulación ha permitido el desarrollo, entrenamiento y evaluación de los sistemas autónomos, puesto que sin ello el análisis de la respuesta del sistema a distintas situaciones de conducción entre ellos casos extremos o desafiantes para un sistema autónomo, sería muy costoso y con problemas de seguridad para probarlos. También surge como una solución potencial en cuanto a recolección de más datos para el entrenamiento de estos sistemas, mejorando así la robustez ante nuevas situaciones (Amini et al., 2020).

En este capítulo se mostrarán el estado del arte de los simuladores de vehículos autónomos de código abierto. Para poder seleccionar uno de ellos, el cual coincida más con el alcance de este trabajo, se han tomado en cuenta los siguientes criterios:

- Escenarios con los cuales cuenta el simulador (autopistas, calles urbanas, otros vehículos, pedestres, etc.).
- Sensores con los cuales cuenta el simulador
- Propósito de desarrollo del simulador.
- Control del simulador
- Estado de desarrollo del simulador
- Lenguaje de programación utilizado
- Documentación y soporte de la comunidad
- Instalación
- Requerimientos computacionales

A continuación, se mostrarán las características de dos simuladores AirSim y CARLA.

4.1 AirSim

Es un simulador desarrollado por Microsoft AI & Research. En él se pueden simular sistemas autónomos como drones y vehículos. Los modelos construidos en él son de alta fidelidad tanto para el ambiente externo como para los sistemas del vehículo autónomo. Está construido sobre el motor gráfico Unreal y un reciente desarrollo sobre Unity. Soporta *hardware in the loop* (HIL) con controladores de vuelo populares (Kapoor, Shah, Dey, & Lovett, 2017).

- Escenario: cuenta con escenarios 3D provistos por Unreal y Unity, donde en algunos es posible tener a otros vehículos, pedestres, señales de tránsito, etc. Proveen algunos escenarios gratuitos.
- Sensores: Cámaras (RGB, *Infrared* (IR), profundidad y segmentación semántica), *Inertial Measurement Unit* (IMU), LiDAR, GPS, barómetro, sensor de distancia y magnetómetro
- Propósito: Simulador de vehículos autónomos, drones, entre otros.
- Control de la simulación: es posible modificar la velocidad de la simulación, cambiar el clima y tener el control sobre los otros vehículos a través de su interfaz de programación de aplicaciones (API, por sus siglas en inglés), entre otros.
- Estado de desarrollo del simulador: en desarrollo. Luego de una pausa entre comienzos del 2019 y 2020.
- Lenguaje de programación utilizado: C++, Python, Java. Es compatible con el sistema operativo para robots (ROS, por sus siglas en inglés).

- Documentación y soporte de la comunidad: documentación detallada disponible y comunidad activa.
- Instalación: Compatible con Windows y Linux.
- Requerimientos computacionales: Requerimientos mínimos de 64GB de RAM, GPU Nvidia discreta, SSD y pantallas 4K para una notebook.

4.2 CARLA

Es un simulador para el desarrollo, entrenamiento y validación de sistemas autónomos de conducción. Fue desarrollado por Intel Labs, Computer Vision Center Barcelona y Toyota Research Institute (Dosovitskiy, Ros, Codevilla, Antonio, & Koltun, 2017).

- Escenario: cuenta con siete modelos 3D de ciudades y sus correspondientes carriles definidos, están basados en el motor gráfico Unreal. Estos son proporcionados gratuitamente.
- Sensores: cámaras (RGB, profundidad y segmentación semántica), detector de colisión, *Global Navigation Satellite System* (GNSS), IMU, LiDAR, invasión de carril, detector de obstáculos y radar.
- Propósito: Simulador de sistemas de conducción autónoma.
- Control del simulador: es posible modificar la velocidad de la simulación, cambiar el clima, controlar a los pedestres a través de algoritmos de inteligencia artificial, control de la calidad de los gráficos afectando los fotogramas por segundo, entre otros.
- Estado de desarrollo del simulador: en desarrollo.
- Lenguaje de programación utilizado: Python. Compatible con ROS.

- Documentación y soporte de la comunidad: documentación detallada disponible y comunidad activa.
- Instalación: Compatible con Windows y Linux.
- Requerimientos computacionales: GPU discreta de 4GB, 30GB de espacio en disco.

4.3 Comparación entre simuladores y elección

Examinando los criterios citados anteriormente de cada uno de los simuladores se encuentran distintas similitudes y los grandes beneficios que ofrece cada uno, como por ejemplo la posibilidad de simular drones autónomos, el soporte para HIL en el caso de AirSim y poder controlar vía algoritmos de inteligencia artificial a los pedestres en el caso de CARLA.

Para la elección del simulador han tomado gran preponderancia los criterios de escenarios y requerimientos computacionales.

Con respecto a los escenarios, AirSim provee algunos de manera gratuita, sin embargo, al momento de la realización de este trabajo, no proveían gratuitamente escenarios con los distintos tipos de rutas, como autopistas, calles urbanas, etc. En cambio, CARLA como ya citado anteriormente provee de manera gratuita siete entornos con los distintos tipos de ruta requeridas para poner a prueba los algoritmos desarrollados.

Los requerimientos computacionales de AirSim son altos en comparación con los requerimientos medios de CARLA y la disponibilidad computacional al momento de este trabajo son de 8GB de RAM, una GPU Nvidia 1050Ti de 4GB y +500GB de disco.

En base a lo dicho anteriormente se ha elegido CARLA como el simulador para la implementación y prueba de los algoritmos desarrollados en este trabajo final de grado.

Marco metodológico

Capítulo 5

Diseño metodológico

5.1 Contexto de la investigación

El presente trabajo de investigación se enmarca en el estudio realizado para obtener el título de grado de Ingeniero Electromecánico con Orientación Electrónica.

Esta investigación forma parte de la continuación de la línea de investigación abierta por el equipo de robótica A2G (Autonomous Arandu Group), con el cual se ha participado en la primera competencia de vehículos autónomos a escala del Brasil RobocarRace (2018) y en el año 2019 con la nueva denominación Aguara'i (debido al vehículo a escala llamado por el mismo nombre), en la mayor competencia de robótica de Latinoamérica Winter Challenge Concórdia. En amaba se ha participado en representación de la Universidad Católica Nuestra Señora de la Asunción, Campus Alto Paraná, Facultad de Ciencias y Tecnología.

Esta investigación busca agregar una nueva capacidad de percepción visual a la línea de investigación ya establecida por los distintos integrantes que han pasado por el equipo A2G, ahora Aguara`i.

5.2 Alcance de la investigación

El alcance de esta investigación es explicativo, puesto que primeramente proporciona y especifica características de todo el sistema de detección de carriles, desde el nivel de preprocesamiento, modelado y seguimiento acompañado de predicciones. Seguidamente también brinda las respuestas al por qué cuando se manipulan ciertas variables de los algoritmos produce un resultado deseado o indeseado y cómo se puede seleccionarlos para obtener una detección de carriles lo más óptima posible de acuerdo con las limitaciones de cada algoritmo.

Según Sampieri, Collado, Lucio, Valencia y Torres (2014), los estudios explicativos contienen a los estudios descriptivos que tienen como finalidad especificar propiedades y características de ciertos fenómenos, es decir, su objetivo es la recolección de estas, además es mejor estructurado y agrega los cuestionamientos tales como, por qué ocurre un cierto fenómeno y en qué condiciones se manifiesta.

5.3 Diseño de la investigación

El diseño es del tipo experimental, puesto que como mencionan Sampieri et al. (2014) en su libro, los diseños experimentales consisten en la manipulación intencional de una o más variables independientes (causas antecedentes) para poder analizar las consecuencias de esta manipulación sobre una o más variables independientes (efectos consecuentes).

En el presente trabajo se manipulan intencionalmente las variables que intervienen tanto en los algoritmos de preprocesamiento de imágenes, segmentación y modelado con el objetivo de analizar los resultados en la correcta detección de carriles.

5.4 Enfoque de la investigación

El enfoque de una investigación es del tipo cuantitativo siempre y cuando exista un proceso secuencial y probatorio en donde se utiliza cálculos matemáticos y un análisis estadístico para probar ciertas hipótesis (Sampieri et al., 2014).

De lo dicho anteriormente se concluye que el presente trabajo de investigación es del tipo cuantitativo, puesto que existe un proceso secuencial a ser seguido de recolección de un conjunto de datos (imágenes), pruebas de las hipótesis en base a cálculos matemáticos para determinar la exactitud y eficacia de la detección de carriles.

5.5 Unidad de estudio

La unidad de estudio son las imágenes provenientes de la cámara. Con estas es posible percibir las características del entorno, especialmente características asociadas a las marcas del carril.

5.6 Técnicas e instrumentos de recolección de datos

Las técnicas e instrumentos de recolección de datos fueron tres: se ha recabado un conjunto de imágenes de TuSimple (imágenes de autopistas), se ha recabado los datos de la cámara del simulador CARLA y por último se ha recabado imágenes con la cámara de un celular, marca iPhone - modelo 6S, los carriles de la supercarretera entre la ciudad de Hernandarias y Ciudad del Este, Paraguay, y los carriles de la avenida gastronómica del Barrio Las Américas.

Capítulo 6

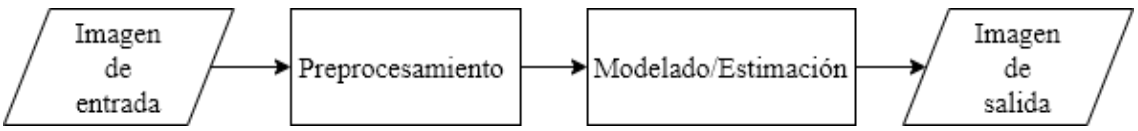
Diseño e implementación

Se ha elegido diseñar e implementar dos algoritmos de detección de carriles basados en procesamiento de imágenes. Ambos constan de dos grandes etapas: preprocesamiento y modelado/estimación. La elección de estos algoritmos fue hecha con el objetivo de modelar al carril de forma lineal y robusta antes ocasionales ruidos en la imagen.

Los algoritmos fueron diseñados y evaluados con el lenguaje de programación Python versión 3.6 en conjunto con las librerías OpenCV, Numpy, Image Processing SciKit, Matplotlib, entre otras. La implementación del control de un vehículo mediante la detección de carriles fue hecha en el simulador CARLA versión 0.9.5.

A continuación, en la Figura 41 se muestra el enfoque que tendrán los dos algoritmos seleccionados. Se realizó un diseño modular de tal manera a tener una distinción clara de las etapas del algoritmo y poder realizar cambios o añadir nuevas características con el menor esfuerzo posible.

Figura 41. Etapas de los algoritmos de detección de carriles propuestos



Fuente: Elaboración propia.

Cabe resaltar que el primer algoritmo es una implementación del código desarrollado por (Desegur, 2018). En cambio, el segundo fue diseñado por el autor de este trabajo, basándose en el código anterior implementado, no obstante, difiere en la etapa de preprocesamiento y modelado. Estas diferencias se introducen a continuación y en el contenido de las secciones siguientes.

En la etapa de preprocesamiento del segundo algoritmo se ha optado por trabajar con una imagen menor y con un ROI diferente por lo cual se ha podido aumentar la cantidad de líneas detectadas en el PPHT sin perder un desempeño aceptable de ejecución, como también se han realizado cambios en la detección de bordes, aumentando la robustez ante cambios de contraste de la imagen y en la separación de las marcas del carril.

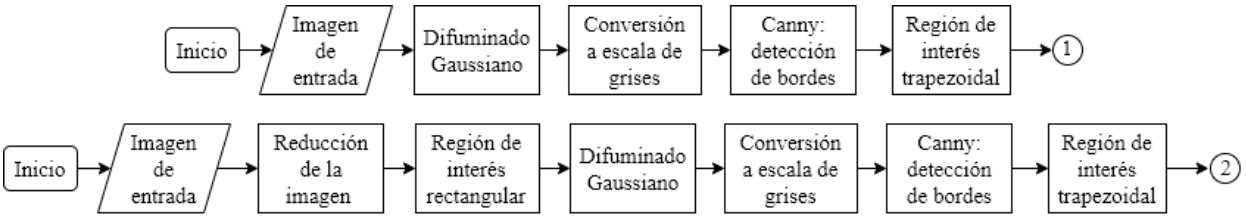
En la etapa de modelado/estimación ambos comparten un mismo modelo lineal, sin embargo, para llegar a este modelo lineal, en el primer algoritmo se hace uso de la Transformada Probabilística Progresiva de Hough (PPHT) en conjunto con una regresión lineal simple y en el segundo algoritmo con el objetivo de obtener una detección más robusta ante *outliers*, se ha agregado el algoritmo RANSAC que finaliza igualmente con una regresión lineal simple con un ajuste por mínimos cuadrados.

La elección de ambos algoritmos fue basada en el artículo escrito por (Xing et al., 2018). Donde la transformada de Hough está presente en algoritmos de menor complejidad y el RANSAC se encuentra en seis de los nueve algoritmos presentados basados en procesamientos de imágenes.

6.1 Preprocesamiento

Esta etapa del algoritmo será la encargada del realce de bordes y suavizado de ruidos, como también la elección de la región en la cual se estimará la presencia de los carriles, entiéndase ruido por bordes que no corresponden a las marcas del carril. Los pasos que se seguirán en esta etapa son visualizados en la Figura 42. Donde el diagrama superior representa al primer algoritmo y el inferior al segundo algoritmo.

Figura 42. Preprocesamiento



Fuente: Elaboración propia.

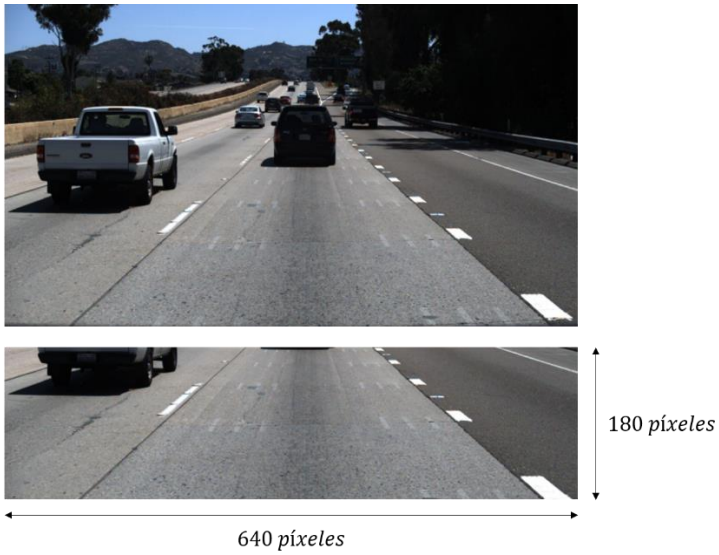
Para ambos algoritmos se trabajará con una imagen de entrada en el espacio de color RGB con resolución espacial de $1280x720$ píxeles. A seguir se expondrá cada proceso de ambos preprocesamientos con sus características particulares y similitudes.

6.1.1 Reducción de la imagen y región de interés.

Para aumentar el desempeño del segundo algoritmo se redujo la resolución espacial de la imagen de entrada a $640x360$ píxeles, disminuyendo así la cantidad de tiempo de ejecución del algoritmo.

Debido a que en la mayoría de los casos los carriles visiblemente están posicionados en la zona frontal del vehículo se definió una nueva imagen, a través de una región de interés rectangular determinada como la mitad de la imagen reducida, para las posteriores etapas del preprocesamiento de este algoritmo. Como resultado se obtiene la reducción del cálculo de 921.600 a 115.200 píxeles, con una resolución espacial de $640x180$ píxeles. Visualmente lo anterior puede ser observado en la Figura 43.

Figura 43. Reducción de la imagen de entrada



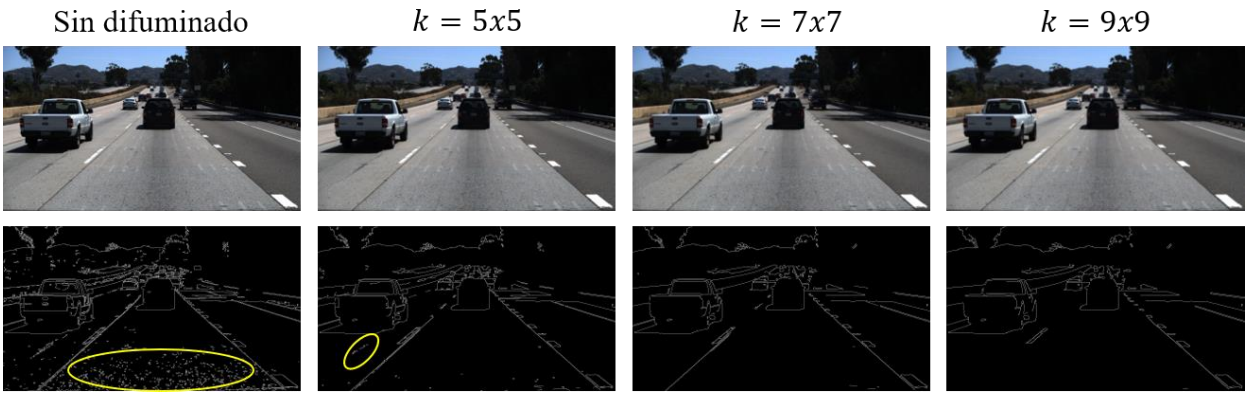
Fuente: Elaboración propia.

6.1.2 Difuminado gaussiano.

El difuminado gaussiano realiza un suavizado a la imagen dependiendo del tamaño del *kernel* y de las desviaciones estándar en la dirección x e y. Si no se definen las desviaciones estándar, son calculadas a partir del tamaño del *kernel*, lo cual es lo utilizado.

En la Figura 44 se observa de izquierda a derecha la aplicación de distintos tamaños del *kernel*. La primera columna de la izquierda consiste en la imagen original y su correspondiente detección de bordes, lo cual será analizado a seguir. Se puede evidenciar que el mejor tamaño del *kernel* y el utilizado es el de $7x7$, debido a que filtra la mayor parte del ruido, pero aún mantiene las marcas del carril. Los ruidos son contorneados en amarillo para una mejor apreciación.

Figura 44. Difuminado gaussiano a diferentes niveles



Fuente: Elaboración propia.

6.1.3 Conversión a escala de grises.

La conversión a escala de grises permitirá el análisis de las razones de cambio de las intensidades entre un píxel que forma parte de una marca de carril y otro que no, como también la disminución del poder de cómputo necesario para analizar toda la imagen.

La elección del algoritmo de conversión a escala de grises es el $Luminance$. Esta elección fue basada en la literatura ya expuesta anteriormente en la sección 2.3.1. El resultado de aplicar este algoritmo se observa en la Figura 45.

Figura 45. Conversión a escala de grises con *Luminance*

Fuente: Elaboración propia.

6.1.4 Canny: detección de bordes.

El algoritmo de Canny posee dos umbrales que deben serle proporcionados para poder determinar cuáles píxeles forman parte de un borde o no. Ambos normalmente son definidos empíricamente durante las pruebas con el conjunto de imágenes de cada caso específico. Debido a que en la detección de carriles las imágenes percibidas están expuestas a cambios de luminosidad, de contraste, se ha optado por determinar automáticamente para cada imagen ambos umbrales.

Para el primer algoritmo el valor de ambos umbrales se lleva a cabo calculando la mediana M_e de las intensidades de la imagen en escala de grises. Por consiguiente, los umbrales $minVal$ y $maxVal$ se definen como sigue.

$$minVal = 0.67M_e \tag{6.1}$$

$$maxVal = 1.33M_e \tag{6.2}$$

El resultado de aplicar el algoritmo de Canny con estos umbrales se muestra en la imagen inferior de la izquierda de la Figura 43.

Para el segundo algoritmo en cambio, ambos umbrales son determinados de acuerdo con el cálculo de umbralización propuesto por Otsu, explicado en la sección 2.3.3. La decisión de optar por el cambio estuvo basada en la observación, realizada por el autor de este proyecto del desempeño del método anterior, donde se constató que no logra detectar las marcas del carril en escenarios de poco contraste.

A raíz de lo anterior se ha buscado un nuevo método más robusto para determinar estos umbrales, por lo que en base a investigaciones se ha llegado al artículo de Fang, Yue, y Yu (2009), donde se demuestra la efectividad del método de Otsu en conjunto con Canny. Siguiendo las recomendaciones de los autores se han establecido los valores de los umbrales de Canny como sigue.

$$minVal = 0{,}5(Umbral\ de\ Otsu) \tag{6.3}$$

$$maxVal = Umbral\ de\ Otsu \tag{6.4}$$

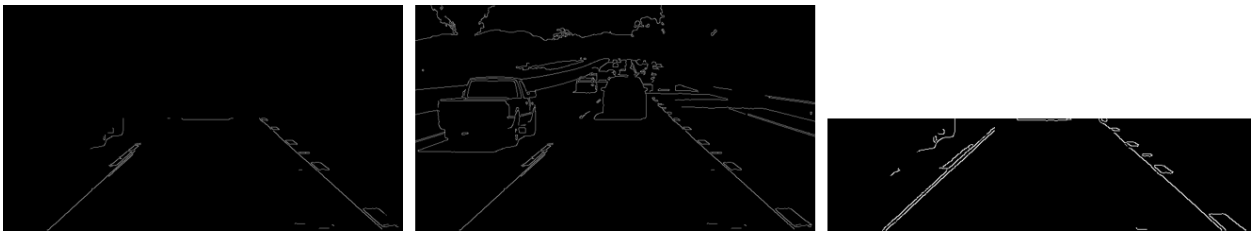
6.1.5 Región de interés.

Debido a que la cámara siempre estará montada en un mismo sitio, se puede asumir que las marcas del carril en la imagen estarán en la mayor parte del tiempo en una cierta región, por tanto, la salida de esta última etapa de preprocesamiento son únicamente los bordes que se encuentren dentro de esta región.

Se hace uso de un ROI estático tipo trapezoidal definido manualmente por cuatro coordenadas. Las mismas son definidas en base al peor caso que es en un carril curvo. A continuación, en la

Figura 46 se puede observar la imagen resultante después de la detección de bordes en el centro, su correspondiente ROI del primer algoritmo en a la izquierda y a la derecha la imagen resultante luego de la aplicación del ROI a la detección de bordes del segundo algoritmo.

Figura 46. Región de interés



Fuente: Elaboración propia.

6.2 Modelado

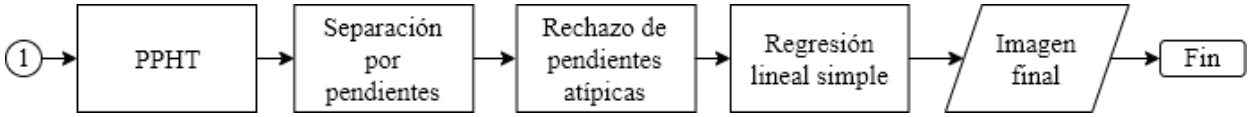
Se ha optado modelar linealmente a los carriles, puesto que en la región más cercana al vehículo las marcas del carril en la mayor parte del tiempo se pueden aproximar satisfactoriamente por una línea recta. Esta sección tendrá dos enfoques donde cada uno corresponde a los dos algoritmos desarrollados respectivamente. La diferencia entre ambos consiste en cómo se llegan a los parámetros más adecuados para el modelo lineal.

6.2.1 PPHT en conjunto con una regresión lineal simple.

Este enfoque consiste en la aplicación de algoritmo PPHT a la imagen binaria obtenida de la etapa de preprocesamiento, seguidamente de una separación entre las rectas pertenecientes a las marcas del carril izquierdo y derecho, rechazo de rectas con pendientes atípicas y por último una

regresión lineal simple de los puntos que determinan a las rectas que han quedado. Esto se puede visualizar en la Figura 47.

Figura 47. Modelado con PPHT en conjunto con regresión lineal simple



Fuente: Elaboración propia.

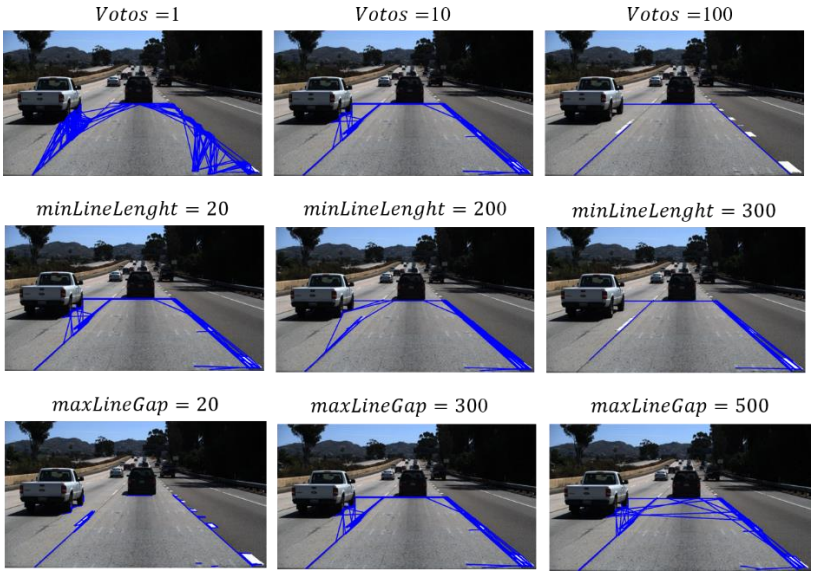
6.2.1.1 Transformada probabilística progresiva de Hough.

Este algoritmo tiene cinco hiperparámetros (por hiperparámetros se comprende todos aquellos valores definidos por el diseñador a la hora de utilizar un algoritmo).

Los dos primeros tienen que ver con la resolución del acumulador, es decir de ρ y θ. Los valores elegidos fueron de un píxel de resolución para ρ y un grado de resolución para θ. Estos valores fueron elegidos en base a lo utilizado por (Matas et al., 2000).

Los siguientes tres corresponden al umbral de $Votos$, la mínima longitud de la línea ($minLineLenght$) y la máxima separación entre dos puntos para que sean considerados pertenecientes a una misma línea ($maxLineGap$). Para tener una noción del comportamiento del algoritmo al variar estos valores, se ha elaborado la Figura 48.

Figura 48. Hiperparámetros de Hough



Fuente: Elaboración propia

Estos 3 hiperparámetros fueron definidos empíricamente en base a una evaluación visual luego de repetidas iteraciones sobre el conjunto de imágenes de entrenamiento con los siguientes valores: $Votos = 20,\ minLineLenght = 100$ y $maxLineGap = 200$.

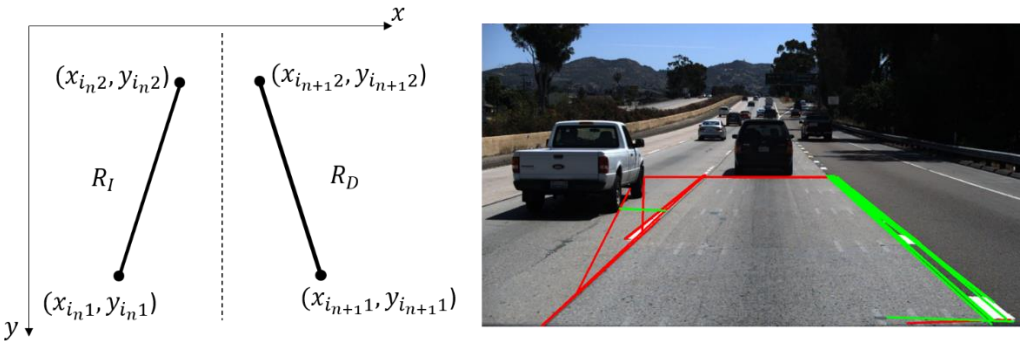
6.2.1.2 Separación por pendientes.

Para poder determinar cuáles segmentos de rectas podrían pertenecer a las marcas del carril izquierdo y derecho, se utilizan los puntos extremos proveídos como salida del PPHT de cada segmento de recta y se calculan sus pendientes. El formato de los segmentos de recta en la imagen luego del PPHT puede ser observado en la Figura 49, como también el resultado de aplicar esta separación. El cálculo de la pendiente se realiza con la siguiente ecuación.

$$m_i = \frac{y_{i_n 2} - y_{i_n 1}}{x_{i_n 2} - x_{i_n 1}} \tag{6.5}$$

Si la pendiente es negativa, el segmento de recta será considerado perteneciente a las posibles marcas del carril izquierdo, sino si es positiva, será considerado perteneciente a las posibles marcas del carril derecho. Segmentos de recta verticales no son considerados.

Figura 49. Formato de los segmentos de recta en la imagen y resultado de esta separación



Fuente: Elaboración propia

6.2.1.3 Rechazo de pendientes atípicas.

Después de esta separación puede ser observado segmentos de rectas con pendientes prácticamente nulas y/o pendientes con valores divergentes. Para poder filtrar estos valores atípicos de pendientes se han establecido dos umbrales k_{max}, k_{min} para cada lado de la marca del carril y un valor T que determina el máximo rango de variación respecto al valor promedio de pendientes presentes, que puede tener la pendiente de un segmento de recta cualquiera. Por tanto, sólo se mantendrán las pendientes m_i que cumplan con las siguientes inecuaciones.

$$m_{if} = \{m_i \mid k_{min} \leq m_i \leq k_{max}\} \tag{6.6}$$

$$\overline{m} = \frac{1}{N}\sum_{i}^{N} m_{if} \tag{6.7}$$

$$m_{final} = \{m_{if} \mid \overline{m} - T \leq m_{if} \leq \overline{m} + T\} \tag{6.8}$$

Para el conjunto de imágenes de entrenamiento fueron definidos empíricamente los siguientes valores para cada uno de los umbrales de ambos carriles: $T = 0.08$, para el carril izquierdo $k_{max} = -0.4$ y $k_{min} = -2.14$, y para el carril derecho $k_{max} = 2.2$ y $k_{min} = 0.5$. En la imagen de la derecha la Figura 50 se observa el resultado de aplicar lo anterior a los segmentos de recta detectados en la imagen de la izquierda.

Figura 50. Resultado del rechazo de pendientes atípicas

Fuente: Elaboración propia

6.2.1.4 Regresión lineal simple y resultado final.

Finalmente, para estimar el modelo lineal de cada lado de las marcas del carril, se aplica una regresión lineal simple con un ajuste por mínimos cuadrados, al conjunto de puntos que determinan los segmentos de rectas restantes. El resultado final de la ejecución del algoritmo en distintas imágenes se puede observar en la Figura 51, donde en ciertas imágenes se comprueba la eficacia

del algoritmo implementado, como también situaciones donde no logra detectar correctamente el carril.

Figura 51. Resultado final del algoritmo PPHT en conjunto con una regresión lineal simple

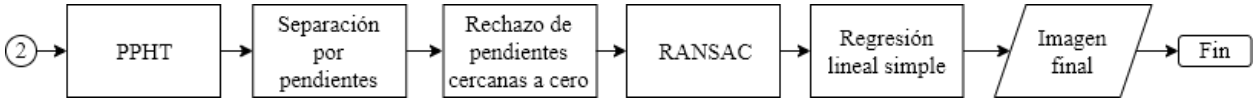
Fuente: Elaboración propia

6.2.2 PPHT en conjunto con RANSAC.

Este algoritmo tiene como objetivo aumentar la robustez ante *outliers* presentes en el conjunto de puntos para posterior modelado y de cierta manera evitar que ciertos hiperparámetros definidos empíricamente no permitan la flexibilidad del algoritmo a la hora detectar carriles, como ejemplo, en curvas que sus marcas del carril presentan una pendiente que no se encuentra dentro de los umbrales establecidos en el enfoque anterior.

Para ello se agrega un nuevo algoritmo el cual es el RANSAC y se modifica las etapas de separación por pendientes y rechazo de pendientes atípicas. El nuevo enfoque resultante es el mostrado en la Figura 52.

Figura 52. Modelado con PPHT en conjunto con RANSAC y regresión lineal simple



Fuente: Elaboración propia

6.2.2.1 Transformada probabilística progresiva de Hough.

De la misma manera que en el algoritmo anterior, se definen las resoluciones del acumulador de un píxel para ρ y un grado para θ.

De igual manera que en el primer algoritmo, se ha definido empíricamente los $Votos$, $minLineLenght$ y $maxLineGap$, sin embargo, los valores seleccionados son menores debido a dos motivos, primeramente la imagen es menor, por lo tanto, las líneas en consecuencia tienen una longitud menor, y segundo, que al tomar valores menores se aumentan la probabilidad de detectar los carriles, pero en consecuencia se adiciona más ruido a la imagen (entiéndase segmentos de línea que no constituyen una marca del carril). Los hiperparámetros seleccionados son los siguientes: $Votos = 10,\ minLineLenght = 10$ y $maxLineGap = 100$.

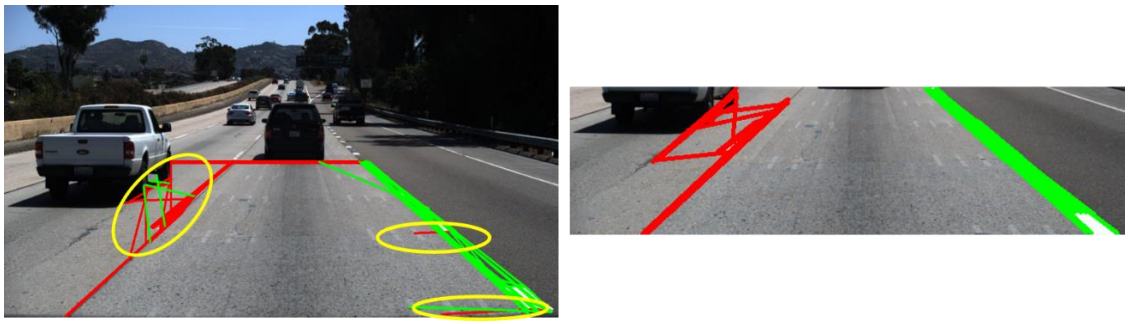
6.2.2.2 Separación por pendiente y cuadrante.

La separación de las marcas del carril para ambos lados, izquierdo y derecho, se da de manera parecida a lo explicado en la sección 6.2.1.2. La única diferencia consiste en agregar una condición más que el signo de la pendiente, la cual es, en cuál región se encuentra el segmento de recta.

Para ello se divide la imagen en dos partes iguales R_I, R_D en la dirección x, por tanto, si el punto medio de un segmento de recta se encuentra en la región de la izquierda y tiene una pendiente negativa, será considerado como un posible segmento de recta perteneciente a la marca del carril izquierdo. La misma lógica se aplica para las marcas del carril derecho, intercambiando región izquierda por derecha y pendiente negativa por positiva. Una noción gráfica de lo anterior puede obtenerse en la imagen de la izquierda en la Figura 49.

A modo de comparación con la separación anterior presentada, se pueden observar la ejecución de ambas sobre la misma imagen y sus respectivas clasificaciones en la Figura 53. A la izquierda el enfoque basado sólo en el signo de la pendiente y a la derecha considerando tanto el signo de la pendiente y la ubicación del segmento de recta.

Figura 53. Comparación entre métodos de separación de marcas del carril



Fuente: Elaboración propia

6.2.2.3 Rechazo de pendientes cercanas a cero.

A diferencia del método anterior de rechazo de segmentos de rectas con pendientes atípicas, este método sólo filtra aquellas con pendientes cercanas a cero, para ello se toma el valor absoluto

de la pendiente de cada segmento de recta y se compara con un umbral determinado empíricamente, lo cual resulta en el siguiente conjunto de pendientes asociados a segmentos de recta. Por lo anterior se reduce de cinco hiperparámetros a solamente uno en comparación con la etapa respectiva del sistema anterior implementado.

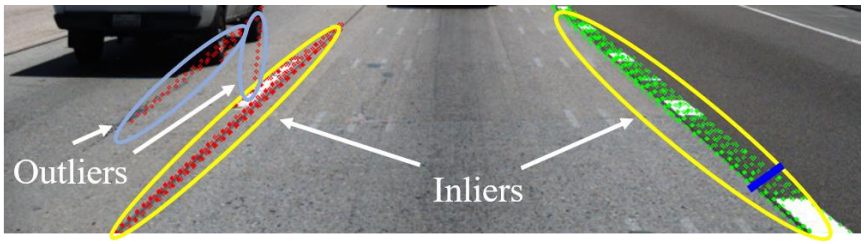
$$m_{final} = \{m_i \,||m_i| > 0.6\} \tag{6.9}$$

6.2.2.4 RANSAC.

Antes de introducir el conjunto de puntos obtenidos hasta el momento al RANSAC, se debe realizar una preparación de estos, que consiste en generar puntos intermedios de cada segmento de recta cada cinco píxeles a través de una interpolación. Esto se realiza con el objetivo que al momento de la ejecución del RANSAC, el algoritmo les dé mayor peso a los segmentos de recta más largos. Esta parte de la preparación fue basada en el código de (Bae, 2017).

Para la ejecución del algoritmo RANSAC, se deben proveer tres hiperparámetros: min_$samples$, $residual\_threshold$ y max _$trials$. Se define min_$samples = 2$ debido a que se pretende modelar una línea recta. Para la elección del valor de los dos restantes se debe observar el conjunto de puntos y estimarlos en consecuencia. El $residual\_threshold$ es el umbral t y max _$trials$ es el número k de iteraciones ambos explicados en la sección 2.4.2.

Figura 54. Conjunto de puntos interpolados



Fuente: Elaboración propia

En la Figura 54 se puede observar los puntos luego de la interpolación. Para poder determinar el $residual\_threshold$ se deben observar estos puntos y determinar el ancho aproximado de las marcas del carril, se ha trazado un segmento de color azul en la imagen en la esquina inferior derecha para ello.

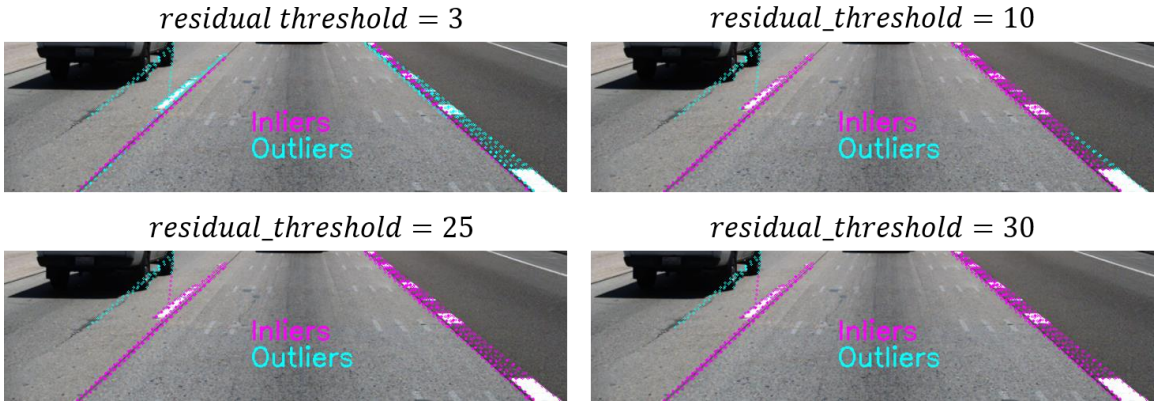
Este segmento tiene longitud de veintiséis píxeles, por tanto, luego de varias pruebas se estima un valor del $residual\_threshold = 10$, puesto que este hiperparámetro es el valor umbral con el cual se decide si un punto será considerado como un *inlier* o un *outlier* respecto al modelo creado, al calcular la distancia ortogonal del punto al modelo. En resumen, el mejor modelo será aquel que pase por el centro longitudinal de las marcas del carril, por ello serán considerados *inliers* los puntos a 10 píxeles de distancia del modelo.

El hiperparámetro max $\_trials$ de acuerdo con la ecuación (2.35) se determina definiendo: el porcentaje de error del modelo z, para el presente trabajo se toma un error del 1%, el número de muestras n que es igual a min$_samples$ y por último el porcentaje de *inliers* ω en el conjunto de puntos. Al observar el conjunto de puntos en la Figura 54, se puede estimar una cantidad de *inliers* alrededor del 80% del conjunto total, sin embargo, para obtener con un factor de seguridad debido

a los hiperparámetros bajos del PPHT se definió ω con un porcentaje del 50%. Aplicando lo anterior en la ecuación citada, resulta en max $\_trials = 16$.

La ejecución del RANSAC variando el hiperparámetro $residual\_threshold$ alrededor de la estimación hecha se puede visualizar en la Figura 55.

Figura 55. Variaciones de *inliers* y *outliers*



Fuente: Elaboración propia

6.2.2.5 Regresión lineal simple y resultado final.

Finalmente, al conjunto de puntos de *inliers* resultantes del RANSAC se aplica una regresión lineal simple con un ajuste por mínimos cuadrados, tanto para las marcas del carril izquierdo y derecho.

El resultado final de la ejecución del algoritmo en distintas imágenes se puede observar en la Figura 56, donde se puede visualizar la eficacia del algoritmo y el efecto de la mejora hecha, tanto en la etapa de preprocesamiento como en la etapa de modelado.

Figura 56. Resultado final del algoritmo PPHT en conjunto con RANSAC y una regresión lineal simple

Fuente: Elaboración propia.

6.3 Simulación

Para cumplir con el objetivo específico propuesto en el trabajo, de implementar un algoritmo de visión computacional para la detección de carriles en el ambiente de un vehículo autónomo se ha utilizado el simulador de vehículos autónomos CARLA, donde se cuenta con el realismo necesario para cumplir con el criterio del ambiente de un vehículo autónomo, y con la posibilidad de controlar la velocidad y dirección del vehículo. Por consiguiente, posibilita el cumplimiento de los requisitos inherentes por el objetivo específico citado.

El control de dirección implementado en el vehículo es un control de lazo cerrado directamente proporcional a la diferencia entre el centro del vehículo y el centro del carril, es decir, este intentará mantener al vehículo en el centro del carril. En cambio, el control de velocidad implementado es con un control Proporcional-Integral (PI) con un set point constante durante todo el recorrido. Cabe

destacar que este control PI no fue diseñado, simplemente fue implementado a partir de ejemplos proporcionados por los desarrolladores del simulador.

Para poder llevar a cabo la simulación, primeramente, se debe configurar la simulación y posterior a esto posibilitar la comunicación y el flujo de datos entre el cliente (códigos en Python) y el servidor (simulador CARLA). Lo escrito e implementado en esta sección están basados en la documentación del simulador (CARLA Team, s.f.) y en la tesis de maestría (Cattaruzza, 2019).

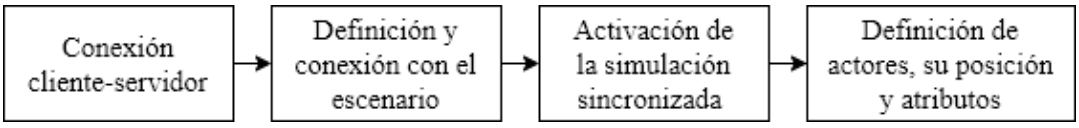
Una noción básica del proceso de simulación es la siguiente: CARLA está construido sobre una arquitectura cliente-servidor, entonces el servidor ejecuta la simulación y el cliente recupera o recibe la información con lo cual exige cambios en el servidor.

Tanto la configuración como la comunicación entre el cliente y el servidor son posibles gracias a las APIs escritas en Python.

6.3.1 Configuración de la simulación.

En la configuración de la simulación se establece la conexión entre el cliente y el servidor, se definen los parámetros de la simulación, los actores y sus características. Lo anterior puede ser dividido en 4 pasos a llevar a cabo que pueden ser visualizados en la Figura 57.

Figura 57. Configuración de la simulación



Fuente: Elaboración propia

El primer paso consiste en la conexión del cliente con el servidor, que se da identificando al cliente con la dirección IP (*Internet Protocol*) local y la comunicación vía dos puertos TCP (*Transmission Control Protocol*) normalmente 2000 y 2001. Al tener esta conexión lista se debe definir el escenario a utilizar, como se ha dicho el simulador posee 7 ciudades, para este trabajo se utilizó la ciudad 4 puesto que presenta una autopista acorde a los requerimientos del trabajo.

La simulación se puede ejecutar en dos modos: sincrónicas o asincrónicas. Se ha elegido la sincrónica, puesto que con esto se asegura al cliente al mando de la simulación y el no desbordamiento de información cuando el cliente se ejecute a una menor frecuencia que el servidor. Como también con esto se asegura que el servidor sólo envíe los datos medidos por cada sensor cuando estos estén totalmente listos para ser enviados.

En el simulador tanto los vehículos, peatones, sensores, entre otros, reciben el nombre de actores. Para acceder a ellos, como también modificar sus atributos, se utiliza la clase *ActorBlueprint* y sus métodos. Los actores utilizados fueron un vehículo Tesla *Model* 3 y una cámara RGB. Los atributos de la cámara establecidos fueron: resolución de la imagen de $1280x720$ y con un campo de visión (FoV, por sus siglas en inglés) de 110 grados.

El último paso por realizar es definir la posición de los actores. La posición inicial del vehículo fue definida según las coordenadas utilizadas en el simulador, para su posición $(x = 391.431, y = -62.8155, z = 1.2)$ y rotación $(pitch = 0,\ yaw = 90.4391,\ roll = 0)$. La posición relativa al vehículo de la cámara definida fue $(x = 1.5,\ y = 0,\ z = 2.4)$ y rotación $(pitch = 0,\ yaw = 0,\ roll = 0)$.

Para el control de velocidad, las constantes del control PI utilizadas fueron las establecidas por los desarrolladores del simulador, las cuales son: $K_P = 1.0\ ,\ K_I = 0.07,\ \Delta t = \frac{1}{30}$.

6.3.2 Comunicación cliente – servidor.

En la Figura 58 se observa el flujo diseñado e implementado para la simulación, que se ejecuta luego de realizar la configuración. Primeramente, la imagen suministrada en el formato BGRA (BGR de los colores primarios y A del canal *Alpha*) por la cámara del lado del servidor es almacenada en una cola, para su posterior paso al algoritmo de detección de carril, la cual es la última imagen que ha ingresado en la cola.

El algoritmo de detección de carriles procesa las imágenes, de acuerdo con cada algoritmo diseñado e implementado en este trabajo. El formato de la posición de las marcas del carril para cada marca del carril (izquierdo y derecho) es el siguiente: un array de cuatro elementos para el izquierdo $(x_{i1}\ \ y_{i1}\ \ x_{i2}\ \ y_{i2})$ y el derecho $(x_{d1}\ \ y_{d1}\ \ x_{d2}\ \ y_{d2})$ que determinan cada segmento de recta.

Para el cálculo del error lateral, es decir, la distancia que el centro geométrico del vehículo se encuentra del centro del carril, se asume que la cámara está situada en el centro geométrico del vehículo, por tanto, el error se calcula como sigue.

$$x_m = entero(ancho\ de\ la\ imagen/2) \tag{6.10}$$

$$x_{cdc} = entero(\frac{x_{i1} + x_{d1}}{2}) \tag{6.11}$$

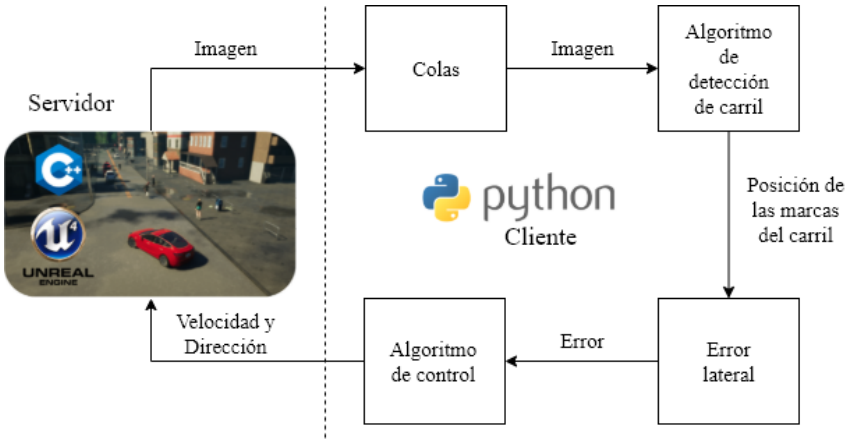
$$Error = x_m - x_{cdc} \tag{6.12}$$

Como se ha dicho el algoritmo de control de dirección es un control de lazo cerrado del tipo proporcional. Para una mayor estabilidad y control del vehículo se debe implementar un algoritmo de control clásico como el PID o un control predictivo basado en el modelo (MPC, por sus siglas en inglés) en esta etapa del sistema, sin embargo, no se ha realizado este diseño e implementación puesto que no forma parte de los objetivos de este proyecto.

Tanto la dirección como la velocidad pueden ser controladas a través de la API del simulador. Para el control de velocidad se fijó un set point constante igual a 80km/h, en cambio el set point de dirección se calcula proporcionalmente al error lateral. Se ha encontrado en base a pruebas que se obtiene la mayor suavidad en la dirección y una respuesta aceptable multiplicando este error por el valor $\frac{1}{720}$.

En caso de no ser posible la obtención del error debido a que el algoritmo de detección de carriles no ha obtenido de manera correcta las posiciones de las marcas del carril, la dirección enviada al servidor es la dirección enviada en la iteración anterior.

Figura 58. Diagrama de flujo de la simulación



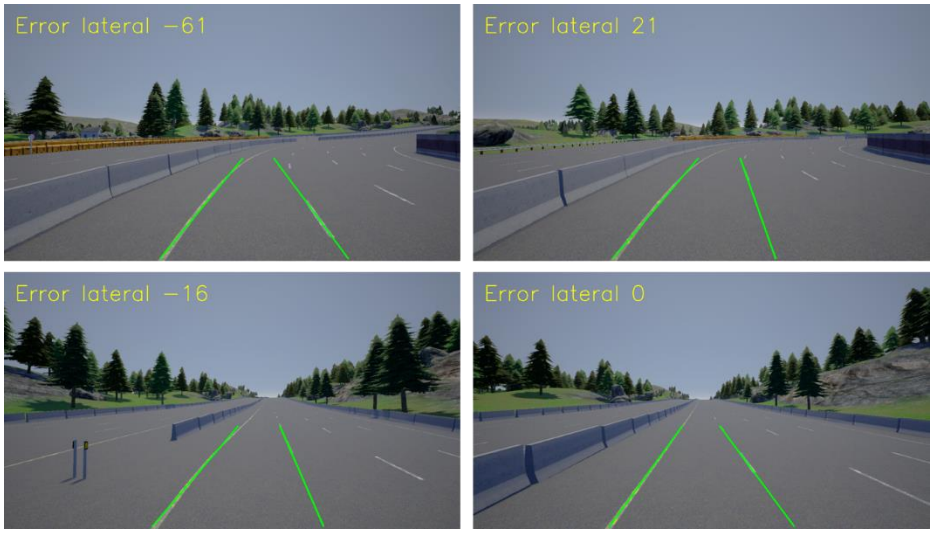
Fuente: Elaboración propia.

6.3.3 Ejecución de la simulación.

Para la ejecución de la simulación primeramente se ejecuta el servidor con un paso de tiempo fijo traducido del inglés *time step*, esto es debido a que, al tener un paso de tiempo variable y el modo sincronizado activado, es el cliente el que define el tiempo de la simulación, entonces podrían darse inconsistencias por ejemplo al esperar al cliente un tiempo mayor de ejecución a 0,1 segundos, al darse esta situación los cálculos físicos ya no son confiables. El paso de tiempo utilizado fue de $\frac{1}{30}$ segundos.

Seguidamente se ejecuta el cliente, es decir el código en Python desarrollado, con lo que se obtendrá la comunicación bidireccional de datos entre el cliente y el servidor, y una ventana donde se podrá observar al vehículo conducirse de forma autónoma, manteniendo su respectivo carril. El resultado se puede observar en la Figura 59.

Figura 59. Valores demostrativos del error lateral en el simulador



Fuente: Elaboración propia.

Capítulo 7

Pruebas y resultados

Para poder validar el desempeño de los algoritmos de forma cuantitativa, ambos algoritmos expuestos en el capítulo 6, fueron puestos a prueba con dos conjuntos de imágenes y como medio de percepción para la navegación autónoma de un vehículo. En las dos primeras se determina la exactitud de cada algoritmo en detectar los carriles y en la última se demuestra si es posible controlar un vehículo de forma autónoma a través de la detección de carriles.

Las especificaciones de la máquina que se utilizó para ejecutar estas pruebas son las siguientes:

- Procesador: Intel i7-8750H
- Memoria RAM: 8GB
- Tarjeta Gráfica dedicada: Nvidia GTX 1050Ti 4GB
- Sistema operativo: Windows 10 Home

7.1 Dataset TuSimple

El dataset TuSimple es un conjunto de imágenes para el entrenamiento y prueba de algoritmos de detección de carriles en autopistas, que fue lanzado por motivos de la competencia de detección de carriles propuesta por la empresa TuSimple, Inc. El dataset contiene 128.160 imágenes de $1280x720$ píxeles en el formato de videoclips de 20 fotogramas cada uno, de los cuales en el vigésimo fotograma de 3626 videoclips es anotada la posición de los carriles, generando así los datos verdaderos del entorno (traducción del inglés *Ground Truth*) (TuSimple, Inc., 2017).

Las imágenes del dataset fueron obtenidas en condiciones climáticas buenas y aceptables, en diferentes horarios del día y con diferentes situaciones del tráfico vehicular.

Los carriles anotados son cuatro, los directamente posicionados en la región frontal del vehículo y los carriles a su izquierda y derecha, no obstante, para esta evaluación solamente se consideraron las anotaciones que se encuentren dentro de la región de interés trapezoidal definida en el capítulo 6, es decir, la exactitud de los algoritmos se cuantificará al estimar correctamente los carriles en la región frontal próxima al vehículo.

Para determinar los hiperparámetros de los algoritmos explicados en el capítulo 6, se utilizaron las 1000 primeras imágenes de los videoclips pertenecientes a la carpeta especificada en el dataset como 530, cabe aclarar que estas imágenes no se encuentran dentro de los 3626 videoclips anotados, en otras palabras, las imágenes de configuración no se han utilizado en las pruebas de los algoritmos.

Para la evaluación primeramente se filtraron las anotaciones proveídas en el formato JSON. Estas anotaciones son descritas en formatos de coordenadas de puntos en el sistema de referencia de la imagen.

Por tanto, para filtrarlas se verificó si cada punto de una misma marca de carril se encuentra dentro del ROI y existen, puesto que originalmente las anotaciones incorporan en coordenadas donde no existen marcas del carril el número -2, luego de este filtro se adjunta este punto a un

nuevo archivo JSON siguiendo el mismo formato del JSON original, donde cada marca del carril es una lista de puntos. El proceso anterior puede ser observado en la Figura 60, donde las circunferencias verdes son guardadas al nuevo JSON, en cambio las de color rojo son descartadas.

Figura 60. Filtrado de las anotaciones de TuSimple y estimaciones

Fuente: Elaboración propia

Para la estimación de cada algoritmo se genera de igual manera un conjunto de puntos en el sistema de coordenadas de la imagen y son guardados a un archivo JSON, siguiendo el mismo formato de las anotaciones del dataset TuSimple. Nuevamente estas estimaciones son filtradas de igual manera a lo explicado en el párrafo precedente.

Para el cálculo del *Accuracy* en el dataset TuSimple se utiliza la ecuación (7.1) donde $c_{videoclip}$ es el número de puntos correctamente estimados y $s_{videoclip}$ es el número de puntos anotados en la imagen del videoclip. Un punto es considerado correctamente estimado, si el punto estimado se encuentra dentro de una cierta distancia definida por un umbral en términos de coordenadas, respecto a un punto anotado. El cálculo del umbral se da de acuerdo con la pendiente de cada marca del carril.

$$Accuracy = \frac{\sum_{videoclip} c_{videoclip}}{\sum_{videoclip} s_{videoclip}} \quad (7.1)$$

La definición de un falso positivo o negativo para las predicciones en este trabajo se da de la siguiente manera, una estimación dada por el algoritmo será considerada como un falso positivo cuando la marca del carril es estimada, pero la marca del carril no se encuentra en la posición donde se ha estimado y como un falso negativo cuando la marca del carril existe, pero ninguna estimación coincide con la anotación.

El término coincidir se observa desde la perspectiva del *Accuracy*, puesto que este representa cuantos puntos de la estimación han coincidido con la anotación, es decir si se tienen 35 puntos a comparar y coinciden 21 puntos de los 35, se tiene un *Accuracy* del 60%. Por tanto, para determinar un falso positivo o negativo se utilizó un valor umbral de coincidencia.

Visualmente lo anterior puede ser observado en la Figura 61, donde se muestra un caso en la imagen de la izquierda que se dan dos estimaciones, pero ambas no coinciden con ninguna marca del carril desde el punto de vista del umbral de coincidencia utilizado, por tanto, tienen un FPR y FNR igual al 100%, y en la imagen de la derecha un caso donde se da solamente una estimación correcta y la otra no coincide con ninguna marca del carril, por tanto la estimación para esta imagen tienen un FPR y FNR igual al 50%. El umbral de coincidencia utilizado en ambas imágenes fue 0,85.

Figura 61. Falsos positivos y negativos

Fuente: Elaboración propia.

Los resultados obtenidos de cada algoritmo propuesto en este trabajo se muestran en la Tabla *1* y Tabla *2*.

Tabla 1
Métricas de los algoritmos propuestos en el dataset TuSimple, umbral de coincidencia = 0,6

Algoritmos	*Accuracy*	FPR	FNR	Tiempo de ejecución
PPHT y Regresión lineal simple	74,30%	6,79%	23,92%	9,26 ms
PPHT, RANSAC y Regresión lineal simple	84,82%	10,95%	13,48%	10,14 ms

Fuente: Elaboración propia

Tabla 2
Métricas de los algoritmos propuestos en el dataset TuSimple, umbral de coincidencia = 0,85

Algoritmos	*Accuracy*	FPR	FNR	Tiempo de ejecución
PPHT y Regresión lineal simple	74,30%	14,77%	30,77%	9,26 ms
PPHT, RANSAC y Regresión lineal simple	84,82%	20,89%	23,23%	10,14 ms

Fuente: Elaboración propia

En la Tabla *1* y Tabla *2*, FPR indica el porcentaje de marcas del carril que fueron estimadas erróneamente y FNR indica el porcentaje de marcas del carril que no fueron detectadas, en cambio, el *Accuracy* indica el porcentaje de puntos que fueron estimados correctamente. Ambas tablas son mostradas para visualizar el impacto del umbral de coincidencia sobre el FPR y FNR, sin embargo, únicamente la Tabla *1* será utilizada como resultados en el dataset TuSimple, puesto que como se han modelado las marcas del carril con una recta, 60% de coincidencia de los puntos indica que se ha logrado detectar las marcas del carril satisfactoriamente.

Se concluye de los resultados numéricos la mejora significativa que causa las modificaciones realizadas en el segundo algoritmo, mejorando 10.79% la exactitud en la detección de carriles con respecto al primer algoritmo, como también puede observarse el resultado de haber disminuido el valor de los hiperparámetros del PPHT del segundo algoritmo en el FPR, es decir, se observa un incremento del 4.16% en la detección de carriles erróneas debido al ruido introducido. Por otro lado, se evidencia igualmente el efecto positivo en el FNR debido al preprocesamiento realizado y al RANSAC en el segundo algoritmo, disminuyendo en un 10.44% los carriles no detectados respecto al primer algoritmo.

A modo de comparativa los sistemas de detección de carriles del estado del arte basados en algoritmos de aprendizaje automático obtuvieron un *Accuracy* superior al 90% en este dataset, como ejemplo el algoritmo propuesto por Neven, Brabandere, Georgoulis y Gool (2018) obtuvo 96,40%, el propuesto por Liu, Yang, Wang y Wang (2018) obtuvo 92,03% y el ganador de la competencia propuesta por la empresa TuSimple, Inc., obtuvo 96,54% (Pan, Jianping, Luo, Wang, & Tang, 2018).

7.2 Dataset propio – Hernandarias

Para poder evaluar los algoritmos propuestos, en carriles de Paraguay y colocar a prueba los algoritmos con un nuevo dataset, se han recolectado en la ciudad de Hernandarias cuatro videos, dos durante el horario de la noche y dos durante el día, específicamente en el trayecto entre la ciudad de Hernandarias y Ciudad del Este, y en la avenida gastronómica del barrio Las Américas.

Estos videos fueron convertidos a imágenes a 30 fotogramas por segundo de video, seguidamente se han seleccionado aleatoriamente 156 imágenes para realizar las anotaciones únicamente de los carriles situados en la región frontal al vehículo. Estas anotaciones fueron llevadas a cabo con la versión gratuita del software Labelbox. El resultado de dichas anotaciones puede ser observado en la Figura 62.

Figura 62. Dataset Hernandarias

Fuente: Elaboración propia

Las anotaciones de puntos se han transformado del formato del JSON de salida del Labelbox, modelandolas con una curva de segundo orden, al formato del dataset TuSimple para llevar a cabo el cálculo de las métricas de desempeño.

Los resultados encontrados son expuestos en la Tabla *3*.

Tabla 3
Métricas de los algoritmos propuestos en el dataset Hernandarias, umbral de coincidencia=0,6

Algoritmos	*Accuracy*	FPR	FNR	Tiempo de ejecución
PPHT y Regresión lineal simple	42,60%	35,78%	63,81%	17,97 ms
PPHT, RANSAC y Regresión lineal simple	87,13%	8,89%	14,13%	11,10 ms

Fuente: Elaboración propia

De los resultados se puede observar un desempeño muy por debajo de lo aceptable del primer algoritmo en el dataset, de esto se concluye que los hiperparámetros definidos en base al dataset TuSimple para este algoritmo, si bien funcionan razonablemente bien para las pruebas en ese dataset, en un nuevo conjunto de imágenes como este se vuelve impracticable su utilización.

Para ambos algoritmos se pudo observar cierta fragilidad en el preprocesamiento de la imagen, es decir en la extracción de bordes, puesto que en zonas donde existen grandes sombras producidas por otros vehículos o árboles, debido a que su funcionamiento de detección de bordes se basa en el gradiente de intensidades, estas sombras introducen ruido, por lo cual la detección de las marcas del carril se vuelve inexacta. No obstante, en regiones de buena visibilidad de las marcas del carril tanto en horarios diurnos y nocturnos, el segundo algoritmo obtiene un buen desempeño.

En conclusión, con estos resultados se puede indicar parcialmente la viabilidad de detectar carriles en rutas del Paraguay, siempre y cuando las marcas del carril estén mínimamente visibles, la adquisición de imágenes se realice con el dispositivo adecuado y con su debida instalación en el vehículo. Aún así se recalca la necesidad de ampliar el dataset para indicar la viabilidad total de la detección de carriles en Paraguay.

7.3 Resultados CARLA

En el simulador CARLA se llevó a cabo la implementación del control autónomo de un vehículo, con el objetivo de mantenerlo en su respectivo carril utilizando como percepción del entorno, el algoritmo de detección de carriles. Para ello se ha utilizado el sistema propuesto en la sección 6.3, con ambos algoritmos de detección de carriles propuestos en este trabajo.

En la simulación se ha recabado a una frecuencia de 23Hz el error lateral del vehículo, es decir la distancia que se encuentra el centro geométrico del vehículo con respecto al centro del carril. Los valores de frecuencia pueden variar entre ejecuciones a causa de la configuración del hardware disponible y de la ejecución en un sistema operativo no determinístico, como un sistema de tiempo real.

Se ha seleccionado un circuito para el recorrido del vehículo como se puede visualizar en el mapa de la Figura 63. El punto inicial del circuito coincide con las coordenadas iniciales definidas del vehículo en la sección 6.3.1.

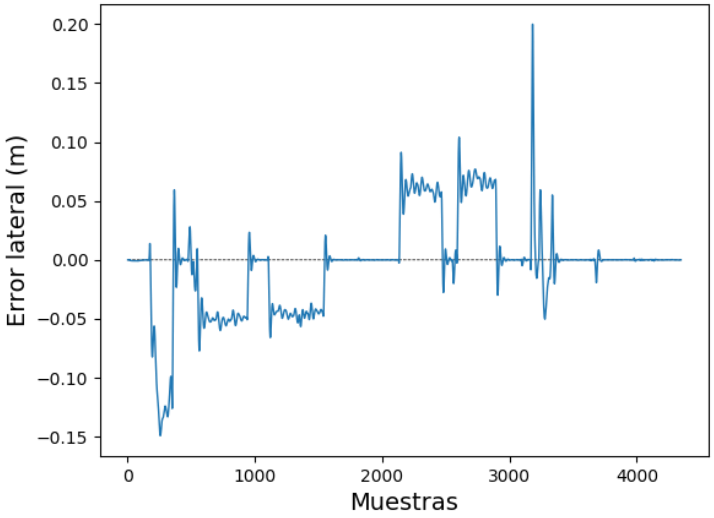
Figura 63. Circuito de la simulación

Fuente: Elaboración propia.

Para obtener un punto de comparación con el control implementado, se ha utilizado una funcionalidad provista por el simulador, la cual es la conducción autónoma del vehículo controlada directamente desde el lado del servidor, es decir la única acción del cliente es la activación o apagado de esta funcionalidad.

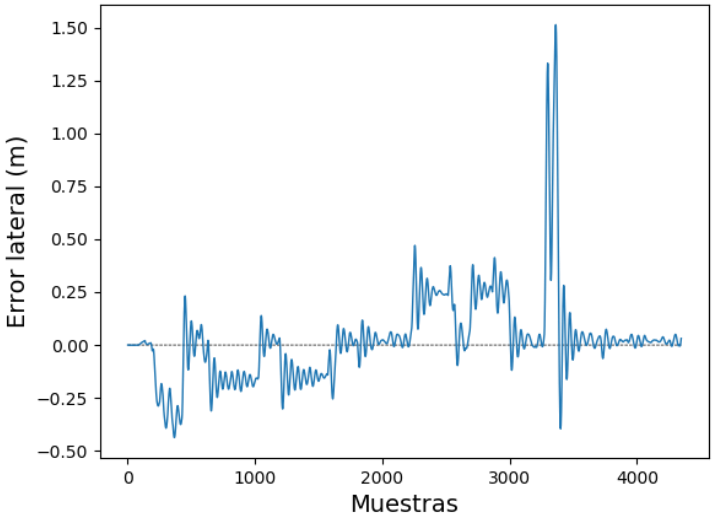
Los resultados obtenidos para el error lateral del vehículo tanto producidos por el modo autónomo del simulador y el control propuesto a través de los algoritmos de detección de carriles en este trabajo se pueden visualizar gráficamente en las Figura 64, Figura 65 y Figura 66. Cabe resaltar que las tres mediciones fueron hechas a una velocidad constante igual a 80 km/h.

Figura 64. Error lateral del vehículo controlado por el modo autónomo del servidor



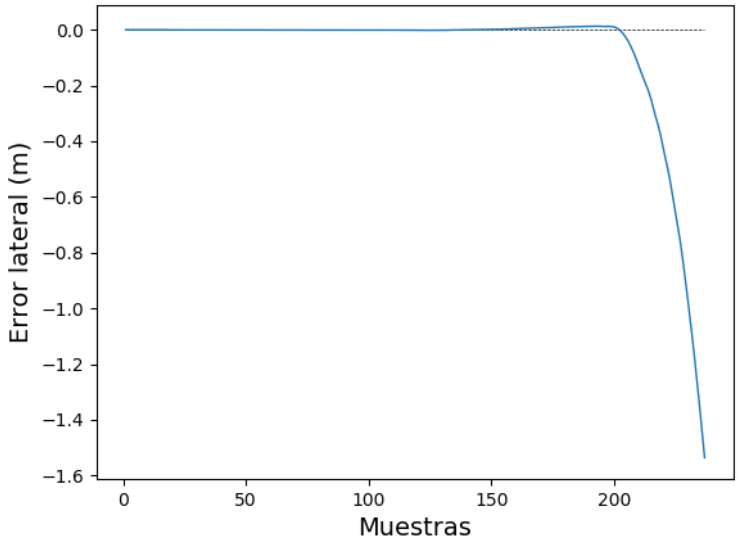
Fuente: Elaboración propia.

Figura 65. Error lateral del vehículo controlado por el algoritmo PPHT, RANSAC



Fuente: Elaboración propia

Figura 66. Error lateral del vehículo controlado por el algoritmo PPHT



Fuente: Elaboración propia

Se han calculado los errores cuadráticos medios (RMSE, por sus siglas en inglés), el error absoluto medio (*MAE*, por sus siglas en inglés) y el valor pico del error lateral de las mediciones de cada prueba. Estos valores se pueden observar en la Tabla *4*.

Tabla 4
Métricas de las pruebas de control realizadas en el simulador CARLA

Control	RMSE (m)	MAE (m)	Valor pico (m)
Servidor	0,041649898	0,025734149	0,199826431
PPHT y Regresión lineal simple	*	*	*
PPHT, RANSAC y Regresión lineal simple	0,217497619	0,128952579	1,512534053

* No ha completado el circuito

Fuente: Elaboración propia.

Una medida importante para tener en cuenta y entender la conclusión a seguir es la longitud transversal del carril, este mide 3,5 metros.

Por los errores y las gráficas presentadas se constata que el algoritmo de control de dirección implementado no es un control óptimo, puesto que como se observa en la Tabla *4* sus errores tanto el MAE como el RMSE son aproximadamente cinco veces mayores que el punto de comparación tomado y se observa bastantes oscilaciones alrededor del set point, Figura 65. Como también en un instante de tiempo el error lateral alcanza un valor pico de 1,51 metros, con lo cual, a pesar de mantener al vehículo en su carril, no impide que gran parte del vehículo esté fuera de su carril

Para solucionar esto como se ha dicho anteriormente, se debe implementar un control de dirección utilizando un control clásico como el PID o un control moderno como el MPC. Esto no fue diseñado ni implementado debido a que no forma parte de los objetivos específicos de este trabajo final de grado.

No obstante, se concluye que el control autónomo del vehículo con el objetivo de mantenerlo en su carril fue alcanzado a través del algoritmo de detección de carriles que utiliza PPHT y RANSAC, puesto que el vehículo ha culminado el circuito de forma exitosa sin abandonar su carril.

Consideraciones finales

Conclusión

El presente proyecto final de grado tuvo como objetivos el estudio, evaluación e implementación de un algoritmo de detección de carriles para la navegación autónoma de un vehículo, los cuales fueron alcanzados satisfactoriamente.

Para alcanzar estos objetivos, primero se realizó un estudio de los algoritmos de detección de carriles, tanto por el aspecto de la academia como por el punto de vista de la industria. De los ejemplos de la industria se observó su importancia en los sistemas ADAS y en los vehículos autónomos.

Se presentó una división propuesta por Xing et al. (2018) respecto a los algoritmos, separando aquellos que se basan únicamente en el procesamiento de imágenes y aquellos con un enfoque novedoso, basados en aprendizaje automático. De los basados en procesamiento de imágenes los algoritmos más utilizados como partes del algoritmo como un todo, son la transformada de Hough, RANSAC, filtro de Kalman y distintos filtros de detección de bordes. En cambio, los basados en aprendizaje automático, las arquitecturas más comunes utilizadas son las CNNs, SVMs como clasificadores, con una etapa posterior de modelado a través, por ejemplo, de RANSAC.

De estos enfoques fue seleccionado el basado en procesamiento de imágenes. Fueron diseñados e implementados dos algoritmos, donde el primero es una implementación del código propuesto por Desegur (2018) y el segundo algoritmo es un diseño del autor de este proyecto.

Para medir el desempeño de ambos algoritmos se ha realizado dos evaluaciones *offline*, la primera en el dataset público TuSimple y la segunda en el dataset privado Hernandarias.

Por último, en el simulador CARLA se ha llevado a cabo la implementación de ambos algoritmos para la navegación autónoma de un vehículo, utilizándolos como medio de percepción del entorno.

De lo anterior se concluye:

Del estudio de los algoritmos de detección de carriles se observó que, a mediados de la presente década hubo un cambio en cómo se daba solución al problema de detectar carriles, puesto que anteriormente esta tarea era en su mayoría resuelta con algoritmos basados en procesamiento de imágenes. Esta nueva solución son los algoritmos basados en aprendizaje automático, que han obtenido un desempeño superior en la detección de carriles.

No obstante, la cantidad de imágenes que representen a las diversas situaciones posibles del entorno, requeridas para entrenar y obtener un modelo con desempeño superior a los algoritmos basados en procesamiento de imágenes, se hace difícil en la mayoría de los casos, como también el coste computacional reflejado en el tiempo de ejecución es un punto crucial que considerar. Por tanto, considerando lo anterior, los algoritmos basados en procesamiento de imágenes continúan siendo totalmente válidos para la detección de carriles.

De las evaluaciones cuantitativas *offline* realizadas en el lenguaje de programación Python se evidenció, una cierta fragilidad de los algoritmos desarrollados en este proyecto con cambios bruscos de luminosidad, y la robustez introducida por los algoritmos RANSAC y el método de Otsu al segundo algoritmo propuesto. Lo anterior es un reflejo del desempeño superior y estable de este segundo algoritmo en todas las evaluaciones llevadas a cabo. De esta manera se ha validado la elección tomada de utilizar el RANSAC, en base al artículo de Xing et al. (2018), y el algoritmo de Canny y Otsu, en base a lo propuesto por Fang et al. (2009).

De la conjetura realizada sobre modelar linealmente a los carriles se comprobó visualmente que, para la región frontal lejana al vehículo, principalmente en las curvas, el modelo presentó una exactitud baja, sin embargo, para la región cercana al vehículo la exactitud es alta y la estimación puede servir como una primera aproximación del carril.

De la implementación en el ambiente de un vehículo autónomo, a través del simulador CARLA, se ha logrado constatar, la posibilidad de mantener a un vehículo en su respectivo carril de forma autónoma únicamente por intermedio de la detección del carril, con el algoritmo diseñado.

Trabajos futuros

Se proponen los siguientes trabajos futuros, donde este trabajo servirá como una fuente de consulta y conocimiento.

- Diseñar un sistema de advertencia de abandono de carril e implementarlo en un vehículo a escala real.
- Diseñar e implementar un algoritmo de detección de carriles basado en redes neuronales convolucionales.
- Implementar los algoritmos propuestos en este proyecto en un vehículo eléctrico a escala.
- Diseñar un algoritmo de control clásico o moderno para el sistema de ayuda de permanencia en el carril.
- En base a los algoritmos propuestos, modelar los carriles con curvas de segundo orden o superior y agregar una etapa de predicción de las siguientes posiciones de las marcas del carril, por ejemplo, a través del filtro de Kalman, el filtro de Kalman extendido o un filtro de partículas.

Bibliografía

Alegre, E., Pajares, G., & de la Escalera, A. (2016). *Conceptos y Métodos en Visión por Computador*. España: Grupo de Visión del Comite Español de Automática (CEA).

Aly, M. (2008). Real time Detection of Lane Markers in Urban Streets. *IEEE Intelligent Vehicles Symposium*, 7-12.

Amini, A., Gilitschenski, I., Phillips, J., Moseyko, J., Banerjee, R., Karaman, S., & Rus, D. (2020). Learning Robust Control Policies for End-to-End Autonomous Driving from Data-Driven Simulation. *IEEE Robotics and Automation Letters*, 1-7.

Ayala, G. (2019). *Estadística Básica.* Valencia: Sam Hocevar.

Bae, C. (2017). *finding-lane-lines*. Obtenido de GitHub, Inc.: https://github.com/windowsub0406/finding-lane-lines/blob/master/SDC_project_1.py

Bertozzi, M., & Broggi, A. (1998). GOLD: A Parallel Real-Time Stereo Vision System for Generic Obstacle Lane Detection . *IEEE Transactions on Image Processing*, 62.

Bloomberg L.P. (15 de 05 de 2020). *The State of the Self-Driving Car Race 2020*. Recuperado el 18 de 05 de 2020, de Bloomberg: https://www.bloomberg.com/features/2020-self-driving-car-race/

Borkar, A., Hayes, M., & Smith, M. T. (2012). A Novel Lane Detection System With Efficient Ground Truth Generation. *IEEE Transaction on Intelligent Transportation Systems*, 365-373.

Buehler, M., Iagnemma, K., & Singh, S. (2009). *The DARPA Urban Challenge.* Berlin: Springer.

CARLA Team. (s.f.). *CARLA Documentation*. Recuperado el 10 de 04 de 2020, de CARLA Documentation: https://carla.readthedocs.io/en/latest/

Cattaruzza, M. (2019). *Design and Simulation of Autonomous Driving Algorithms (Tesis de Maestría)*. Politecnico Di Torino, Turin.

Chen, C., Seff, A., Kornhauser, A., & Xiao, J. (2015). DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving. *IEEE International Conference on Computer Vision*, 2722.

Desegur, L. (2018). *A Lane Detection Approach for Self-Driving Vehicles*. Obtenido de GitHub, Inc.: https://github.com/ldesegur/CarND-LaneLines-P1

Dosovitskiy, A., Ros, G., Codevilla, F., A. L., & Koltun, V. (2017). CARLA: An Open Urban Driving Simulator. *Proceedings of the 1st Annual Conference on Robot Learning*, 1-16.

Espinosa Aranda, J. L., Fernández Carrobles, M., & Vállez, N. (2016). Operaciones sobre el histograma y filtrado de la imagen. En E. Alegre, G. Pajares, & A. d. Escalera, *Conceptos y Métodos en Visión por Computador* (págs. 37-45). España: Grupo de Visión del Comité Español de Automática (CEA).

Euro NCAP. (06 de 2019). *Test Protocol – Lane Support Systems.* Obtenido de Euro NCAP: https://cdn.euroncap.com/media/53143/euro-ncap-lss-test-protocol-v302.pdf

Euro NCAP. (s.f.). *Euro NCAP | Newsroom*. Recuperado el 20 de 02 de 2020, de Euro NCAP: https://euroncap.newsmarket.com/

Fagnant, D. J., & Kockelman, K. . (2015). Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, 167.

Fang, M., Yue, G., & Yu, Q. (2009). The Study on An Application of Otsu Method in Canny Operator. *Proceedings of the 2009 International Symposium on Information Processing (ISIP'09)*, 109-112.

Forsyth, D. A., & Ponce, J. (2012). *Computer Vision a Modern Approach.* New Jersey : Pearson.

Galván, P. (s.f.). *Los 6 Niveles de Autonomía*. Recuperado el 13 de 02 de 2020, de Software Guru: https://sg.com.mx/revista/55/los-6-niveles-autonom

Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks.* Toronto: Springer.

Guijarro, M., Herrera, P. J., & Montalvo, M. (2016). Segmentación de regiones . En E. Alegre, G. Pajares, & A. d. Escalera, *Conceptos y Métodos en Visión por Computador* (págs. 99-107). España: Grupo de Visión del Comité Español de Automática (CEA).

Hall, R. (1989). *Illumination and Color in Computer Generated Imagery.* New York: Springer-Verlag.

Herrera, P. J., Guijarro, M., & Guerrero, J. M. (2016). Operaciones de Transformación de Imágenes. En E. Alegre, o. Pajares, & r. d. Escalera, *Conceptos y Métodos en Visión por Computador* (pág. 70). España: Grupo de Visión del Comité Español de Automática (CEA).

IndustryARC. (s.f.). *Advanced Driver Assistance Systems (ADAS) Market - Forecast(2020 - 2025).* Recuperado el 17 de 02 de 2020, de IndustryARC: https://www.industryarc.com/Report/20/advanced-driver-assistance-systems-market.html

Kala, R. (2016). *On-Road Intelligent Vehicles Motion Planning for Intelligent Transportation Systems.* Oxford: Elsevier Inc.

Kanan, C., & Cottrell, G. W. (2012). Color-to-Grayscale: Does the Method Matter in Image Recognition? *PLoS ONE*, 1-7.

Kapoor, A., Shah, S., Dey, D., & Lovett, C. (14 de Noviembre de 2017). AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. *Field and Service Robotics*, 1-14. Obtenido de Microsoft AirSim now available on Unity: https://www.microsoft.com/en-us/research/blog/microsoft-airsim-now-available-on-unity/

Kiryati, N., Eldar, Y., & Bruckstein, A. M. (1991). A probabilistic Hough transform. *Pattern Recognition*, 303-316.

Kumar, A. M., & Simon, P. (2015). Review of Lane Detection and Tracking Algorithms in Advanced Driver Assist System. *International Journal of Computer Science & Information Technology (IJCSIT)*, 76-77.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 2278–2324.

Li, J., Mei, X., Prokhorov, D., & Tao, D. (2016). Deep Neural Network for Structural Prediction and Lane Detection in Traffic Scene . *IEEE Transactions on Neural Networks and Learning Systems*, 1-12.

Li, Q., Chen, L., Li, M., Shaw, S.-L., & Nüchter, A. (2014). A SensorFusion Drivable-Region and Lane-Detection System for Autonomous Vehicle Navigation in Challenging Road Scenarios. *IEEE Transactions on Vehicular Technology*, 540.

Liu, P., Yang, M., Wang, C., & Wang, B. (2018). Multi-lane Detection via Multi-task Network in Various Road Scenes. *2018 Chinese Automation Congress (CAC)*, 2750-2754.

Maini, R., & Aggarwal, H. (2009). Study and Comparison of Various Image Edge Detection Techniques . *International Journal of Image Processing (IJIP)*, 1-10.

Martín, D., García, F., & Armingol, J. M. (2016). El color: Modelos y Transformaciones de los Espacios de Color . En E. Alegre, G. Pajares, & A. d. Escalera, *Conceptos y Métodos en Visión por Computador* (págs. 47-55). España: Grupo de Visión del Comité Español de Automática (CEA).

Matas, J., Galambos, C., & Kittler, J. (2000). Robust Detection of Lines Using the Progressive Probabilistic Hough Transform. *Computer Vision and Image Understanding*, 119-137.

Mendenhall, W., Beaver, R. J., & Beaver, B. M. (2010). *Introducción a la probabilidad y estadística* . México, D.F.: Cengage Learning.

Mobileye. (s.f.). Mobileye 6-Series User Manual. Recuperado el 17 de 02 de 2020, de http://www.c2sec.com.sg/Files/UserManualMobileye6.pdf

National Highway Traffic Safety Administration . (s.f. (a)). *Vehículos Automatizados para la Seguridad*. Recuperado el 14 de 02 de 2020, de NTHSA: National Highway Traffic Safety Administration: https://www.nhtsa.gov/es/tecnologia-e-innovacion/vehiculos-automatizados-para-la-seguridad

National Highway Traffic Safety Administration. (02 de 2013). Lane Departure Warning System Confirmation Test and Lane Keeping Suport Performance Documentation. 14-29. Washington, DC .

National Highway Traffic Safety Administration. (s.f. (b)). *About NHTSA*. Recuperado el 17 de 02 de 2020, de NHTSA: National Highway Traffic Safety Administration: https://www.nhtsa.gov/about-nhtsa

National Highway Traffic Safety Administration. (s.f. (c)). *Driver Assistance Technologies*. Recuperado el 17 de 02 de 2020, de NHTSA: National Highway Traffic Safety Administration: https://www.nhtsa.gov/equipment/driver-assistance-technologies

Neven, D., Brabandere, B. D., Georgoulis, S., & Gool, M. P. (2018). Towards End-to-End Lane Detection: an Instance Segmentation. *2018 IEEE Intelligent Vehicles Symposium (IV)*, 286-290.

Open Source Computer Vision. (s.f.). *Canny Edge Detection*. Recuperado el 01 de 03 de 2020, de OpenCV Open Source Computer Vision: https://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html

Otsu, N. (1979). A Tlreshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 62-66.

Özgüner, Ü., Acarman, T., & Redmill, K. (2011). *Autonomous Ground Vehicles.* Norwood: Artech House .

Pan, X., J. S., Luo, P., Wang, X., & Tang, X. (2018). Spatial As Deep: Spatial CNN for Traffic Scene Understanding. *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 7276-7282.

Pohl, J., Birk, W., & Westervall, L. (2007). A driver-distraction-based lane-keeping assistance system. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control*, 541-548.

Prince, S. J. (2012). *Computer vision: models, learning and inference.* New York: Cambridge University Press.

Robert Bosch GmbH. (07 de 2013). *Bosch @ Auto Shanghai 2015*. Obtenido de http://auto2015.bosch.com.cn/ebrochures2015/automated/cc/da/mpc2/datenblatt_mpc2_en.pdf

Robert Bosch GmbH. (s.f.). *Lane departure warning*. Recuperado el 19 de 02 de 2020, de BOSCH Invented for life: https://www.bosch-mobility-solutions.com/en/products-and-services/passenger-cars-and-light-commercial-vehicles/driver-assistance-systems/lane-departure-warning/

Rosco Collision Avoidance. (s.f.). *Mobileye Single Camera*. Recuperado el 18 de 02 de 2020, de ROSCO Collision Avoidance: https://rosco-adas.com/single-camera/

Rosique, F., Navarro, P. J., Fernández, C., & Padilla, A. (2019). A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research. *Sensors*, 2-8.

SAE International. (2018). Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. *SAE On-Road Automated Vehicle Standards Committee*, 1-25.

Sampieri, R. H., Collado, C. F., Lucio, M. d., Valencia, S. M., & Torres, C. P. (2014). *Metodología de la investigación (6ta edición).* México D.F.: Mc Graw Hill.

Sánchez Salmerón, A. J., & Ricolfe Viala, C. (2016). Sistema de Captura de Imágenes. En E. Alegre, G. Pajares, & A. d. Escalera, *Conceptos y Métodos en Visión por Computador* (págs. 11-12). España: Grupo de Visión del Comité Español de Automática (CEA).

Setiawan, B. D., Rusydi, A. N., & Pradityo, K. (2017). Lake Edge Detection Using Canny Algorithm and. *2017 International Symposium on Geoinformatics (ISyG)*, 72-75.

Sinha, U. (s.f.). *Greyscale and RGB color spaces*. Recuperado el 24 de 02 de 2020, de AI Shack: https://aishack.in/tutorials/color-spaces-1/

Stanford University School of Engineering. (11 de 08 de 2017). *Lecture 1 | Introduction to Convolutional Neural Networks for Visual Recognition [Archivo de video]*. Recuperado el 14 de 05 de 2020, de Youtube: https://www.youtube.com/watch?v=vT1JzLTH4G4&t=3032s

Sucar, L. E., & Gómez, G. (s.f.). *Visión Computacional.* Instituto Nacional de astrofísica, óptica.

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications.* Springer.

Tanaka, T., Nakajima, S., Urabe, T., & Tanaka, H. (2016). Development of Lane Keeping Assist System Using LateralPosition-Error Control at Forward Gaze Point. *SAE Technical Paper*, 1.

Tesla. (27 de 01 de 2020). *Model S Owner`s Manual.* Obtenido de Tesla: https://www.tesla.com/sites/default/files/model_s_owners_manual_north_america_en_us.pdf

The MathWorks, Inc. (s.f. (a)). *MathWorks*. Recuperado el 05 de 03 de 2020, de ROI-Based Processing: https://www.mathworks.com/help/images/roi-based-processing.html

The MathWorks, Inc. (2016). *The MathWorks, Inc.* Obtenido de Developing Reliable ADAS and Autonomous Driving Functions in MATLAB and Simulink: https://au.mathworks.com/content/dam/mathworks/mathworks-dot-com/company/events/conferences/automotive-conference-stuttgart/2016/proceedings/developing-reliable-adas-autonomous-driving-functions-in-matlab-simulink.pdf

Trespaderne, F. M., de la Fuente, E., & Gómez García Bermejo, J. (2016). Ajuste y detección de modelos geométricos en imágenes. Algoritmo RANSAC. En E. Alegre, G. Pajares, & A. d. Escalera, *Conceptos y Métodos en Visión por Computador* (págs. 292-300). España: Grupo de Visión del Comité Español de Automática (CEA).

TuSimple, Inc. (2017). *TuSimple Lane Detection Challenge*. Recuperado el 20 de 04 de 2020, de Github, Inc.: https://github.com/TuSimple/tusimple-benchmark/tree/master/doc/lane_detection

Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M., . . . Ferguson, D. (2009). Autonomous Driving in Urban Environments: Boss and the Urban Challenge . En M. Buehler, K. Iagnemma, & S. Singh, *The DARPA Urban Challenge Autonomous Vehicles in City Traffic* (pág. 23). Berlin: Springer.

Vaa, T., Penttinen, M., & Spyropoulou, I. (2007). Intelligent transport systems and effects on road traffic accidents: state of the art. *Intelligent Transport Systems and Services*, 81-88.

Vélez Serrano, J. F., Moreno Díaz, A. B., Sánchez Calle, Á., & Sánchez Marín, J. L. (2003). *Visión por Computador.* Madrid: Dykinson.

Xing, Y., Lv, C., Chen, L., Wang, H., Wang, H., Cao, D., . . . Wang, F.-Y. (2018). Advances in Vision-Based Lane Detection: Algorithms, Integration, Assessment, and Perspectives on ACP-Based Parallel Vision . *IEEE/CAA Journal of Automatica Sinica*, 646.

Yadav, S., Patra, S., Arora, C., & Banerjee, S. (2017). Deep CNN with color lines model for unmarked road segmentation. *IEEE International Conference on Image Processing* , 586.

Apéndice

Apéndice A

Estadística básica

Los conceptos y ecuaciones a continuación fueron obtenidas de (Ayala, 2019).

A.1. Medidas de centralización

A.2.1 Media muestral

La media aritmética o muestral de un conjunto de $\{x_1, x_2, \dots, x_n\}$ de n datos se obtiene al sumar cada elemento de este conjunto y dividirlo por la cantidad de elementos. Su valor está dado por la siguiente ecuación.

$$\overline{x} = \sum_{i=1}^{n} \frac{x_i}{n} \tag{A.1}$$

A.2.2 Mediana

La media muestral a pesar de ser un valor que indica el orden de grandeza de los datos está sujeta a grandes variaciones a causa de valores extremos. En cambio, una medida más robusta ante estos valores extremos es la mediana M_e, que divide los datos ordenados de menor a mayor en dos partes, el 50% de los n datos son menores a su valor y el 50% restante son superiores. Una definición formal de lo anterior se da con las siguientes ecuaciones.

$$\frac{|\{x_i|\ x_i \leq M_e\}|}{n} = 0{,}5$$
$$\frac{|\{x_i|\ x_i \geq M_e\}|}{n} = 0{,}5 \qquad \text{(A.2)}$$

A.2. Probabilidad

A.2.1 Probabilidad de un suceso

Para definir la probabilidad de un suceso primeramente se debe definir lo siguiente.

Un espacio muestral S es aquel conjunto de posibles resultados de un cierto experimento, como lanzar una moneda o un dado.

Un suceso aleatorio A es un subconjunto del espacio muestral. Luego, la probabilidad de un suceso, teniendo en cuenta el número de elementos A denotados por $|A|$ y el espacio muestral está dada por la siguiente ecuación.

$$P(A) = \frac{|A|}{|S|} = \frac{Casos\ favorables\ a\ que\ ocurra\ A}{Casos\ posibles} \qquad \text{(A.3)}$$

A.2.2 Variable aleatoria

La variable aleatoria es aquella función X que asocia todos los elementos de S a los números reales, formalmente se denota con la siguiente ecuación.

$$X: S \rightarrow \mathbb{R} \qquad \text{(A.4)}$$

A.2.3 Variable discreta

Es una variable aleatoria que toma un número finito de valores o bien toma un número infinito de posibles valores que podemos numerar.

A.2.4 Varianza y media de una variable aleatoria discreta

Una manera de describir de una manera simple a una variable se logra a través de hallar su media y su varianza, donde la media se puede interpretar, como el valor alrededor de cual se encuentran los valores aleatorios de la variable y la varianza indica la dispersión de estos valores alrededor de la media. Su definición matemática se da a continuación.

La media de esta variable aleatoria discreta que toma los valores $\{x_1, x_2, \dots\}$ está dada por,

$$E(X) = \mu = \sum_{i=1}^{\infty} x_i P(X = x_i) \tag{A.5}$$

La varianza de esta puede ser calculada con la siguiente ecuación,

$$var(X) = \sigma^2 = E(X - \mu)^2 = \sum_{i=1}^{\infty} (x_i - \mu)^2 P(X = x_i) \tag{A.6}$$

A.2.5 Variable normal

Una variable aleatoria X, con media μ y varianza σ^2 se dice que sigue una distribución normal o gaussiana cuando su función de densidad se representa de la siguiente manera.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad -\infty < x < +\infty \tag{A.7}$$

Esta se denota con $X \sim N(\mu, \sigma^2)$ y cuando la media es nula y la varianza es unitaria, esta variable presenta una distribución llamada normal estándar.

Apéndice B

Convolución en imágenes

Una de las formas de realizar ajustes a la imagen tales como suavizar, acentuar bordes o remover bordes puede ser llevado a cabo con la convolución entre una imagen y una matriz. A continuación, se detallará cómo se realizan los cálculos de esta convolución partiendo desde su definición formal hasta ejemplos concretos, como también se expondrá una noción intuitiva de los elementos de esta matriz.

B.1 Definición y cálculo

Según Szeliski (2010), la convolución de una imagen f con una matriz h, está dada por la siguiente ecuación,

$$g(i,j) = \sum_{x,y} f(x,y)h(i-x, j-y) = f * h \tag{B.1}$$

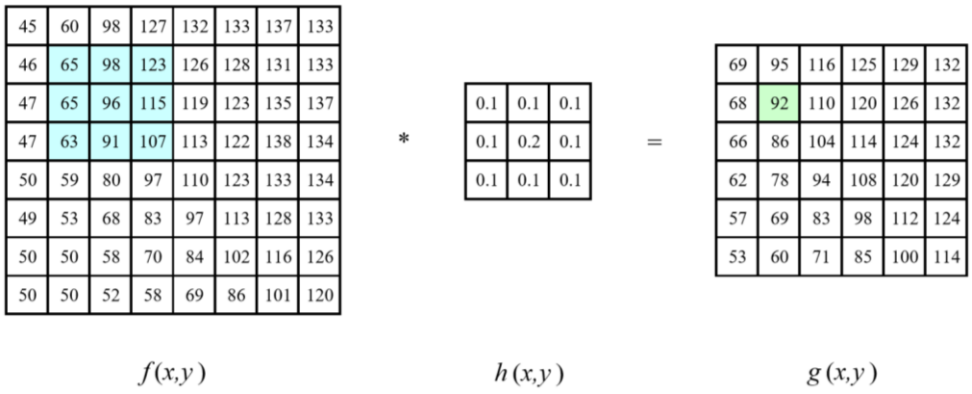
Esta ecuación puede interpretarse como la superposición (suma) de una matriz, que representa a la función impulso, multiplicada por los píxeles de entrada de la imagen y que da como resultado la nueva imagen g. Normalmente a esta matriz h se la llama máscara o *kernel*. Una de las características a tener en cuenta es que esta es una transformación lineal de la imagen, por lo tanto, cumple las propiedades de linealidad.

La forma de calcular esta convolución es sencilla y se demostrará con la Figura 67. Considerando en la imagen $f(x,y)$ con sus respectivas intensidades, el elemento 65 en la posición

(1,1) y el *kernel* $h(x, y)$, el primer paso consiste en "situar" el primer elemento del *kernel* sobre el elemento en cuestión y multiplicar cada píxel de la imagen bajo el elemento correspondiente del *kernel* y finaliza con la adición de estas multiplicaciones, desarrollando lo anterior numéricamente se tiene lo siguiente.

$$g(1,1) = (65 * 0{,}1) + (98 * 0{,}1) + (123 * 0{,}1) + (65 * 0{,}1) + (96 * 0{,}2) + (115 * 0{,}1)$$
$$+ (63 * 0{,}1) + (91 * 0{,}1) + (91 * 0{,}1) + (107 * 0{,}1) = 91{,}9 \cong 92$$

Figura 67. Cálculo de la convolución entre una imagen y un *kernel*



Fuente: Szeliski (2010).

Luego este proceso se repite para el siguiente elemento de la columna y al llegar al último elemento de esta, se desciende a la siguiente fila, para comenzar el proceso nuevamente.

Claramente esto da problemas en los píxeles del borde puesto que elementos del *kernel* se encuentran fuera de la imagen, para ello existen varias técnicas que han sido propuestas como la técnica *zero*, que establece todos los píxeles fuera de la imagen con el valor cero, la técnica *wrap* que aplica un lazo en la configuración toroidal a la imagen, entre otras.

Este proceso observado de multiplicar elementos asociados y luego sumarlos es justamente lo mismo que considerar a los elementos de la imagen como un vector, de la misma manera a los elementos del *kernel* y calcular su producto escalar o punto (Forsyth & Ponce, 2012).

Esta es una analogía que brinda la posibilidad de observar al resultado de esta convolución, como cualquier otro producto escalar. De esta manera un valor mayor como resultado indica que el vector de la imagen es paralelo al vector del *kernel* o el filtro, es decir, en regiones de la imagen donde su patrón coincida con el patrón del *kernel,* se obtendrá valor mayores o en términos de intensidades una región más brillante.

B.2 Elementos del *kernel*

Los elementos del *kernel* definen el tipo de transformación que se obtendrá de la imagen original. Para obtener una noción intuitiva del porqué de sus valores, se dará un ejemplo para la detección de bordes al derivar una imagen, en base a lo escrito por (Forsyth & Ponce, 2012).

Debido a que la función de una imagen es bidimensional se debe derivarla parcialmente, por tanto, la derivada en la dirección x de la imagen está dada por,

$$\frac{\partial f}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon, y) - f(x, y)}{\epsilon} \tag{B.2}$$

Esta derivada parcial puede ser estimada por una diferencia finita simétrica, es decir, tomar en cuenta la diferencia entre los dos píxeles vecinos al píxel a ser derivado:

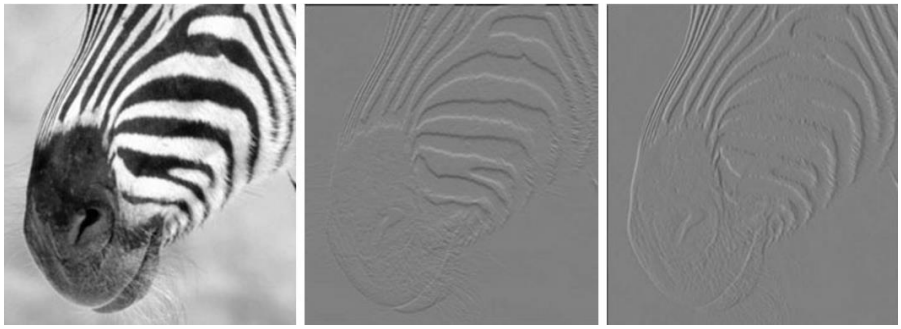
$$\frac{\partial f}{\partial x} \approx f_{i+1,j} - f_{i-1,j} \tag{B.3}$$

Es en este momento donde los elementos del *kernel* permiten el cálculo de esta aproximación. De la ecuación (B.3) se visualiza que el siguiente *kernel* permite llevar a cabo su cálculo en toda la imagen.

$$h(x,y) = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \tag{B.4}$$

Aplicando este *kernel* a la imagen de la izquierda de la Figura 68, se obtiene el resultado observado en la imagen de la derecha. La imagen del centro corresponde al resultado de aplicar una derivada parcial en la dirección y.

Figura 68. Resultado de la derivada de una imagen



Fuente: Forsyth y Ponce (2012).

Este *kernel* tiene un desempeño pobre cuando en la imagen se introduce ruido, debido a su limitada extensión espacial, para poder solucionar se ha de aumentarlo, como también realizar previamente la aplicación de un filtro tal como el gaussiano. Unos ejemplos del aumento de esta

extensión espacial pueden darse con los operadores de Prewitt (B.5) y Sobel (B.6). Donde con estos operadores es posible encontrar tanto la magnitud y dirección del gradiente en cada píxel de la imagen (Prince, 2012).

$$K_x = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} \quad K_y = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \tag{B.5}$$

$$K_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \quad K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \tag{B.6}$$

Cabe resaltar que el patrón de estos *kernels* es siempre el mismo, pero dependiendo del autor que se consulte o de la implementación consultada los signos pueden estar intercambiados, lo cual afectará únicamente en el signo de este gradiente y la representación del borde cuando se pase de una región clara a oscura o viceversa.