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

Ingeniería Electromecánica con Orientación Electrónica

Jesús María Franco Santacruz - 46204

jesus.franco@uc.edu.py

***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, 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.**

I. Introducción

IDENTIFICAR 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 en ambos la detección de carriles juega un papel fundamental en asistir al conductor [1].

Normalmente los algoritmos de detección de carriles pueden ser desmenuzados en cuatro aspectos [1]. 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. 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 el algoritmo.

Investigaciones/trabajos precedentes: a lo largo de los años se han propuesto distintos métodos para la detección de carriles. Algunos de estos algoritmos para extraer las características del carril utilizaron, la umbralización adaptativa por intermedio el método umbralización mínimo entre máximos (MBMT, por sus siglas en inglés) [2], filtros orientables [3], o a través de los operadores de Sobel [4]. Para el post procesamiento de estas caracteristicas y fijarlas a un cierto modelo algunos de los algoritmos utilizados fueron, la transformada probabilistica adaptativa de Hough (ARHT, por sus siglas en ingles) [5], el modelado de los carriles por una recta y por splines cúbicas de Bézier a través del RANSAC [6], como también la division de la imagen percibida en carriles ecercanos y lejanos, donde los carriles cercanos se han modelado a través de una linea con la transformada de Hough y para la region lejana a una hiperbola a traves del RANSAC [7]. Para la predicción de los carriles y robustez frente a ruido se han aplicado, el filtro de partículas [8], como también el filtro de Kalman [9]. Un nuevo enfoque para dar solución a este problema consiste en aplicar los sistemas de aprendizaje automático, ejemplos de algoritmos de estos sistemas son: las redes neuronales convolucionales

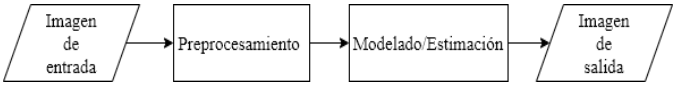

Fig. 1 Etapas de los algoritmos propuestos.

(CNN, por sus siglas en inglés), las redes neuronales recurrentes (RNN, por sus siglas en inglés) y las máquinas de vectores de soporte (SVM, por sus siglas en inglés). De los anteriores se pueden citar CNN y RNN como clasificadores y regresores [10] y CNN espaciales [11].

Considerando todos los desafíos mencionados anteriormente, se propone dos algoritmos de detección de carriles basados en procesamiento de imágenes. Ambos constan de dos grandes etapas: preprocesamiento y modelado/estimación como puede verse en la Fig. 1. 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.

Cabe resaltar que el primer algoritmo es una implementación del código desarrollado por [12]. 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 la siguiente sección.

En la etapa de preprocesamiento, en ambos se aplica un filtro gaussiano para la eliminación de ruido y como algoritmo de extracción de bordes se utiliza Canny. En el segundo algoritmo se ha optado por trabajar con una imagen menor y con una región de interés (ROI, por sus siglas en inglés) diferente por lo cual se ha podido aumentar la cantidad de líneas detectadas en la transformada probabilística progresiva de Hough (PPHT, por sus siglas en inglés) sin perder un desempeño aceptable de ejecución, como también se han realizado cambios en la separación de las marcas del carril y en la detección de bordes, al utilizar el algoritmo de Canny con la umbralización de Otsu, aumentando de esta manera la robustez ante cambios de contraste de la imagen.

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 del algoritmo PPHT en conjunto con una regresión lineal simple y en el segundo algoritmo con el objetivo de obtener una detección más robusta ante datos atípicos (*outliers)*, se ha agregado el algoritmo RANSAC que finaliza igualmente con una regresión lineal simple con un ajuste por mínimos cuadrados.

Organización del articulo: en la Sección II se explica el diseño e implementación de los algoritmos utilizados para la extracción de características y modelado, como también se presenta la implementación realizada en el simulador CARLA. En la Sección III se exponen los resultados cuantitativos obtenidos en los datasets y en el simulador. En la Sección IV se presenta las conclusiones del trabajo realizado y futuros trabajos.

Fig. 2 Imagen de entrada. Ejemplo de carril del dataset TuSimple.

II. Detección de carriles

A. Preprocesamiento

Como se observa en la Fig. 3, el preprocesamiento propuesto de ambos algoritmos presenta similitudes que serán explicadas para ambos, como también diferencias puntuales a resaltar. Para ambos algoritmos se trabajará con una imagen de entrada como el de la Fig. 2 en el espacio de color RGB (*Red*, *Green*, *Blue*) de resolución espacial de 1280x720 píxeles.

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. 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 los posteriores algoritmos del preprocesamiento de este algoritmo.

2) Difuminado gaussiano

Para la eliminación de ruido, entiéndase por cambios bruscos de intensidad que no son debido a las marcas del carril, se aplica el difuminado gaussiano (*Gaussian Blur*) con un *kernel* de tamaño 7x7, con desviaciones estándar de 1,4 calculadas a partir del tamaño del *kernel*.

3) Conversión a escala de grises

El algoritmo utilizado para la conversión es el *Luminance*, puesto que presenta los mejores resultados en el reconocimiento de texturas de acuerdo a [13]. El valor de esta conversión está dado por la siguiente ecuación.

$$I_{Luminance} = 0{,}3R + 0{,}59G + 0{,}11B \quad (1)$$

4) Canny: detección de bordes

Los dos umbrales de Canny debido a la variabilidad de las condiciones de iluminación se calculan de forma automática a partir de la imagen. 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.

$$minVal = 0{,}67M_e \quad (2)$$

$$maxVal = 1{,}33M_e \quad (3)$$

Para el segundo algoritmo en cambio, ambos umbrales son determinados de acuerdo con el cálculo de umbralización

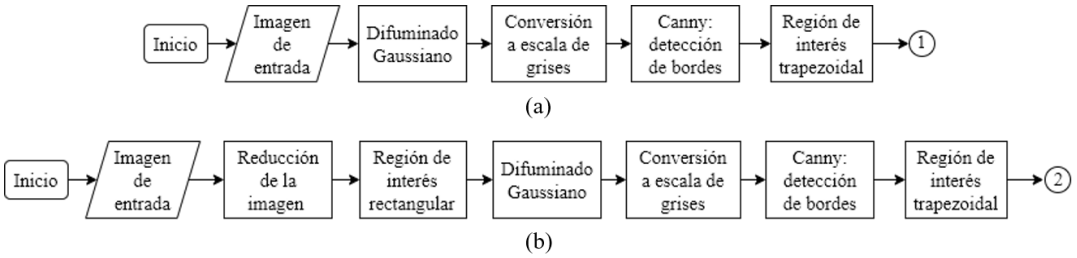

Fig. 3 Preprocesamiento de ambos algoritmos. (a) corresponde al primer algoritmo y (b) al segundo algoritmo.

propuesto por Otsu [14]. El valor de los umbrales se han fijado como sigue según lo recomendado en [15].

$$minVal = 0{,}5(Umbral\ de\ Otsu) \tag{4}$$

$$maxVal = Umbral\ de\ Otsu \tag{5}$$

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.

El resultado final de ambos preprocesamientos se puede observar en la Fig. 4.

B. 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. El diagrama de flujo completo de esta etapa se puede observar en la Fig. 6.

1) PPHT en conjunto con una regresión lineal simple

Para el reconocimiento de las líneas, en el conjunto de puntos de la imagen binaria de salida del preprocesamiento, se aplica el PPHT, donde en base a [16], y a pruebas se han definido sus hiperparámetros de entrada. 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. De esta manera 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.

No obstante, 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 establece 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.

$$m_{if} = \{m_i \mid k_{min} \leq m_i \leq k_{max}\} \tag{6}$$

$$\overline{m} = \frac{1}{N}\sum_{i}^{N} m_{if} \tag{7}$$

$$m_{final} = \left\{m_{if} \mid \overline{m} - T \leq m_{if} \leq \overline{m} + T\right\} \tag{8}$$

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 Fig. 5, donde en ciertas imágenes se comprueba la eficacia del algoritmo implementado, como también situaciones donde no logra detectar correctamente el carril.

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

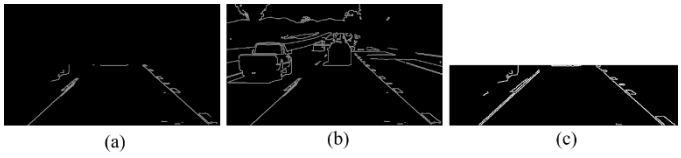

Fig. 4 Resultado del preprocesamiento. (a) ROI del primer algoritmo, (b) Imagen binaria de salida del Canny (c) ROI del segundo algoritmo.

Fig. 5 Resultado final del algoritmo PPHT en conjunto con una regresión lineal simple.

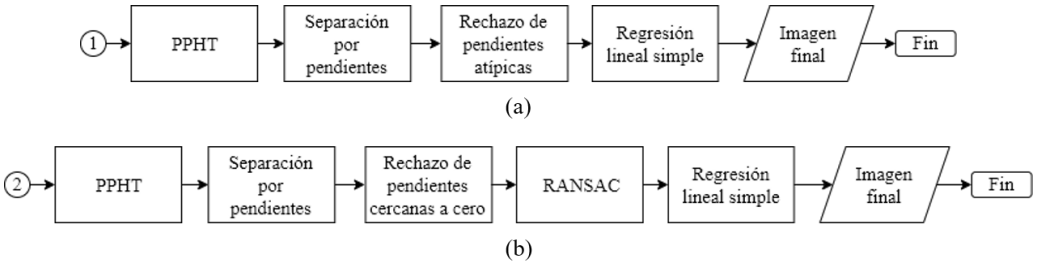

Fig. 6 Modelado de ambos algoritmos. (a) corresponde al primer algoritmo y (b) al segundo algoritmo.

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 algoritmo 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.

De la misma manera que en el primer algoritmo se aplica el PPHT, sin embargo, sus hiperparámetros son definidos con un valor menor 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 introduce más ruido a la imagen (entiéndase segmentos de línea que no constituyen una marca del carril).

La separación de las marcas del carril para ambos lados izquierdo y derecho, se da de manera parecida a lo explicado en la sección II-B-1. 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 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.

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.

$$m_{final} = \{m_i \mid |m_i| > 0{,}6\} \tag{9}$$

Al tener este conjunto de posibles marcas del carril, se aplica el algoritmo RANSAC para eliminar los outliers en los datos y encontrar el mejor modelo lineal. Para ello el RANSAC clasifica a los datos de manera iterativa entre *inliers* y *outliers* y modela a los *inliers* a través de una regresión lineal simple con ajuste por mínimos cuadrados. Como resultado final se tiene aquel modelo lineal con la mayor cantidad de *inliers* [17].

El resultado final de la ejecución del algoritmo en distintas imágenes se puede observar en la Fig. 7, donde se comprueba la eficacia del algoritmo y el efecto de la mejora realizada, tanto en la etapa de preprocesamiento como en la etapa de modelado.

C. *Simulación*

En la simulación se utiliza como medio de percepción del entorno a los dos algoritmos de detección de carriles explicados anteriormente. El objetivo es mantenerle al vehículo en su respectivo carril en un determinado circuito.

Para ello el control de dirección implementado en el vehículo es un control simple 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 un control Proporcional-Integral (PI) con un set point constante durante todo el recorrido. Cabe destacar que este PI no fue diseñado, simplemente fue implementado a partir de ejemplos proporcionados por los desarrolladores del simulador.

Lo escrito e implementado en esta sección están basados en la documentación del simulador [18] y en la tesis de maestría [19].

Para el cálculo del error lateral se asume que la cámara está situada en el centro geométrico del vehículo. De esta manera se calcula la distancia entre el centro del vehículo y el centro estimado del carril y la diferencia entre ambos constituye el error lateral. El diagrama de flujo de la simulación se observa en la Fig. 8.

III. Pruebas y Resultados

Para poder validar el desempeño de los algoritmos de forma cuantitativa, ambos algoritmos expuestos en la sección II,

Fig. 7 Resultado final del algoritmo PPHT en conjunto con RANSAC.

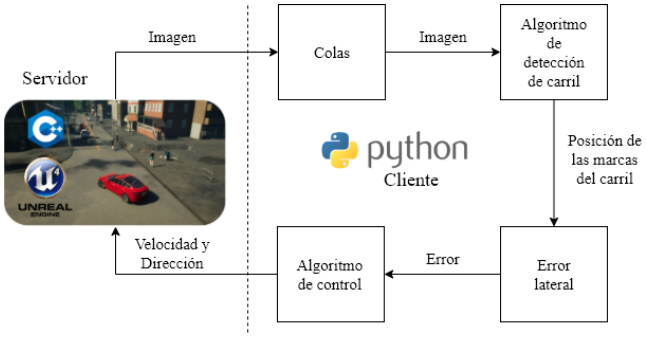

Fig. 8 Diagrama de flujo de la simulación.

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.

Para el cálculo de la exactitud (*Accuracy*) se utiliza la ecuación (10), donde c_{image} es el número de puntos correctamente estimados y s_{image} es el número de puntos anotados en la imagen.

$$Accuracy = \frac{\sum_{image} c_{image}}{\sum_{image} s_{image}} \quad (10)$$

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 se han modelado las marcas del carril con una recta, 60% de coincidencia o *Accuracy* de los puntos indica que se ha logrado detectar las marcas del carril satisfactoriamente y es el umbral que se ha utilizado para para la obtención de los resultados tanto de los falsos positivos y negativos.

Para esta evaluación solamente se consideraron las anotaciones y estimaciones que se encuentren dentro de la región de interés trapezoidal definida en la sección II-A-5, es decir, la exactitud de los algoritmos se cuantificará al estimar correctamente los carriles en la zona frontal próxima al vehículo. Lo anterior puede observarse en la Fig. 9 donde las circunferencias verdes son tomadas en cuenta en la evaluación, en cambio las de color rojo son descartadas.

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

Fig. 9 Filtrado de las anotaciones de los datasets y estimaciones.

A. *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 3626 imágenes en las cuales fue anotada la posición de los carriles, generando así los datos verdaderos del entorno (*Ground Truth*). Las imágenes del dataset fueron obtenidas en condiciones climáticas buenas y aceptables, en diferentes horarios del día y con diferentes situaciones del tráfico vehicular [20].

Los resultados obtenidos de cada algoritmo propuesto en este dataset se muestran en la Tabla I.

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.

B. *Dataset Hernandarias*

Para poder evaluar los algoritmos propuestos, en carriles de Paraguay y poner a prueba con un nuevo dataset a los algoritmos, 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 zona frontal al vehículo. Estas anotaciones fueron llevadas a cabo con la versión gratuita del software Labelbox.

Los resultados obtenidos de cada algoritmo propuesto en este dataset se muestran en la Tabla I.

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

TABLA I
MÉTRICAS DE LOS ALGORITMOS PROPUESTOS EN EL DATASET TUSIMPLE Y HERNANDARIAS

Dataset	Algoritmo	*Accuracy*	FPR	FNR	Tiempo de ejecución
TuSimple	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
Hernandarias	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

razonablemente bien para las pruebas en ese dataset, en un nuevo conjunto de imágenes como este se vuelve impracticable su utilización.

En relación a 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 inestable e 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 y 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.

C. Resultados CARLA

En la simulación se ha recabado a una frecuencia de 23 Hz el error lateral del vehículo. Estos 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 Fig. 10.

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.

Fig. 10 Circuito de la simulación.

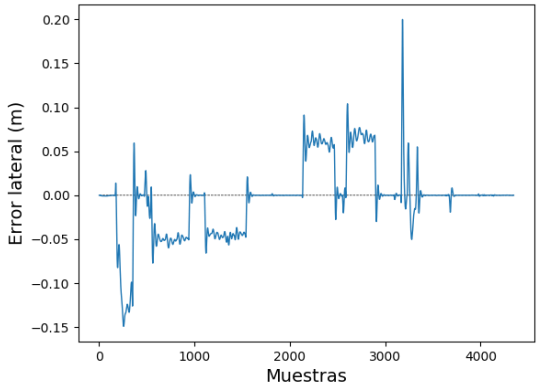

Fig. 11 Error lateral del vehículo controlado por el modo autónomo del servidor.

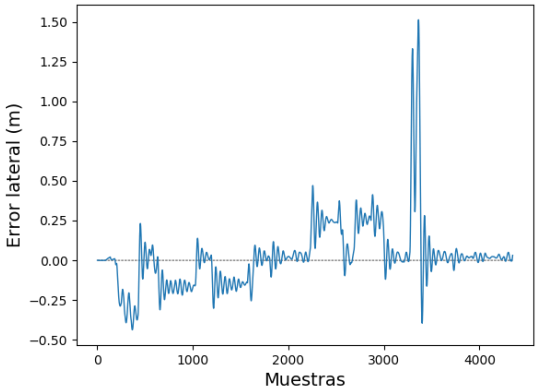

Fig. 12 Error lateral del vehículo controlado por el algoritmo PPHT, RANSAC.

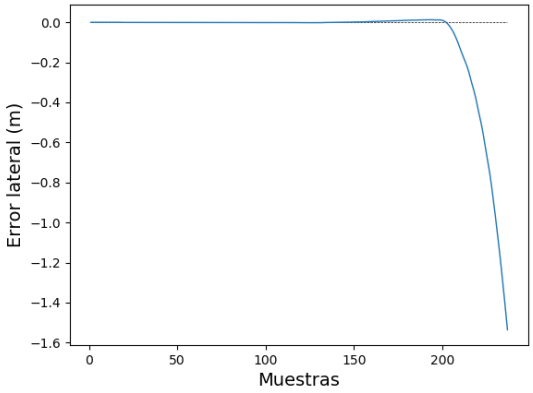

Fig. 13 Error lateral del vehículo controlado por el algoritmo PPHT.

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 la Fig. 11, Fig. 12 y Fig. 13. Cabe resaltar que las tres mediciones fueron hechas a una velocidad constante igual a 80km/h.

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 II.

TABLA II
MÉTRICAS DE LAS PRUEBAS DE CONTROL REALIZADAS EN EL SIMULADOR CARLA

Control	RMSE (m)	MAE (m)	Valor pico (m)
Servidor	0,041649	0,02573	0,19982
PPHT y Regresión lineal simple	*	*	*
PPHT, RANSAC y Regresión lineal simple	0,217497	0,12895	1,51253

*No ha completado el circuito

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 II 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, Fig. 12. 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 control predictivo basado en el modelo (MPC, por sus siglas en inglés). 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 visión computacional que utiliza PPHT y RANSAC, puesto que ha culminado el circuito de forma exitosa sin abandonar su carril.

IV. CONCLUSIONES

Se evidenció, cierta fragilidad de los algoritmos desarrollados con cambios bruscos de luminosidad, como también la robustez introducida por los algoritmos RANSAC y el método de Otsu al segundo algoritmo propuesto, debido a su desempeño superior en todas las evaluaciones.

Se comprobó visualmente que, para la región frontal lejana al vehículo, principalmente en las curvas, el modelo lineal 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 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, a través del segundo algoritmo.

Se sugieren como trabajos futuros:

- 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.

V. BIBLIOGRAFÍA

[1] Y. Xing *et al.*, "Advances in Vision-Based Lane Detection: Algorithms, Integration, Assessment, and Perspectives on ACP-Based Parallel Vision," *IEEE/CAA J. Autom. Sin.*, vol. 5, no. 3, pp. 645–661, 2018, doi: 10.1109/JAS.2018.7511063.

[2] U. Suddamalla, S. Kundu, S. Farkade, and A. Das, "A novel algorithm of lane detection addressing varied scenarios of curved and dashed lanemarks," in *5th International Conference on Image Processing, Theory, Tools and Applications 2015, IPTA 2015*, 2015, pp. 87–92, doi: 10.1109/IPTA.2015.7367103.

[3] J. C. McCall, D. P. Wipf, M. M. Trivedi, and B. D. Rao, "Lane change intent analysis using robust operators and sparse Bayesian learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 3, pp. 431–440, 2007, doi: 10.1109/TITS.2007.902640.

[4] D. J. Kang and M. H. Jung, "Road lane segmentation using dynamic programming for active safety vehicles," *Pattern Recognit. Lett.*, vol. 24, no. 16, pp. 3177–3185, 2003, doi: 10.1016/j.patrec.2003.08.003.

[5] Q. Li, N. Zheng, and H. Cheng, "Springrobot: A prototype autonomous vehicle and its algorithms for lane detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 4, pp. 300–308, 2004, doi: 10.1109/TITS.2004.838220.

[6] M. Aly, "Real time detection of lane markers in urban streets," in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2008, pp. 7–12, doi: 10.1109/IVS.2008.4621152.

[7] H. Tan, Y. Zhou, Y. Zhu, D. Yao, and K. Li, "A novel curve lane detection based on Improved River Flow and RANSA," in *2014 17th IEEE International Conference on Intelligent Transportation Systems, ITSC 2014*, 2014, pp. 133–138, doi: 10.1109/ITSC.2014.6957679.

[8] Z. W. Kim, "Robust lane detection and tracking in challenging scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 1, pp. 16–26, 2008, doi: 10.1109/TITS.2007.908582.

[9] A. Borkar, M. Hayes, and M. T. Smith, "Robust lane detection and tracking with Ransac and Kalman filter," in *Proceedings - International Conference on Image Processing, ICIP*, 2009, pp. 3261–3264, doi: 10.1109/ICIP.2009.5413980.

[10] J. Li, X. Mei, D. Prokhorov, and D. Tao, "Deep Neural Network for Structural Prediction and Lane Detection in Traffic Scene," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 28, no. 3, pp. 690–703, 2017, doi: 10.1109/TNNLS.2016.2522428.

[11] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: Spatial CNN for traffic scene understanding," in *32nd AAAI Conference on*

Artificial Intelligence, AAAI 2018, 2018, pp. 7276–7283.

[12] L. Desegur, “A Lane Detection Approach for Self-Driving Vehicles,” 2018. https://github.com/ldesegur/CarND-LaneLines-P1.

[13] C. Kanan and G. W. Cottrell, “Color-to-grayscale: Does the method matter in image recognition?,” *PLoS One*, vol. 7, no. 1, pp. 1–7, 2012, doi: 10.1371/journal.pone.0029740.

[14] N. Otsu, “A Tlreshold Selection Method from Gray-Level Histograms,” *IEEE Trans. Syst. Man. Cybern.*, vol. 9, no. 1, pp. 62–66, 1979, doi: 10.1109/TSMC.1979.4310076.

[15] M. Fang, G. Yue, and Q. Yu, “The study on an application of otsu method in canny operator,” in *Proceedings of the 2009 International Symposium on Information Processing (ISIP'09)*, 2009, pp. 109–112, [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:The+Study+on+An+Application+of+Otsu+Method+in+Canny+Operator#0.

[16] J. Matas, C. Galambos, and J. Kittler, “Robust detection of lines using the progressive probabilistic hough transform,” *Comput. Vis. Image Underst.*, vol. 78, no. 1, pp. 119–137, 2000, doi: 10.1006/cviu.1999.0831.

[17] D. A. Forsyth and J. Ponce, *Computer Vision a Modern Approach*, 2nd ed. New Yersey: Pearson, 2012.

[18] CARLA Team, “CARLA Documentation.” https://carla.readthedocs.io/en/latest/ .

[19] M. Cattaruzza, “Design and Simulation of Autonomous Driving Algorithms (Tesis de Maestría),” Politecnico Di Torino, 2019.

[20] I. TuSimple, “TuSimple Lane Detection Challenge,” 2017. https://github.com/TuSimple/tusimplebenchmark/tree/master/doc/lane_detection .