Proyecto de Fin de Grado (Ingeniería de Computadores)

# Diseño e implementación de Seguimiento de Carril de Robocar

RUBÉN HIGUERA CASTILLO
2023/2024

# 1 Contenido

# 2 Introducción

## 2.1 Contexto y Justificación

Una definición de conducción autónoma podría ser[1]: *"Un automóvil autónomo es un vehículo capaz de detectar su entorno y operar sin la participación humana. No se requiere que un pasajero humano tome el control del vehículo en ningún momento, ni se requiere que un pasajero humano esté presente en el vehículo en absoluto. Un automóvil autónomo puede ir a cualquier lugar al que vaya un automóvil tradicional y hacer todo lo que hace un conductor humano experimentado."*

Según los optimistas [1] predicen que para 2030, los vehículos autónomos serán lo suficientemente fiables, asequibles y comunes para desplazar la mayor parte de la conducción humana, proporcionando enormes ahorros y beneficios. Sin embargo, son buenas las razones para ser escépticos. La mayoría de las predicciones optimistas son hechas por personas con intereses en la industria o basados en la experiencia con tecnologías disruptivas como la digital cámaras, teléfonos inteligentes y computadoras personales. Tienden a ignorar los obstáculos significativos para el desarrollo de vehículos autónomos, y a exagerar los beneficios futuros.

Se alcance o no la conducción autónoma, la realidad es que se van a producir (ya lo están haciendo) grandes avances gracias sensorización (ver figura). Hoy en día esto ya lo podemos experimentar con los llamados ADAs, los asistentes a la conducción que nos ayudan a aparcar o nos avisan de la salida de carril entre otras muchas cosas.

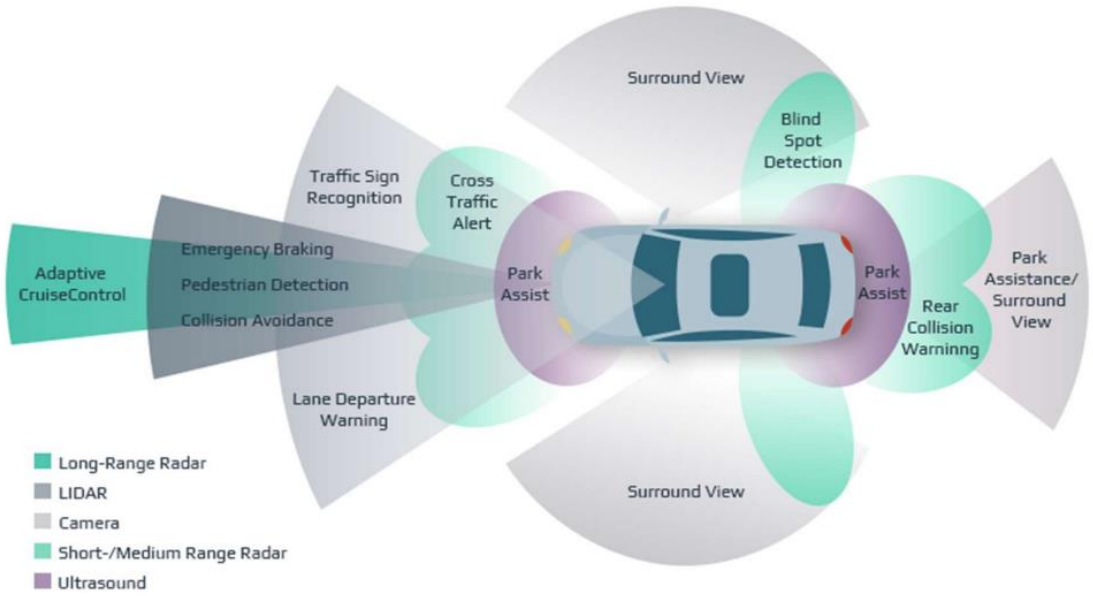


*Figura 1: Sensórica de coche autónomo [2]*

[1] https://www.synopsys.com/

La *Society of Automotive Engineers* (SAE) actualmente define 6 niveles de automatización de conducción que van desde el Nivel 0 (totalmente manual) hasta el Nivel 5 (totalmente autónomo). La SAE utiliza el término automatizado en lugar de autónomo. Una razón es que la palabra autonomía tiene implicaciones más allá de lo electromecánico. Un automóvil autónomo sería consciente de sí mismo y capaz de tomar sus decisiones. Por ejemplo, dices "llévame al trabajo", pero el auto decide llevarte a la playa. Sin embargo, un automóvil completamente automatizado seguiría las órdenes y luego se conduciría solo.

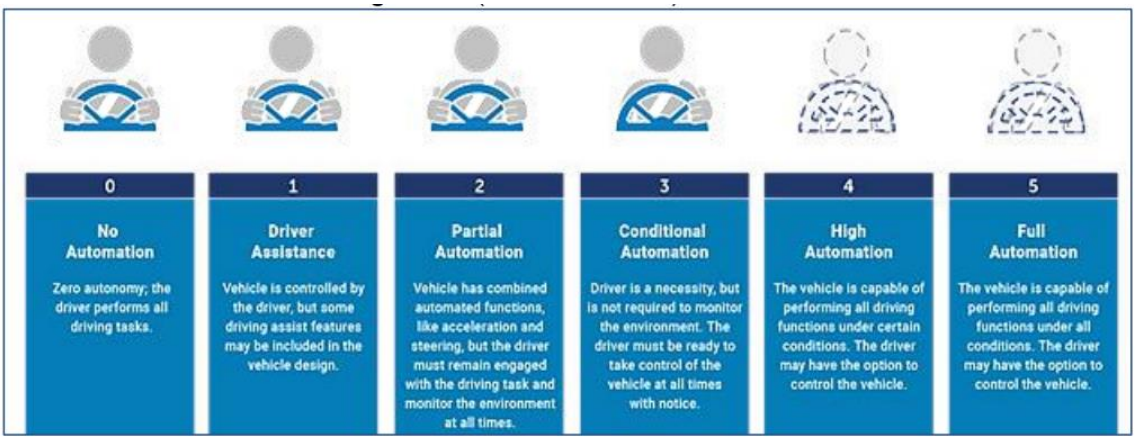


*Figura 2: Niveles de autonomía*

- Nivel 0: No hay automatización en la conducción
- Nivel 1: Asistencia a la conducción. En el nivel uno, el vehículo tiene asistencia al conductor, pero todavía es necesario que el conductor realice las tareas principales como acelerar, frenar y monitorear. El vehículo puede ayudar con funciones simples, por ejemplo, aplicar un poco más el freno cuando se acerca demasiado a otro automóvil en la carretera.
- Nivel 2: Conducción automática parcial. El Nivel Dos, también conocido como Automatización Parcial, funciones de dirección, aceleración asistida por el vehículo. Los conductores se desvinculan de algunas de sus tareas, pero deben estar listos para tomar el control del vehículo en situaciones de emergencia.
- Nivel 3: Conducción automática condicional. En el Nivel Tres (Automatización Condicional), la mejora más importante es que el vehículo comenzó a usar sensores (LIDAR, Radar, etc.) para controlar todo el monitoreo de su entorno. Todavía se necesita un conductor, pero en funciones críticas de seguridad, el vehículo puede responder por sí mismo.
- Nivel 4: Alta automatización de la conducción. En el Nivel Cuatro (Alta Automatización), además de otras características de los niveles, el vehículo puede determinar cuándo cambiar de carril, girar y usar las señales. Pero no puede determinar la congestión del tráfico ni incorporarse a la autopista.
- Nivel 5: Automatización total de la conducción. El Nivel Cinco es la Automatización Completa. No es necesario que un conductor humano controle ninguna tarea del vehículo. Por lo tanto, no hay volante, pedales ni frenos. Todas las tareas son manejadas por el propio vehículo.

Es claro que es una tarea compleja en la medida que la realidad del entorno de conducción es también compleja. Para acometer esta ingente tarea muchos has dibujado mapas funcionales de todas las tareas que debería un AV llevar a cabo. Una arquitectura es la que propones en [2]

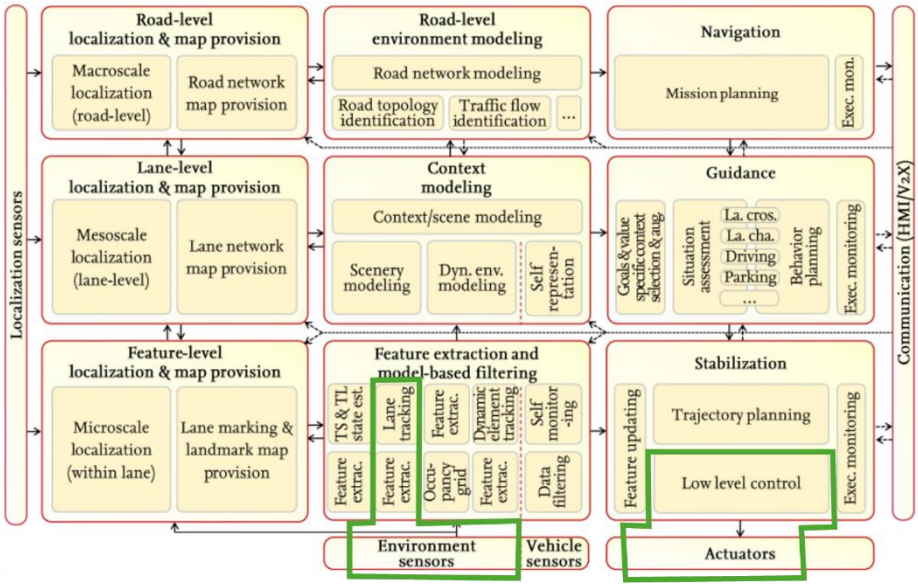


*Figura 3: Arquitectura vehículo autónomo*

En este trabajo, nos centraremos en parte de los siguientes elementos funcionales (los marcados en verde en la figura):

- "Feature extraction and model-based filtering", para realizar un seguimiento de carril en base a sensores
- "Stabilization", para realizar un control que nos permita mantener el robot en el carril

Este trabajo está basado en otros muchos estudios, en especial son destacables los siguientes:

- Para la parte más teórica y descriptiva de las tecnologías existentes: "Evaluation of Lane Detection Algorithms based on an Embedded Platform" [1] es un Máster Thesis de la Universidad Técnica de Chemnitz
- Para la parte de implementación el repositorio https://github.com/emirelesg/Self-Driving-Vehicle/tree/master

## 2.2 Objetivos del Proyecto (Requerimientos de alto nivel)

Las dos partes más importantes de los sistemas avanzados de asistencia al conductor (ADAs) son un sistema de evasión de colisiones y un sistema de asistencia de mantenimiento de carril, los cuales podrían ayudar a reducir el número de accidentes de tráfico.

Una técnica fundamental para la evasión efectiva de colisiones y el mantenimiento de carril es un método robusto de detección de carriles. Especialmente, ese método debería detectar un carril recto o curvo en el campo lejano de visión.

Un automóvil que se desplaza a una velocidad determinada necesitará cierto tiempo para detenerse o reducir la velocidad manteniendo la estabilidad.

Esto significa que es necesario detectar el carril de la carretera tanto en el campo cercano como en el campo lejano de visión.

Aquí divide el campo de visión en cercano y lejano, a velocidad constante baja el lejano no tiene sentido, por tanto, habrá que tener en cuenta solo el cercano, esto lógicamente tiene que ver

con la posición de la cámara, igual no tiene mucho sentido tenerla muy elevada y en el caso de que sea así debe tener un ángulo suficiente para enfocar la parte cercana por delante del robot.

En [22], McCall y Trivedi describen tres objetivos principales de los algoritmos de detección de carril:

- **Sistema de advertencia de cambio de carril**: Basado en la información sobre los laterales desviación del vehículo al centro del carril, velocidad, etc., tenemos que predecir la trayectoria del vehículo: si se está saliendo del carril o no y dónde el punto de cruce es.
- **Sistema de Atención al Conductor**: En la tarea de monitoreo, hay muchos factores que deben tenerse en cuenta para gestionar la atención del conductor. Un elemento importante es la información que pueden proporcionar los algoritmos de detección de carril es la suavidad en mantenimiento de carril.
- **Sistema Automatizado de Control de Vehículos**: Este es el más complicado en los tres objetivos y recientemente ha recibido una considerable atención por parte de investigadores de todo el mundo. Requiere de los algoritmos de detección de carril el Desviación lateral precisa en el momento y predicciones en distancia por delante.

Este TFG recoge el resultado de otro TFG titulado “Diseño y Construcción de Robocar” cuyo resultado es un robot físico dotado de sensores y controlado mediante el software ROS2.

El objetivo es que Robocar pueda hacer un seguimiento de carril bajo los supuestos y requisitos que más adelante se detallan implementando las tecnologías que actualmente se están desarrollando para los sistemas de asistencia a la conducción (ADAs) y coches autónomos.

## 2.3 Estructura del Documento

Este documento pretende realizar el camino desde lo general, es decir, encuadrar el trabajo dentro de las tecnologías que se están desarrollando en el sector automovilístico para proporcionar asistencia a la conducción (ADA) hasta una implementación de un sistema de seguimiento de carril particular sobre el robot Robocar. Este camino se recorre a través de los siguientes capítulos:

- **Introducción**: dónde se encuadra este trabajo (el seguimiento de carril y el control de la dirección) en el universo de funciones disponibles en un coche autónomo.
- **Marco Teórico**: qué tecnologías hay disponibles
- **Tecnologías Aplicadas**: cuáles de las tecnologías que se recogen en el Marco Teórico se seleccionan para la implementación
- **Modelo de Sistema**: cómo parametrizamos el sistema físico (robot y entorno) para poder aplicar las tecnologías seleccionadas, dicho de otra forma, cómo llevamos el sistema físico a uno virtual
- **Implementación**: se implementan los algoritmos necesarios para que el robot pueda realizar un seguimiento de carril con las condiciones modeladas en el capítulo anterior.
- **Resultados**: qué conclusiones se pueden obtener del camino recorrido

# 3 Marco Teórico

En este capítulo se describen las diferentes formas de realizar un seguimiento de carril. La siguiente figura ilustra en grandes bloques el proceso en bucle cerrado. En este capítulo se resumen en cada bloque implementa diferentes tecnologías. Más adelante se seleccionarán algunas de ellas para la implementación.

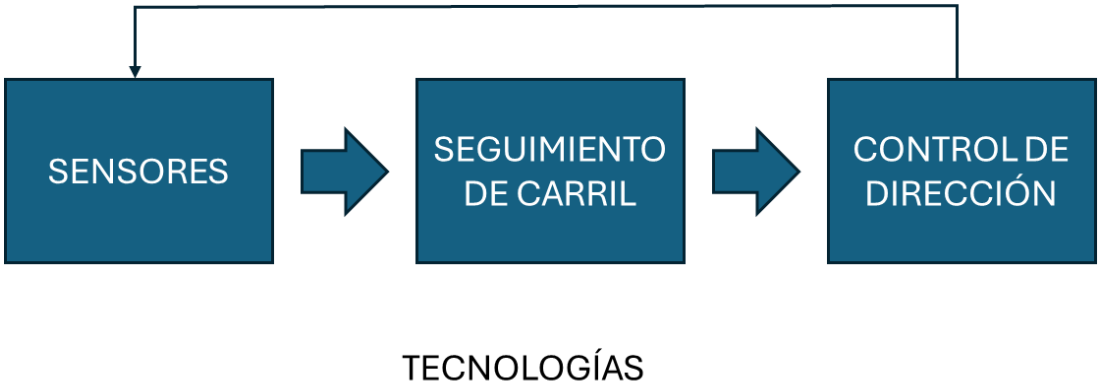


*Figura 4: Proceso seguimiento de carril*

## 3.1 Sensores

Se han usado sensores diferentes para dar entradas a los algoritmos de detección de carriles, algunos son cámaras, Sistema de Posicionamiento Global (GPS), Radio Sensor de detección y alcance (RADAR), detección y alcance de luz (LIDAR), etc. Cada tipo de sensor proporciona información diferente, como visión, distancia y posición. Es por eso por lo que un sensor puede funcionar bien en unas condiciones, pero fracasar en otras. La siguiente tabla muestra las ventajas y desventajas de cada tipo de sensor:

| Sensores | Ventajas | Desventajas |
|---|---|---|
| **Sensores de visión** | Funcionan bien en una gran cantidad de escenarios sin la ayuda de mapas o infraestructuras externas. | No rinden bien bajo condiciones lumínicas bajas o donde los carriles no estén bien definidos. |
| **Sensores de estado interno del vehículo** | Sensores secundarios que dan soporte con datos adicionales para otro tipo de sensores. | Tiene que colaborar con otros sensores (no son independientes). |
| **Sensores de líneas** | Pueden dar una medición muy precisa de la desviación con respecto a los carriles. | No pueden proporcionar información acerca de curvas a distancia media/lejana. |
| **LIDAR y RADAR** | Especializados en detectar objetos y calcular distancias. Especialmente útiles en zonas donde la carretera no está bien definida, como en áreas rurales. | No rinden bien para detección de líneas, por lo que tienen que trabajar junto con otros sensores. |
| **GPS y mapas digitales** | Proporciona la ubicación del vehículo con una precisión con 10cm de error. Muy útil | Necesita de infraestructura externa para conseguir buena precisión. Además, los |

| | | |
|---|---|---|
| | para ver localización del vehículo. | mapas digitales deben ser actualizados. |

## 3.2 Sistemas de Seguimiento de Carril (revisión de literatura)

Este capítulo trata sobre la base de diferentes publicaciones de revisar los métodos para el seguimiento de carriles, y posteriormente se seleccionarán los que se implementarán en este trabajo según criterios de factibilidad, eficiencia y física del robot.

Como en otros entornos tecnológicos la primera gran división está basada en la utilización de leyes físicas que gobiernan los sistemas o no, simplemente basados en los datos obtenidos para entrenar un algoritmo y con esto modelar el sistema (aprendizaje automático)

Los modelos basados en la física de los sistemas a menudo se consideran más precisos y confiables, ya que se basan en leyes físicas y ecuaciones matemáticas bien definidas. Sin embargo, los modelos basados en datos (como el aprendizaje automático) han ganado popularidad debido a su capacidad para adaptarse a sistemas complejos y no lineales, y a su velocidad de ejecución.

En algunos casos, una combinación de ambos enfoques (modelos físicos y datos) puede ser la mejor solución, ya que los datos pueden utilizarse para entrenar y mejorar los modelos físicos.

No existe una solución única, sino que la elección dependerá de las características específicas del problema que se esté abordando.

En base a lo anterior podemos categorizar los diferentes enfoques en los siguientes:

- Enfoque basado en características (detección y seguimiento de carriles basado en imagen y sensor)
- Enfoque basado en modelos (detección y seguimiento robusto de carriles)
- Enfoque basado en el aprendizaje (detección y seguimiento de carriles con controlador predictivo)

El enfoque basado en características utiliza características visuales locales como bordes, color, brillo, textura, etc., que son relativamente insensibles a las formas de la carretera, pero sensibles a los efectos de iluminación. El enfoque basado en modelos aplica modelos globales de carretera para ajustar características de niveles bajos más robustas contra los efectos de iluminación, pero sensibles a las formas de la carretera.

McCall y Trivedi [3] presentan un estudio sobre la estimación y seguimiento de carriles basada en sensores de tipo cámara para la asistencia al conductor (ADA). Proponen (ver figura) una serie de funciones genéricas que todos los métodos basados en la física implementan.

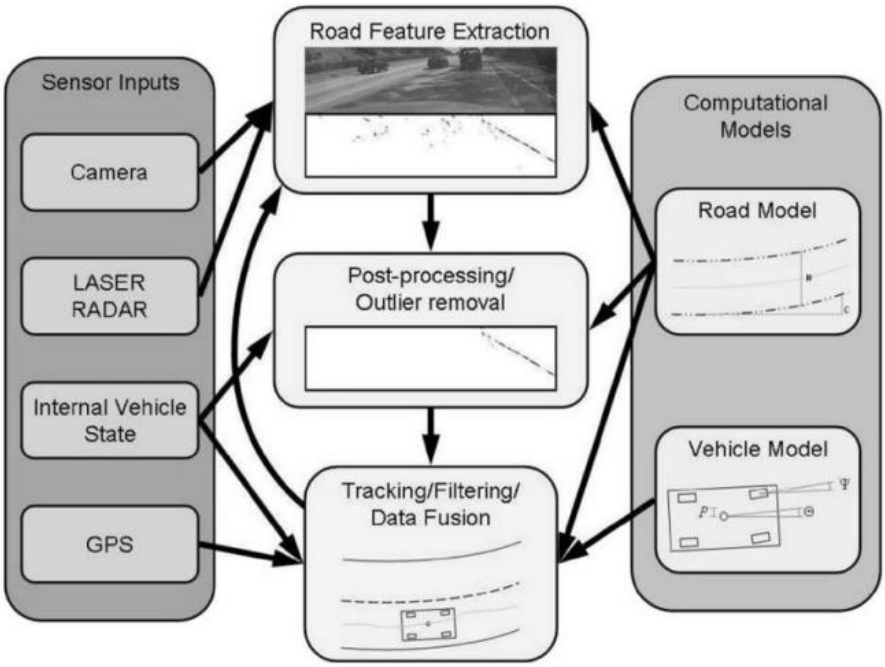


*Figura 5: McCall and Trivedi*

1. En primer lugar, se utiliza un sistema de sensores de entrada para recoger la información del entorno. Hay varios tipos de sensores que se pueden utilizar y cada uno proporciona un tipo diferente de datos, por ejemplo, la cámara, el tipo más popular, o el GPS o LIDAR.
2. A continuación, se propone un **modelo de carretera** o de señalización de carriles. Puede ser una simple línea recta o un modelo más complejo.
3. Los datos de entrada se procesan con ayuda del modelo de carretera propuesto para extraer la información sobre las características de la carretera, como bordes, texturas.
4. Dependiendo de los escenarios con los que estemos tratando, se puede utilizar una aproximación lineal o un modelo de movimiento de dirección real para mejorar la estimación de la etapa de seguimiento
5. Por último, puede aplicarse una etapa de seguimiento para ayudar en la predicción y el suavizado. Las dos técnicas más populares utilizadas en esta etapa son el Filtro de Kalman y el Filtro de Partículas.

Waykole et al. [4] realizan un análisis exhaustivo de los algoritmos de detección y seguimiento de carriles en los sistemas ADAS:

**Enfoque Basado en Características**

- Detección y Seguimiento de Carriles Basados en Imágenes y Sensores
- Método: Utiliza bordes, gradiente, color, brillo, textura, orientación y variaciones.
- Ejemplo: Se implementó un sistema de mantenimiento de carril basado en visión que utiliza la salida de la cámara y los valores de los sensores para seguir el carril.

**Enfoque Basado en Modelos**

- Detección y Seguimiento de Carriles Robustos
- Método: Aplica modelos globales de carretera para ajustar características que son más robustas contra efectos de iluminación, pero sensibles a las formas de la carretera.
- Ejemplo: Lee y Moon [5] propusieron un sistema para detectar y seguir carriles bajo diversas condiciones ambientales utilizando tres fases: inicialización, detección de carril y seguimiento de carril.

**Enfoque Basado en Aprendizaje**

- Controlador de Cambio de Carril Basado en Aprendizaje por Refuerzo
- Método: Consiste en dos etapas: entrenamiento y clasificación.
- Ejemplo: Wang et al. [6] propusieron un controlador de cambio de carril basado en aprendizaje por refuerzo utilizando control longitudinal y lateral con un aproximador de la función Q para un espacio de acción continuo.

| Método | Pasos | Herramientas | Datos | Tipo de Método | Información |
|---|---|---|---|---|---|
| Detección y Seguimiento de Carriles Basados en Imágenes y Sensores | a. Las imágenes son preprocesadas<br>b. Se aplica algoritmo de detección de carril<br>c. Los valores de los sensores se usan para el seguimiento | a. Cámara<br>b. Sensores | Valores de los sensores | Basado en características | Se requiere una calibración frecuente |
| Controlador predictivo para detección de carril | Machine Learning | a. Modelo de controlador predictivo<br>b. Algoritmo de Reinforcement Learning | Datos obtenidos del controlador | Basado en modelo de aprendizaje | Es de las mejores opciones para evitar detecciones de carriles falsas |
| Detección y seguimiento de carril robusta | a. Capturar una imagen desde la cámara<br>b. Uso de detector de bordes para extraer características de la imagen<br>c. Determinación del punto de fuga | Basado en algoritmos de detección robusta de carril | Tiempo real | Basada en modelo | Proporciona mejores resultados en diferentes condiciones ambientales. La cámara juega un papel fundamental en la detección del carril |

*Tabla 1: Métodos de detección de carriles*

Las conclusiones que alcanza en este estudio son las siguientes:

- Se requiere una calibración frecuente para una toma de decisiones precisa en un entorno complejo.

- El aprendizaje por refuerzo con el modelo de control predictivo podría ser una mejor opción para evitar la detección de carriles falsos.

- Los enfoques basados en modelos (detección y seguimiento de carriles robustos) proporcionan mejores resultados en diferentes condiciones ambientales.

- La calidad de la cámara juega un papel importante en la determinación de la señalización del carril.

- El rendimiento del algoritmo depende del filtro utilizado, y el filtro de Kalman se usa para rastrear carriles.

- En un sistema basado en visión, el suavizado de imágenes es la etapa inicial de detección y seguimiento de carriles, que desempeña un papel vital en el aumento del rendimiento de los sistemas.

- Las perturbaciones externas, como las condiciones climáticas, la calidad de la visión, las sombras y los resplandores, y las perturbaciones internas, como las marcas de carril demasiado estrechas, demasiado anchas y poco claras, disminuyen el rendimiento del algoritmo.

- La mayoría de los investigadores (>90%) han utilizado conjuntos de datos Se requiere una calibración frecuente para una toma de decisiones precisa en un entorno complejo.

- Se han utilizado cámaras monoculares, estéreo e infrarrojas para capturar imágenes y videos.

- La precisión del algoritmo depende del tipo de cámara utilizada, y una cámara estéreo ofrece un mejor rendimiento que una cámara monocular.

- Las marcas de carril pueden ser ocluidas por un vehículo cercano mientras se realiza un adelantamiento.

- Hay un cambio brusco en la iluminación cuando el vehículo sale de un túnel. Los cambios bruscos de iluminación afectan a la calidad de la imagen y disminuyen el rendimiento del sistema.

- Los resultados muestran que la tasa de eficiencia de detección y seguimiento de carriles en condiciones de lluvia seca y ligera es cercana al 99% en la mayoría de los escenarios.

- Sin embargo, la eficiencia de la detección de marcas de carril se ve significativamente afectada por las condiciones de lluvia intensa.

- Se ha visto que el rendimiento del sistema disminuye debido a las marcas de carril poco claras y degradadas.

- La IMU (unidad de medición de inercia) y el GPS son ejemplos que ayudan a mejorar el rendimiento de la medición de distancias de RADAR y LIDAR.

- Uno de los mayores problemas de los ADAS actuales es que los cambios en el entorno y en las condiciones ambientales tienen un efecto negativo en el rendimiento del sistema de detección de carriles.

H. Saleem et al. (Steering Angle Prediction Techniques, IEEE) [7] realiza una extrordinaria taxonomía de los métodos y tecnologías utilizadas para realizar un seguimiento de carril.

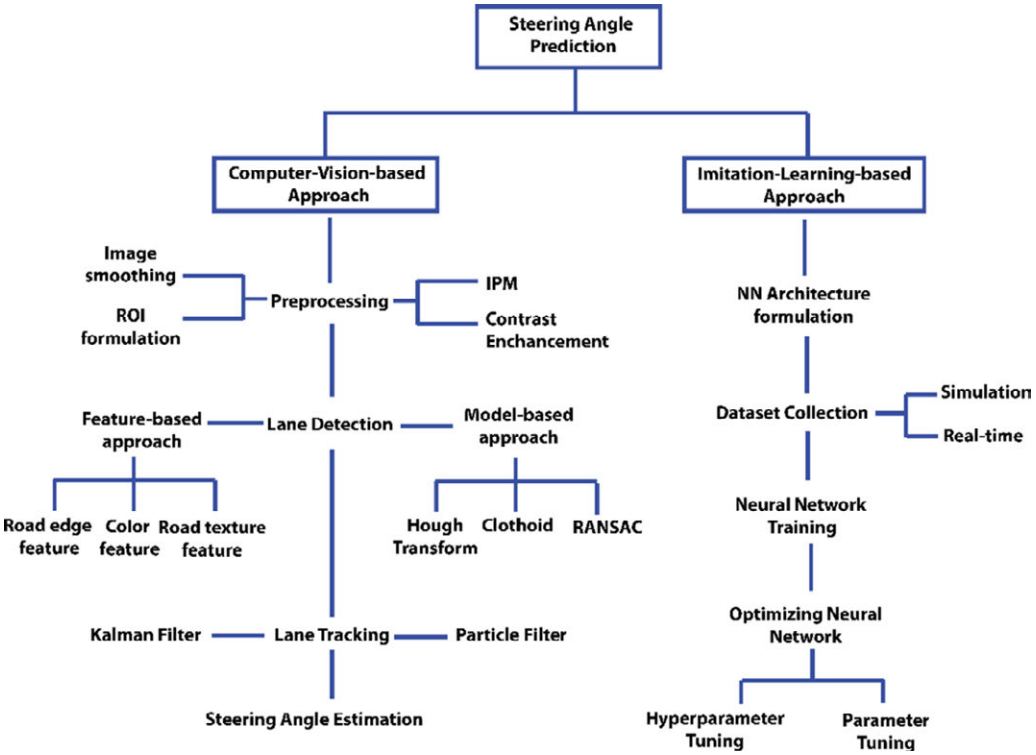


*Figura 6: Taxonomía de seguimiento de carril*

**Modelo de carril**

La velocidad de los vehículos (en nuestro caso, un robot) es importante en la detección y, sobre todo, en el seguimiento de carril, los estudios distinguen entre el campo cercano y lejano.

En el campo cercano se pueden hacer aproximaciones de carriles lineales (decisiones a corto plazo) mientras que en el campo lejano se hace una aproximación curva que permita tomar decisiones a futuro como bajar la velocidad si es necesario.

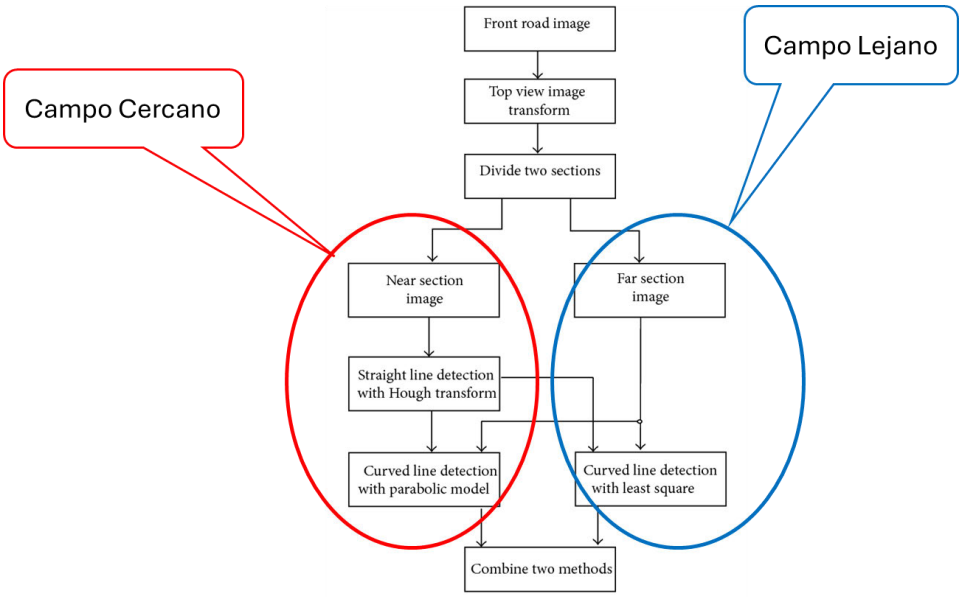


*Figura 7: Métodos de detección de carril*

Por las ventajas y desventajas de cada modelo de carril que se describen en las investigaciones anteriores, es difícil decir qué modelo es el mejor para todos los casos. La elección del modelo de carril depende de la situación a la que tenga que hacer frente el sistema de detección de posición de carril. Por ejemplo, las carreteras de carretera normalmente tienen una estructura bastante simple, por lo tanto, es más razonable usar modelos simples. Además, si el vehículo se mueve a baja velocidad, el sistema de seguridad puede requerir solo unos 10 m de la carretera por delante, por lo que un modelo de carril lineal puede satisfacer los requisitos en este caso. Pero considerando una situación más compleja, para el sistema como el Sistema de Advertencia de Salida de Carril o el Sistema de Control Automatizado de Vehículos, es necesario predecir con precisión la trayectoria del vehículo en los próximos al menos unos segundos. En la situación de la carretera, el vehículo se mueve a alta velocidad, la preocupación debe estar al menos 30/40 m por delante. En este caso, sería esencial un modelo más complejo como la curva parabólica.

Según las características de nuestro sistema, se suele definir una región de interés (ROI) dentro de cada imagen a analizar, así se focaliza el esfuerzo de análisis sobre la parte más relevante de la imagen.

- **Ubicación:** La ROI se establece en la parte inferior y central de la imagen, donde se espera que aparezcan las líneas del carril frente a la carretera.
- **Forma**: A menudo, la ROI tiene una forma trapezoidal. La base más ancha del trapezoide se encuentra en la parte inferior de la imagen (cerca del vehículo) y se estrecha hacia la parte superior (lejos del vehículo), imitando la perspectiva de la carretera que se aleja en la distancia.
- **Tamaño:** La ROI debe ser tan grande como para capturar la anchura de los carriles y suficiente longitud para detectar y rastrear las líneas del carril. Es obvio que para vehículos con velocidad baja la ROI puede ser más pequeña, este va a ser nuestro caso como veremos más adelante.

**Extracción de características**

Esta etapa es básica, la más importante, dado que el resultado final de la detección y seguimiento de carril dependerá de ella. El método de extracción de características variará dependiendo de las condiciones de del carril. Las técnicas basadas en detección de los bordes (función Canny de OpenCV) del carril son las más populares especialmente cuando las marcas de los carriles están bien definidas, otras técnicas relevantes están basadas en la detección del color, brillo, textura, etc.

**Post-procesamiento**

Basándose en el conocimiento sobre las marcas de carril que obtenemos después de la etapa de extracción de características del carril, junto con el modelo de carretera propuesto, es necesario realizar una etapa de post-procesamiento para estimar las posiciones de las marcas de carril. Las técnicas más comunes que se utilizan en esta etapa son la Transformada de Hough y el Consenso de Muestra Aleatoria (RANSAC). El más usado es la Transformada de Hough.

**Seguimiento de carril**

Esta etapa no siempre se realiza, no obstante, en general, el rendimiento aumenta si se implementa. Normalmente, los algoritmos de seguimiento se usan junto con la extracción de características del carril para formar un sistema de retroalimentación en bucle cerrado. Esta etapa nos proporciona las siguientes funciones

- En sistemas donde la detección del carril no es muy fiable genera una estimación del carril, en carreteras muy sinuosas esto podría ser un inconveniente.
- "Suaviza" los cambios en la detección de la posición del vehículo con respecto al carril derivados de una mala detección

Los algoritmos de seguimiento de carril más usados son el Filtro de Kalman y el Filtro de Partículas. Entre estos el más popular es el filtro de Kalman.

## 3.3 Control de la dirección del robot

[7] Después de realizar el proceso de extracción de límites de carril, hay que realizar el cálculo del ángulo de dirección. Se han utilizado diferentes técnicas propuestas por los investigadores para este fin. Los métodos que se describen aquí tienen como resultado hacia dónde se tiene que dirigir el vehículo, luego dependiendo del tipo de mecanismo (en el caso de Robocar es de tipo Ackermann la habitual en los coches) de dirección tenga habrá que aplicar un control determinado.

- Punto de Fuga (intersección de los bordes del carril) propuesta por [8] Yang et al.

Este método se basa en encontrar el Punto de Intersección (POI) de las líneas de frontera del carril detectadas en una imagen. El procedimiento es:

- Detección de Líneas de Frontera: Utiliza técnicas de detección de bordes y transformada de Hough para identificar las líneas de los carriles izquierdo y derecho.
- Cálculo del Punto de Intersección (POI): El punto de intersección de las dos líneas detectadas se calcula y se toma como referencia.

- Determinación del Ángulo: Si el POI está alineado verticalmente con el centro de la imagen, el ángulo de dirección es cero, indicando que el vehículo está centrado. Si no, se dibuja una línea desde el POI hasta el centro inferior de la imagen. El ángulo entre esta línea y la línea vertical que pasa por el centro de la imagen se utiliza como el ángulo de dirección requerido.

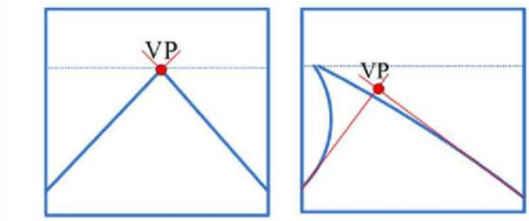


*Figura 8: Determinación del ángulo de curvatura*

Se trata de un método sencillo de implementar y efectivo en situaciones donde las líneas de carril son claramente visibles y rectas. Es menos robusto en carreteras con curvas pronunciadas o condiciones de iluminación variables.

- **Promedio Lineal de Líneas Detectadas** (propuesta por Tu et al. [9]):

Esta técnica aborda situaciones en las que puede haber detección parcial de las líneas de carril (es decir, solo una línea detectada o ninguna). El procedimiento es:

- Detección de Líneas de Frontera: Detecta las líneas de los carriles utilizando técnicas de procesamiento de imágenes.
- Cálculo del Promedio:
  - Una Línea Detectada: Si solo se detecta una línea (izquierda o derecha), se utiliza esta línea para calcular el ángulo de dirección. Se mide el ángulo entre la línea detectada y una línea vertical central en la imagen.
  - Ambas Líneas Detectadas: Si se detectan ambas líneas, se calcula el promedio lineal de las dos líneas. El ángulo de dirección es el ángulo entre esta línea promedio y la línea vertical central.
  - Ninguna Línea Detectada: Si no se detectan líneas, se utiliza el ángulo de la iteración previa para mantener la dirección.

Es un método muy robusto en condiciones donde la detección de carriles puede ser incompleta o inexacta. Depende de la calidad de las detecciones de las líneas de carril, y puede no ser preciso en todas las condiciones de la carretera.

- **Árbol de Decisiones** (Propuesta por Abdelrahman et al [9]).

Utiliza un árbol de decisiones para determinar la acción de dirección basada en la situación actual del vehículo en la carretera. El procedimiento es:

- Segmentación de la Imagen de la Carretera: La carretera se divide en varios segmentos verticales, generalmente una parte superior y una inferior.
- Extracción de Características: Se extraen características como la detección de las líneas de frontera y su ubicación en cada segmento.

**Árbol de Decisiones:**

  - Segmento Superior: Se utiliza un árbol de decisiones para analizar el segmento superior de la imagen, determinando si las líneas de carril están presentes y su posición relativa.
  - Segmento Inferior: Otro árbol de decisiones analiza el segmento inferior de la imagen.

- Combinación de Decisiones: Los resultados de ambos árboles de decisión se combinan para decidir el ángulo de dirección.
- Factores de Decisión: Los árboles de decisión consideran factores como la presencia de líneas de frontera, su posición relativa, y si el vehículo está dentro del carril o se aproxima a una curva.

Puede manejar escenarios complejos y variabilidad en la detección de carriles, proporcionando una decisión robusta y adaptativa. La construcción y entrenamiento de árboles de decisión pueden ser complicados y requieren un conjunto de datos representativo de diferentes escenarios de conducción para el aprendizaje.

## 3.4 Tecnologías de Robótica Aplicadas

En el anterior capítulo se describen brevemente los sistemas de detección y tecnologías asociadas para el seguimiento de carriles, en este se realiza una selección de estas para la implementación. La selección puede estar basada en diferentes motivos:

- Eficiencia
- Simplificación del sistema
- Disponibilidad de sensores
- Física del robot
- Limitaciones de procesamiento

Para ello se van a establecer unas limitaciones, restricciones y asunciones previas a la selección de tecnologías, todas ellas tienen que ver con las características del carril y del robot

**Carril**

- El carril será de ancho constante y suficiente para la ubicación del robot
- El carril está delimitado por una línea roja
- El carril no tendrá curvas muy pronunciadas, dado que el máximo ángulo de giro del robot es 30º

*Figura 9: Circuito*

**Robot**

- La velocidad del robot será constante y suficientemente lenta para realizar las curvas sin tener que disminuir la velocidad
- Adicionalmente, la velocidad del robot será suficientemente lenta como para en un contexto de 5 FPS no recorra más de 5 cms, Esto significa que cada 5 cms como máximo el robot puede rectificar la dirección para mantenerse dentro del carril.
- Un Raspberry Pi 4B soportará el conjunto de algoritmos seleccionados sobre una plataforma ROS2. El siguiente gráfico nos puede dar una idea.

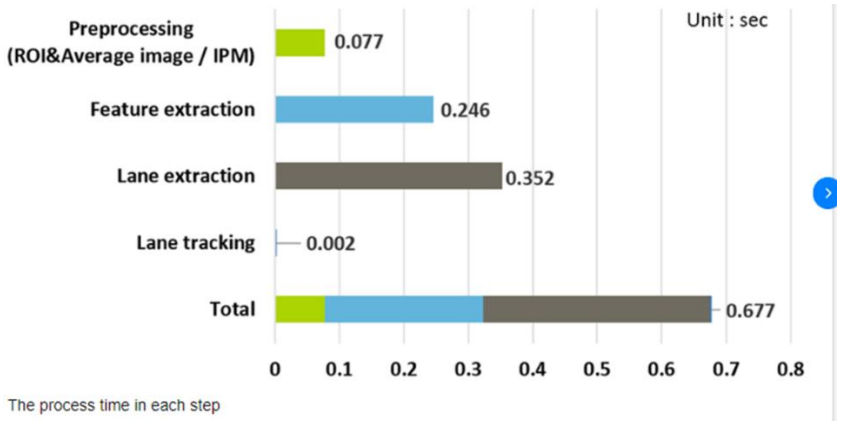


*Figura 10: Tiempo de procesamiento*

*Tabla 2: Elección del modelo*

| Tecnología | Selección | Razones |
|---|---|---|
| Sensor | Cámara | OpenCV casi un estándar |
| Extracción de características | Límites rojos (detección del rojo) | Detección de características fiable |
| Post-procesamiento | Transformada de Hough | Casi un estándar en la detección de rectas en OpenCV |
| Seguimiento de carril | Filtro de Kalman | Su uso es generalizado, es eficiente (poco proceso) |
| Modelo de carril | Lineal (campo cercano) | Sencillez |
| Posición respecto al carril | Offset a final de ROI | Por baja velocidad y sencillez |
| Control de Dirección | Promedio Lineal de Líneas Detectadas | Por pérdida de detección de bordes de carril |
| Gestión de control | PID | Eficiencia, ampliamente usado en procesos industriales |

### 3.4.1 Sensor Cámara: Visión Artificial

En general, las tareas de visión artificial incluyen métodos para:

- **Adquisición**: ¿Cómo adquieren los sensores, por ejemplo, las cámaras, imágenes de**l** mundo circundante? La forma en que esas imágenes se codifican, las características de la escena, como vértices, aristas, materiales, iluminación, etc. En nuestro caso se generarán escenas (frames) a una velocidad (FPS) suficiente para que el robot se mantenga dentro del carril.
- **Procesamiento**: Basado en cómo se encuentra la información sobre el mundo circundante es codificada, las imágenes se procesan aplicando varias técnicas de tratamiento de señales. En nuestro caso se detectarán los límites del carril rojos
- **Análisis**: ¿Cuáles son los métodos o algoritmos utilizados para extraer información de las imágenes procesadas y proporcionar una descripción de la escena (fotograma)? En nuestro caso mediante la Transformada de Hugh se generará líneas candidatas a ser el límite del carril.
- **Comprensión**: Cómo son las descripciones de los objetos y del entorno es almacenado y representado

El procesamiento de imágenes es un tipo de procesamiento de señales, que tiene una imagen, una serie de imágenes o fotogramas (“frames”) de vídeo como entrada, mientras que la salida son imágenes procesadas, tal vez en un formato diferente. El procesamiento de imágenes se explota a menudo en la visión artificial para manipular imágenes digitales.

La visión puede considerarse la parte más importante de la percepción humana y desempeña el mismo papel en la detección de máquinas. Sin embargo, a diferencia de los humanos, las máquinas "ven" el mundo como imágenes digitales de escenas 2D o 3D producidas por sensores. La imagen digital 2D, que, en muchos casos, representa una proyección de una escena 3D, puede definirse como una matriz bidimensional de muestras de intensidad llamadas píxeles. Los píxeles normalmente representan niveles de gris, colores, opacidades, etc. La imagen digital 2D puede ser procesadas y manipuladas por programas informáticos para producir información útil sobre el mundo

**OpenCV**

OpenCV (Open Source Computer Vision Library) es una biblioteca de software de código abierto ampliamente utilizada en el campo de la visión por computadora y el aprendizaje automático. Desarrollada inicialmente por Intel, ahora es mantenida por la comunidad y disponible bajo una licencia BSD, lo que la hace libre de usar tanto para aplicaciones académicas como comerciales. A continuación, se presenta una descripción general de OpenCV y sus funcionalidades principales:

Funcionalidades Principales

- Procesamiento de Imágenes Básico.
    - Lectura y Escritura de Imágenes y Videos: Funciones como imread(), imwrite(), VideoCapture() y VideoWriter().
    - Transformaciones Geométricas: Redimensionamiento (resize), rotación (rotate), traslación y deformación de imágenes (warpAffine y warpPerspective).
    - Operaciones Aritméticas y Lógicas: Adición, sustracción, operaciones bit a bit entre imágenes.
- Filtrado y Transformaciones de Imagen

    - Filtrado de Imágenes: Filtros como Gaussiano (GaussianBlur), Mediana (medianBlur), Bilateral y de Caja.
    - Transformaciones de Fourier: DFT (dft) y IDFT (idft) para análisis de frecuencia.
    - Operaciones Morfológicas: Erosión, dilatación, apertura y cierre (erode, dilate, morphologyEx).
- Detección y Descripción de Características
    - Detección de Bordes: Algoritmo de Canny (Canny), Sobel y Laplaciano.
    - Detección de Esquinas: Algoritmo de Harris (cornerHarris), Shi-Tomasi (goodFeaturesToTrack).
    - Descriptores de Características: SIFT, SURF, ORB para detección y descripción de puntos clave.
- Reconocimiento y Seguimiento de Objetos
    - Detección de Objetos: Clasificadores en cascada (CascadeClassifier) como Haar y LBP para detección de caras.
    - Seguimiento de Objetos: Métodos de seguimiento como MeanShift, CAMShift, y varios algoritmos de seguimiento basados en la región (KCF, GOTURN).
- Visión en Movimiento y Estructura a partir del Movimiento
    - Detección de Movimiento: Detección de fondo (BackgroundSubtractorMOG, BackgroundSubtractorKNN).
    - Óptica de Flujo: Algoritmos como Lucas-Kanade (calcOpticalFlowPyrLK) y Farneback (calcOpticalFlowFarneback).
    - Estructura desde el Movimiento: Estimación de la estructura 3D y movimiento a partir de secuencias de imágenes.
- Visión Estéreo y 3D
    - Calibración de Cámaras: Calibración de cámaras individuales y estéreo (calibrateCamera, stereoCalibrate).
    - Rectificación y Correspondencia Estéreo: Rectificación estéreo (stereoRectify), cálculo de mapas de disparidad (StereoBM, StereoSGBM).
- Aprendizaje Automático
    - Modelos de Aprendizaje Supervisado: SVM, KNN, Árboles de Decisión, Bosques Aleatorios, Redes Neuronales Artificiales.
    - Modelos de Aprendizaje No Supervisado: K-means, GMM (Modelos de Mezcla Gaussiana).
    - Interfaz con DNN: Carga y ejecución de modelos preentrenados de redes neuronales profundas (DNN) desde frameworks como TensorFlow, Caffe, y PyTorch.

Ejemplos de Uso

- Reconocimiento Facial: Detección y reconocimiento de caras en imágenes y videos.
- Detección de Objetos en Tiempo Real: Uso en sistemas de vigilancia y vehículos autónomos.
- Reconstrucción 3D: Creación de modelos 3D a partir de imágenes estéreo o secuencias de video.
- Realidad Aumentada: Superposición de información digital en el mundo real.

### 3.4.2 Transformada de Hough

La transformada de Hough es una técnica utilizada en procesamiento de imágenes y visión por computadora para la detección de características geométricas en una imagen. Fue desarrollada por Paul Hough y patentada por IBM en 1962. La versión original estaba orientada a la detección de líneas rectas, pero ha sido extendida para reconocer otras formas, como círculos y elipses.

La transformada de Hough es una técnica utilizada en el procesamiento de imágenes y la visión por computadora para detectar características geométricas simples, como líneas, círculos y otros contornos regulares, en una imagen. Es especialmente robusta para detectar estas características incluso en imágenes con ruido o donde las características están parcialmente ocultas.

La idea central de la transformada de Hough es transformar el problema de la detección de características geométricas en una imagen desde el espacio de la imagen (espacio de píxeles) a un espacio de parámetros (espacio de Hough). Este cambio de espacio permite identificar características geométricas como líneas y círculos mediante la identificación de acumulaciones de puntos en el espacio de parámetros.

**Transformada de Hough para Líneas**

En lugar de representar una línea en la forma tradicional $y=mx+b$, la transformada de Hough utiliza la forma paramétrica (polar):

$\rho=x\cos\theta+y\sin\theta$

donde ρ es la distancia perpendicular de la línea al origen y θ es el ángulo de la perpendicular con el eje x.

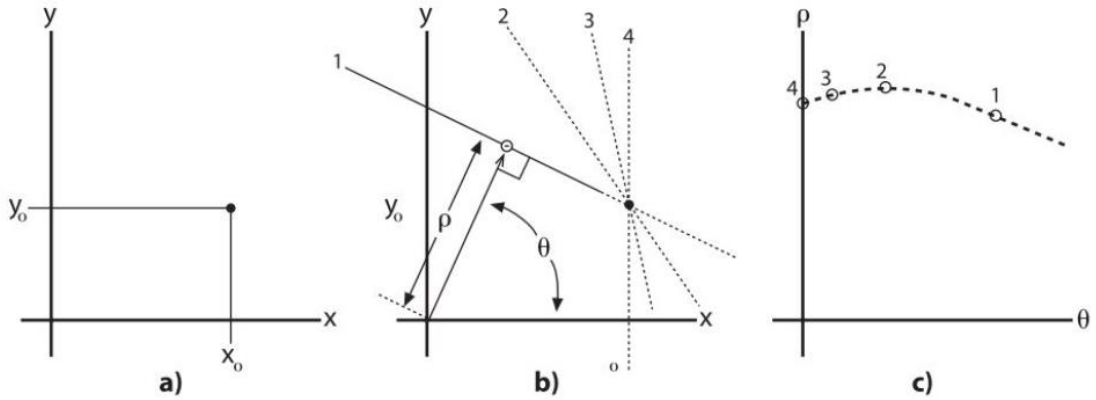


*Figura 11: Representación de rectas con Hough*

Proceso para la detección de líneas:

- Detección de bordes, puntos en el Espacio de la Imagen: Se consideran los puntos de borde en la imagen, típicamente obtenidos a través de un detector de bordes como Canny.
- Transformación al Espacio de Parámetros: Cada punto (x, y) en la imagen se transforma en una curva sinusoidal en el espacio de parámetros (ρ, θ).

- Acumulación: Se utiliza una matriz de acumulación para contar cuántas de estas curvas pasan por cada celda en el espacio de parámetros. Cada intersección indica la presencia de una línea con los parámetros correspondientes.
- Detección de Picos: Las celdas con el mayor número de intersecciones (picos en la matriz de acumulación) indican la presencia de líneas.

La transformada de Hough es una herramienta fundamental en visión por computadora para la detección robusta de formas geométricas básicas, proporcionando una base sólida para aplicaciones avanzadas en procesamiento de imágenes y análisis de datos visuales.

En este trabajo la Transformada de Hough juega un papel vital para transformar la detección de características de bordes de carril rojo en rectas con su representación matemática.

### 3.4.3 Filtro de Kalman

El filtro de Kalman es una herramienta matemática muy útil en el campo del procesamiento de señales y el control de sistemas. Rudolph E. Kalman lo introdujo por primera vez en 1960. Su principal función es estimar el estado de un sistema dinámico a partir de mediciones ruidosas o incompletas.

El filtro de Kalman actúa bajo las siguientes asunciones:

- El sistema que se está modelando debe ser lineal.
- El ruido al que están sujetas las mediciones es "blanco", o, en otras palabras, "no correlacionado en el tiempo".
- El ruido es de naturaleza gaussiana.

Bajo esas suposiciones, el filtro de Kalman tiene la capacidad de estimar en el paso de tiempo actual un nuevo modelo para el estado del sistema que sea lo más posiblemente preciso, teniendo en cuenta el modelo en el paso de tiempo anterior con su incertidumbre y la medición actual con su incertidumbre

El estado x del sistema (en nuestro caso la posición del vehículo con respecto al carril) en el tiempo t se generaliza como función del estado en el tiempo t-1:

$$xt = Axt\text{-}1 + But + wt$$

- xt y xt-1 son 2 vectores n-dimensionales (posición respecto al carril) en los time-stamps t y t-1.
- ut es un vector c-dimensional de control en el time-stamp t.
- A es una matriz nxn que asocia el estado previo en t-1 con el actual en t (matriz de transición de estados o de dinámica del sistema).
- B es una matriz nxc que asocial la entrada de control (en nuestro caso la dirección) al estado x.
- wt es una variable aleatoria que representa el ruido en el estado de transición. Se asume una distribución Gaussiana N(0;Q), donde Q es la matriz de covarianza nxn.

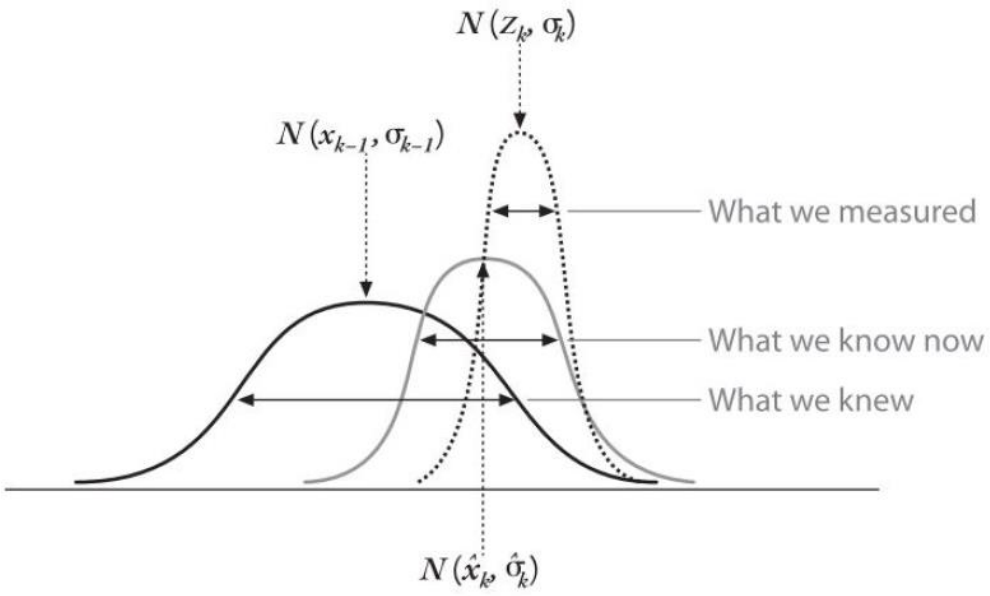


*Figura 12: Función filtro de Kalman*

En general la operación con el filtro de Kalman puede dividirse en dos etapas:

- **Predicción**: estima el valor de la posición con respecto a los carriles
- **Corrección**: la corrige teniendo en cuenta las mediciones

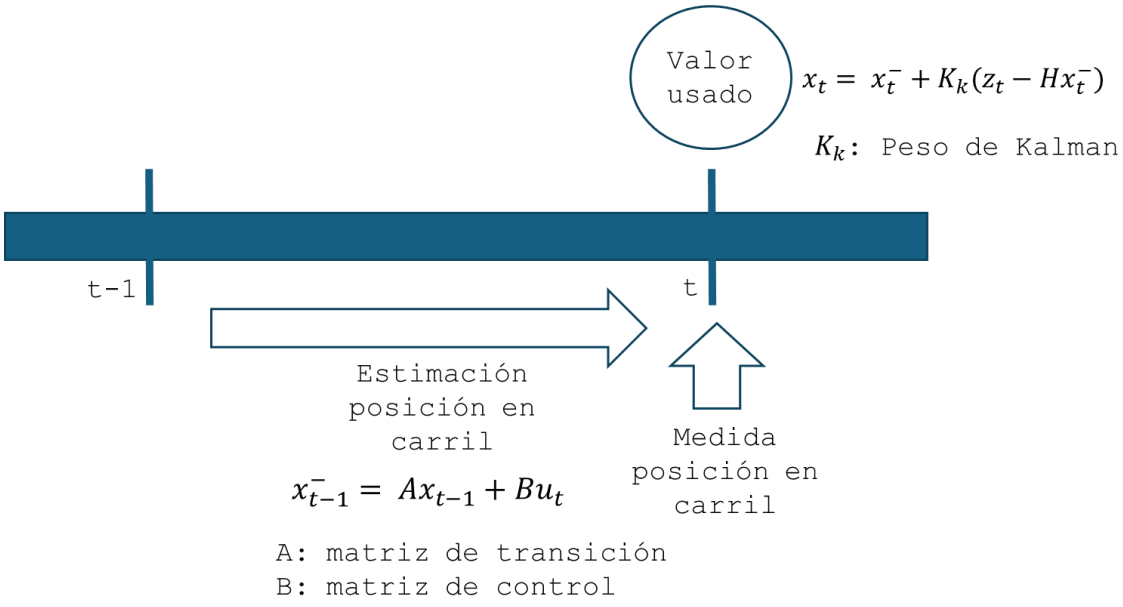


*Figura 13: Filtro de Kalman para carriles*

La matriz A, también conocida como matriz de transición de estado o matriz de dinámica del sistema, describe cómo evoluciona el estado del sistema de un paso de tiempo al siguiente. En muchos casos prácticos, esta matriz puede ser constante si el modelo dinámico del sistema es lineal y no cambia con el tiempo. Por ejemplo, para un sistema lineal como un péndulo simple o un vehículo que se mueve a velocidad constante, A podría ser constante y representar la relación lineal entre el estado actual y el estado siguiente.

Sin embargo, en sistemas más complejos o no lineales, A podría ser variable y depender explícitamente del estado actual del sistema. Por ejemplo, en sistemas no lineales, A podría ser una función del estado actual del sistema, lo que implica que cambia en cada paso de tiempo.

La matriz B se utiliza para modelar cómo las entradas de control afectan el estado del sistema. Esta matriz se multiplica por el vector de entrada de control $uk$ en la fase de predicción del filtro de Kalman. Al igual que $A$, $B$ puede ser constante o variable dependiendo del modelo específico del sistema.

**Constante**: En muchos casos, $B$ es constante si el efecto de las entradas de control sobre el sistema es lineal y no cambia con el tiempo.

**Variable**: En sistemas donde el efecto de las entradas de control es no lineal o depende del estado actual del sistema, $B$ podría variar en cada paso de tiempo.

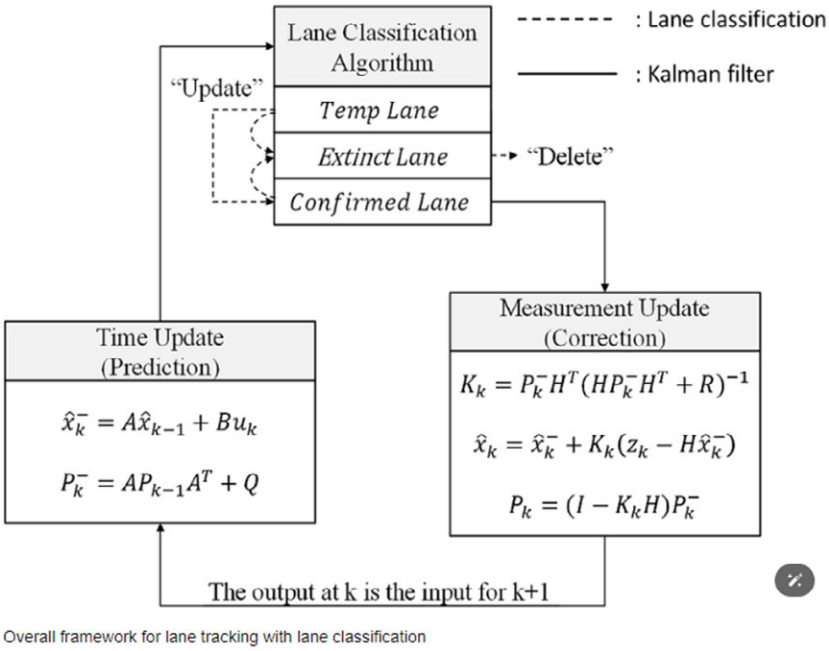


*Figura 14: Diagrama general de Kalman*

Algunas de las principales aplicaciones y utilidades del filtro de Kalman son:

- Seguimiento de objetos en movimiento: El filtro de Kalman es ampliamente utilizado para rastrear y predecir la posición, velocidad y aceleración de objetos en movimiento, como vehículos, aviones, proyectiles, etc. Esto es particularmente útil en aplicaciones como navegación, control de tráfico, detección de objetivos, etc.
- Fusión de sensores: Cuando se tienen múltiples sensores que miden la misma cantidad, el filtro de Kalman permite combinar estas mediciones de manera óptima, reduciendo el ruido y la incertidumbre de las estimaciones.
- Estimación de estados en sistemas lineales: El filtro de Kalman se utiliza para estimar variables de estado como posición, velocidad, aceleración, etc. en sistemas dinámicos lineales, a partir de mediciones ruidosas.
- Predicción y control: Gracias a su capacidad de estimar el estado futuro de un sistema, el filtro de Kalman se emplea en aplicaciones de control y predicción, como en el control de vuelo de vehículos aéreos o el control de procesos industriales.
- Procesamiento de señales: El filtro de Kalman se utiliza para filtrar y suavizar señales ruidosas, mejorando la calidad de la información extraída de ellas.

El filtro de Kalman es una herramienta poderosa y versátil que permite estimar de manera óptima el estado de sistemas dinámicos a partir de mediciones imperfectas, lo cual lo convierte en un elemento clave en una amplia gama de aplicaciones de ingeniería.

La aplicación en la detección de carriles es muy directa dado que los sensores debido a las condiciones ruidosas ambientales generan a menudo detecciones de carriles incompletas o incluso ausentes, se puede pensar en caso frecuente de detección de solo uno de los carriles, el filtro de Kalman estima en base a las detecciones anterior(es) el carril actual. Además, es una herramienta poderosa en el "suavizado" de las detecciones permitiendo de esta forma un mejor control de la dirección del vehículo.

### 3.4.4 Control PID

Los coches autónomos y los robots no se mueven con precisión perfecta. Su trayectoria puede verse afectada por su entorno (como superficies no planas) y por ligeros desajustes en su mecánica (como ruedas desalineadas).

Un ser humano que conduce un coche con neumáticos imprecisamente alineados puede corregir constantemente para asegurarse de que está conduciendo recto. ¿Cómo podemos enseñar a un coche autónomo a hacer lo mismo?

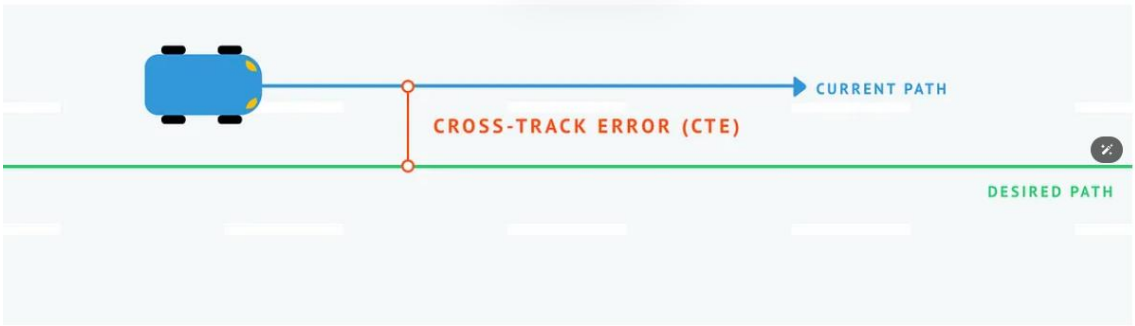


*Figura 15: Ejemplo de control PID*

Suponiendo que tenemos una trayectoria deseada para que el coche la siga, podemos implementar un algoritmo conocido como Control PID para asegurarnos de que el coche se realinee con el objetivo.

Para entender por qué los sistemas utilizan el Control PID (o los Sistemas de Control en general), es útil comprender la teoría básica del control en bucle abierto y cerrado.

Si un coche se adhiriera a un sistema de bucle abierto, la salida de ese sistema no tendría ningún efecto sobre la acción de control de la señal de entrada. En su lugar, el coche seguiría fielmente las instrucciones del comando de entrada independientemente del resultado final. Por lo tanto, si le indicamos a la computadora de un coche que se mueva cinco metros, siempre se moverá cinco metros, incluso si termina deteniéndose antes o después de la posición objetivo.


*Figura 16: Funcionamiento del PID*

En un sistema de bucle cerrado, la salida se retroalimenta como entrada para reducir errores y aumentar la estabilidad. Su arquitectura de paso hacia adelante es similar a la de un sistema de bucle abierto, pero se diferencia en que parte de la salida proporciona retroalimentación a la entrada. Esto se conoce como control de retroalimentación. En un sistema así, la señal de referencia (que representa el objetivo deseado) se puede comparar con el valor medido, proporcionando posteriormente un término de error.

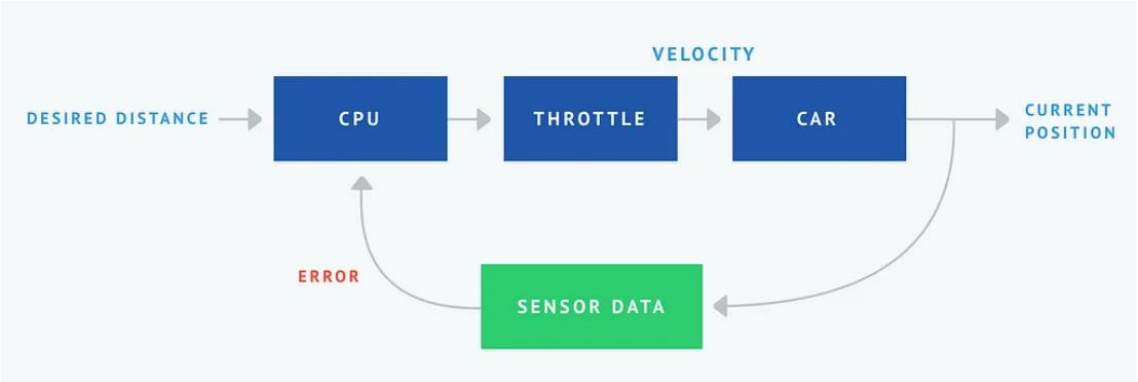


*Figura 17: Bucle cerrado PID*

La variable de proceso es un parámetro del sistema que necesita ser controlado. Se utilizan sensores para medir la variable de proceso y proporcionar retroalimentación al sistema de control.

El punto de ajuste es el valor deseado de la variable de proceso. El controlador o compensador utiliza la diferencia entre la variable de proceso y el punto de ajuste para determinar la salida del actuador que impulsará el sistema o planta.

Forzar al actuador a girar el volante a la izquierda hará que el coche gire a la izquierda, lo que resultará en un cambio en la variable de proceso, ya que la distancia entre el coche y el límite también cambiará.

En la vida real, girar y la velocidad no serán los únicos factores que afecten la distancia al límite, ya que las condiciones de las ruedas y la carretera también influirán en la posición final del coche. Estas influencias externas se conocen como perturbaciones.

El error en estado estacionario es la diferencia final entre la variable de proceso y el punto de ajuste.

PID significa Proporcional Integral Derivativo, diferentes métodos para tratar el error entre la variable de proceso y la señal de referencia, que se utilizarán para impulsar el sistema. Una forma de entender PID es descomponerlo en términos de análisis del error utilizando información pasada (Integral), información presente (Proporcional) e información futura (Derivativa).

$$u(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(t')\,dt' + K_\mathrm{d} \frac{de(t)}{dt}$$

Observa que, en la expresión matemática para PID, cada respuesta tiene un coeficiente de ganancia K que ajusta la sensibilidad del sistema a cada una de las componentes. Las componentes pueden ser anuladas estableciendo las ganancias en 0. Por ejemplo, podríamos crear un controlador P configurando I y D a cero.

**Respuesta Proporcional**

La respuesta proporcional examina un error en el presente, ya que evalúa el error de distancia proporcionalmente desde un paso de tiempo dado. En otras palabras, si la cantidad de error es baja, la corrección es pequeña. Por el contrario, si hay mucho error, la corrección es mucho mayor.


*Figura 18: Respuesta proporcional en Control PID*

Pensar en un péndulo sostenido en la mano de alguien. Cuando no está en reposo, oscilará de un lado a otro, ya que la gravedad siempre lo atraerá de nuevo a la posición inicial con una fuerza que depende de la distancia desde esa posición. Cuando está lejos de la posición inicial, el péndulo oscilará con mucha fuerza.

El problema de usar Control Proporcional es el sobre impulso, donde un objeto corrige en exceso su posición. Es similar a un péndulo que oscila más allá de la posición de reposo. Por sí solo, el Control Proporcional puede tener problemas cuando un objeto tiene masa o inercia, porque estos factores afectarán la velocidad del objeto independientemente de la disminución de la influencia de lo que esté acelerando el objeto, como la gravedad o el motor de un coche. No anticipa que está regresando a la posición objetivo, por lo que tiende a sobrepasarse y oscilar.

Si el Control Proporcional tiene una ganancia demasiado alta, el controlador compensará en exceso constantemente en ambas direcciones, resultando en una oscilación notable. Es similar a acelerar un coche a fondo desde una posición de reposo y luego frenar bruscamente cuando inevitablemente supera el límite de velocidad.

**Respuesta Integral**

La integral ayuda a examinar los datos del pasado al sumar los errores pasados. Incluso un error pequeño aumentará progresivamente la respuesta integral, considerando que algo no se ha corregido adecuadamente con el tiempo.

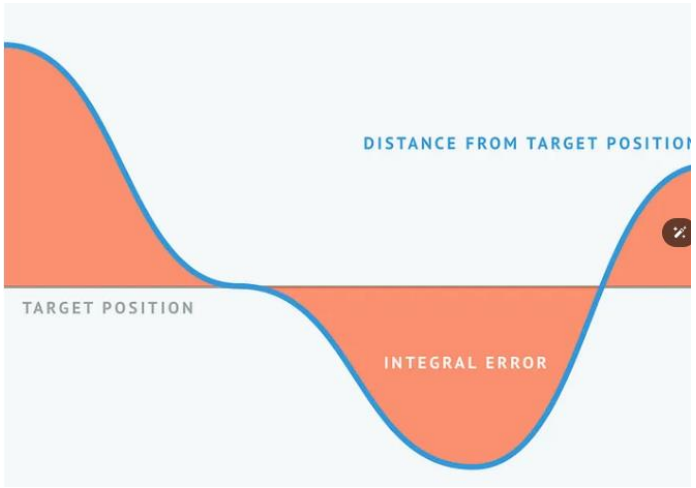


*Figura 19: Respuesta Integral en Control PID*

Es útil para eliminar errores constantes en un sistema de control, ya que no importa el pequeño error constante, eventualmente la suma de ese error será tan significativa como para ajustar la salida del controlador. El efecto reduce el error en estado estacionario a cero.

**Respuesta Derivativa**

La respuesta derivativa anticipa el futuro examinando la tasa de cambio del error que contribuye en un paso de tiempo dado. La respuesta derivativa es proporcional a la tasa de cambio de la variable de proceso. Cuando el cambio del error se produce lentamente, la ruta derivativa es pequeña. Cuanto más rápido cambia el error, mayor es la ruta derivativa. A veces se le llama control anticipatorio.

Si la ganancia es demasiado grande, el controlador podría volverse inestable porque las fluctuaciones normales del controlador se exagerarán. Si es demasiado pequeña, puede tardar demasiado en responder a cambios dinámicos.

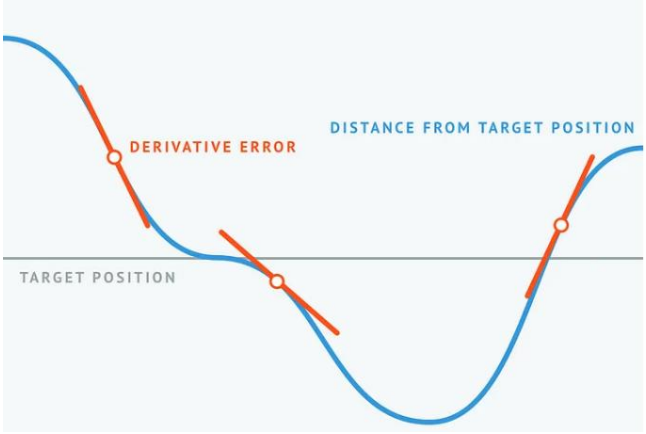


*Figura 20: Respuesta derivativa en Control PID*

A diferencia de las respuestas Proporcional e Integral, la Respuesta Derivativa no tiene impacto en el error en estado estacionario. Su función es principalmente reducir el sobre impulso.

## 3.5 Modelo de Sistema (robot y entorno)

En este capítulo modelizaremos el robot y el entorno, básicamente el carril. Modelizarlo significa adaptar las tecnologías vistas en el capítulo anterior (que nos permiten realizar un seguimiento de carril) a la realidad física de nuestro sistema.

La idea es hacer la modelización para que el bucle de control pueda realizarse eficientemente en sus diferentes etapas ilustradas en la figura:

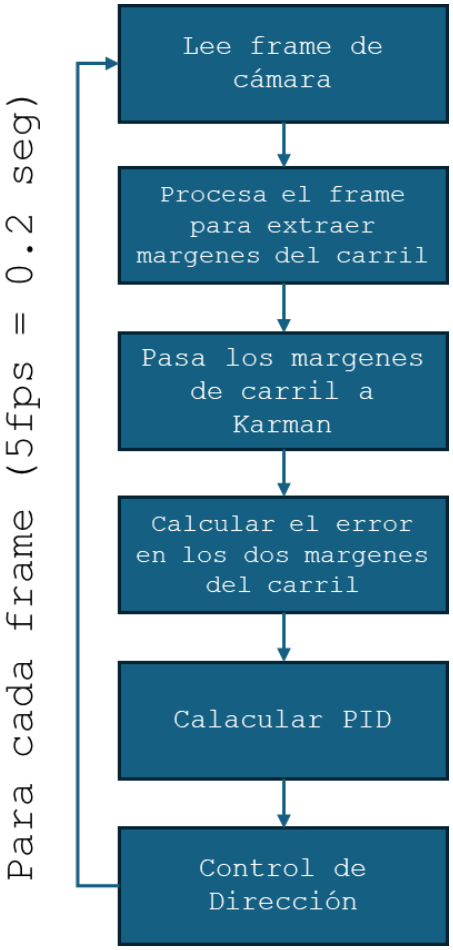


*Figura 21: Modelo del Sistema*

1- Adquisición de imágenes:
   - Captura de imágenes del entorno de la carretera utilizando una cámara montada en el vehículo.
   - Definición de la región de interés (ROI)
2- Preprocesamiento de la imagen:
   - Aplicación de técnicas como conversión a escala de grises, ecualización del histograma, suavizado, etc. para mejorar la calidad de la imagen.
   - Segmentación de la imagen para resaltar los elementos relevantes, como las marcas del carril.
3- Detección de carriles:
   - Aplicación de algoritmos de detección de características, como la transformada de Hough, para identificar las líneas que delimitan los carriles.

- Ajuste de modelos geométricos, como rectas o curvas, a las características detectadas.

4- Seguimiento de carriles:
   - Utilización de técnicas de seguimiento, como filtros de Kalman o correlación de patrones, para mantener la continuidad de la detección de carriles entre frames.
   - Actualización de los modelos geométricos a medida que el vehículo avanza.
5- Estimación de la posición del vehículo:
   - Cálculo de la posición y orientación del vehículo dentro del carril, a partir de los modelos geométricos estimados.
6- Control del vehículo:
   - Envío de las señales de control (dirección, aceleración, frenado) al sistema de conducción del vehículo, para mantenerlo centrado en el carril y seguir la trayectoria deseada.

### 3.5.1 Velocidad del Robot. ROI

Es interesante realizar algunos cálculos que se derivan de la velocidad del robot. En los requisitos se estableció que el robot tendrá una velocidad constante y suficientemente baja para poder realizar el procesamiento necesario y trazar las curvas correctamente.

Es claro que el control de la dirección se realiza en cada fotograma, por tanto, entre frames, no hay control, es importante que la distancia recorrida no sea muy grande como para que el robot se salga del carril, esta circunstancia viene determinada por la pérdida de los dos bordes del carril. En ocasiones puede perderse un borde en situaciones de giro, esta circunstancia debe de ser gestionada por el robot.

- Velocidad del robot = v metros/seg
- Frames por segundo = FPS
- Tiempo sin control = 1/FPS seg
- Distancia sin control = v/FPS metros

Si fijamos los valores de velocidad a 0.18 m/s y 5 FPS obtenemos los siguientes valores:

- Tiempo sin control = 1/5 = 0.20 seg
- Distancia sin control = 3.6 cms

Estos valores parecen bastante adecuados.

En este punto se puede establecer una ROI adecuada. El análisis de la ROI nos da información acerca de cómo se va a comportar el carril a futuro.

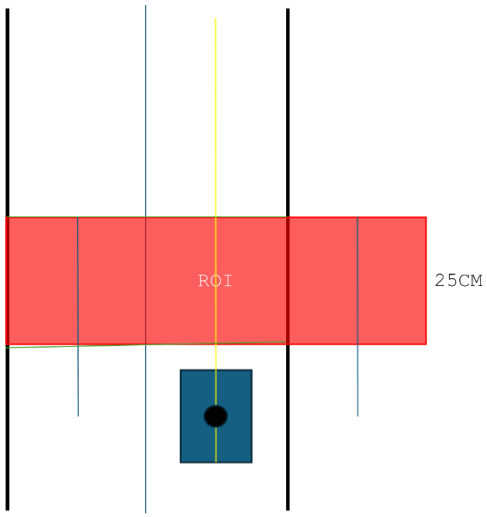


*Figura 22: Región de Interés del Robot*

ROI = 25 cms

Tiempo de dejar atrás la ROI = 1.4 seg

Además, al ser la ROI corta, podemos representar los bordes del carril como rectas, esto simplifica mucho el análisis.

Los valores calculados aquí son teóricos y pudieran ser cambiados en la fase de implementación y pruebas para una correcta operación del robot

### 3.5.2 Modelo de Carril

El carril tiene las siguientes características:

- Delimitado por una línea roja, esta es la característica que se tiene que extraer para la detección del carril
- Anchura, suficiente para las dimensiones del robot: 27 cms.
- Radio de curvatura no inferior a 20 cms, este valor permite al robot poder hacer la trazada de la curva.

*Figura 23: Frame centrado*

### 3.5.3 La cámara. Medición de Distancias.

La calibración de la cámara no forma parte del funcionamiento del sistema, pero para todos los sistemas de visión por computadora tiene una influencia extremadamente grande en el resultado. Los parámetros incluyen la posición, el ángulo de visión y parámetros internos de la cámara.

Los parámetros internos de una cámara son la posición del centro de la imagen, la longitud focal, los factores de escala y la distorsión del lente. Definir correctamente estos parámetros puede ayudar significativamente a aumentar la precisión del sistema.

Con una cámara correctamente calibrada podemos medir distancias en la imagen en base al número de pixeles

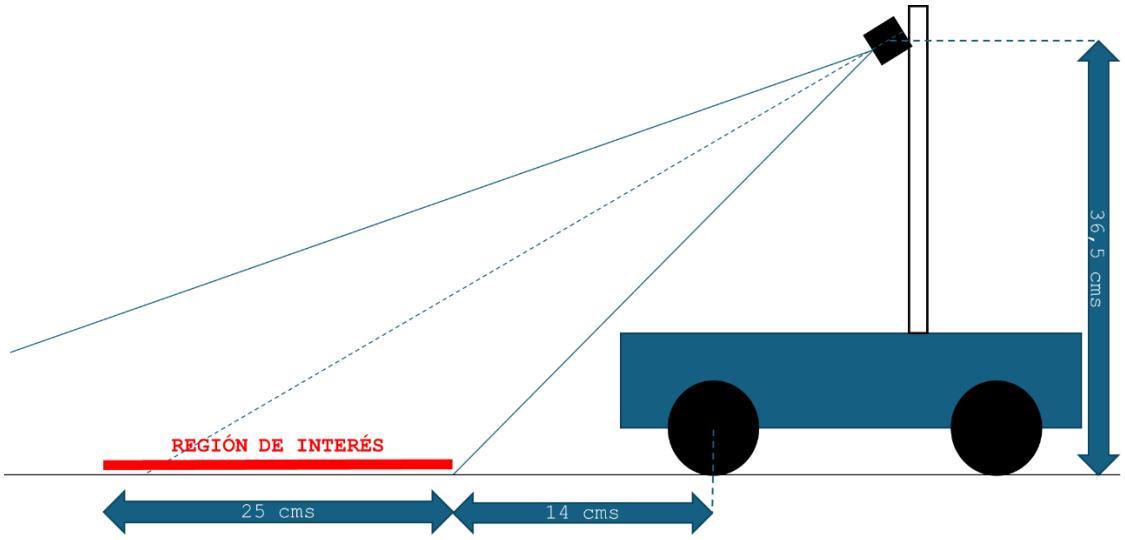


*Figura 24: Esquema ángulo de la cámara*

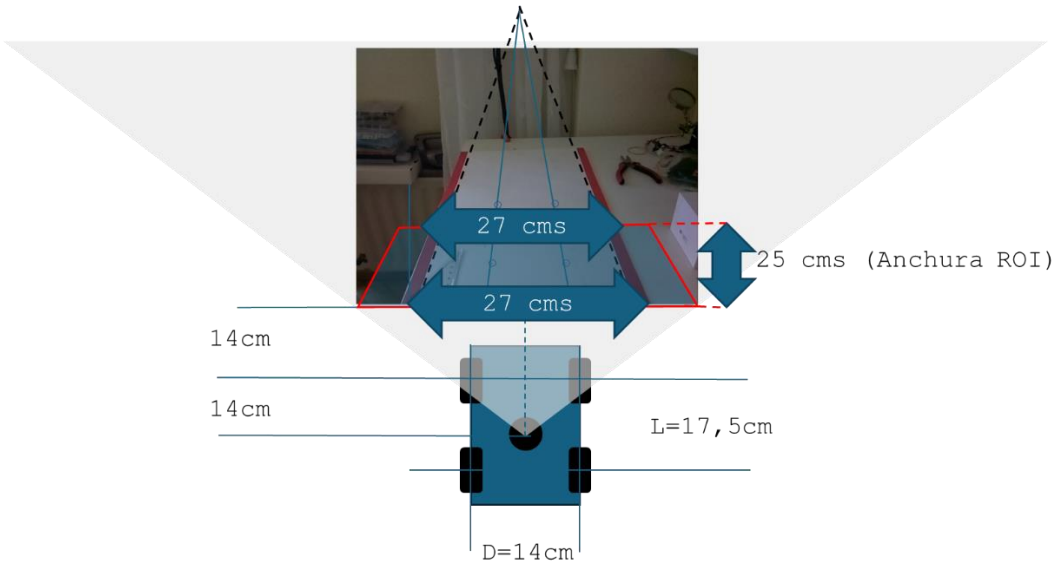


*Figura 25: Esquema del foco*

**Datos del Robot**

- Distancia entre ejes L = 17,5 cms
- Anchura D = 14 cms
- Diámetro de ruedas r = 6,5 cms
- Ángulo Máximo de giro (dirección tipo Ackermann) θ = 30º

### 3.5.4 Modelo Geométrico del Robot [10]

Una simplificación común de un vehículo con dirección Ackerman utilizado para el seguimiento geométrico de trayectorias es el modelo de bicicleta. Para el propósito del seguimiento geométrico de trayectorias, solo es necesario mencionar que el modelo de bicicleta simplifica el coche de cuatro ruedas combinando las dos ruedas delanteras juntas y las dos ruedas traseras juntas para formar un modelo de dos ruedas, como una bicicleta. La segunda simplificación es que el vehículo solo puede moverse en un plano. Estas simplificaciones resultan en una relación geométrica simple entre el ángulo de dirección de la rueda delantera y la curvatura que seguirá el eje trasero. Como se muestra en la figura, esta relación geométrica simple puede expresarse como:

$$\tan(\delta) = L/R$$

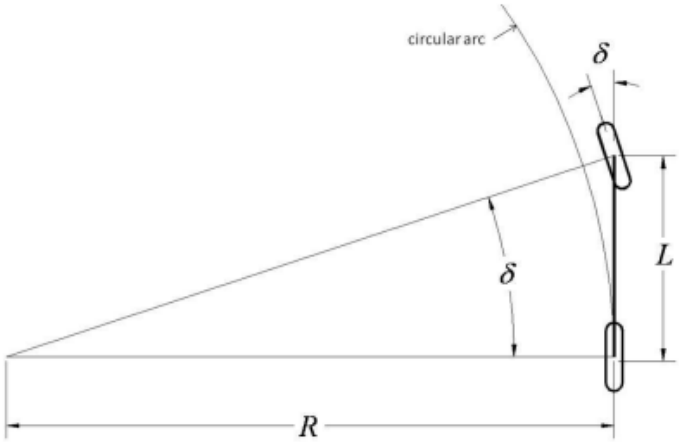


*Figura 26: Esquema modelo geométrico*

### 3.5.5 Posición respecto al carril

Un dato importe para poder controlar un vehículo es conocer la posición de este respecto al carril, si la conocemos podremos controlar la dirección para ubicar el vehículo en la posición correcta.

Existen dos parámetros que definen la posición:

- Desplazamiento con respecto al centro del carril (d en la figura)
- Ángulo del vehículo con respecto a los bordes que definen los límites del carril (α en la figura)

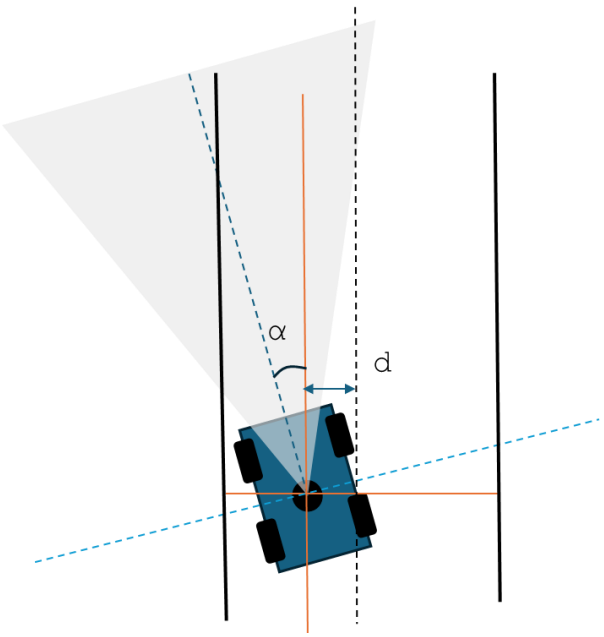


*Figura 27: Cálculo de posición respecto al carril*

Para una posición óptima tendríamos valores para d y α de cero. Es claro lo siguiente:

- d y α son negativos: hay que girar a la derecha
- d y α son positivos: hay que girar a la izquierda

Si d es positivo y α es negativo (o viceversa) ya no está tan claro cómo debemos de actuar.

Para poder controlar el vehículo es importante generar una función de error que nos proporcione información de la posición del vehículo y de cómo debemos actuar para corregirla, o lo que es lo mismo, minimizar la función de error. La función de error debe ser también fácil de extraer con las técnicas de visión artificial empleadas.

En muchos casos, se acude a la diferencia de puntos de fuga (vanishing points) entre el carril real y un carril virtual en donde el vehículo estuviera posicionado bien (valor d en el siguiente gráfico).

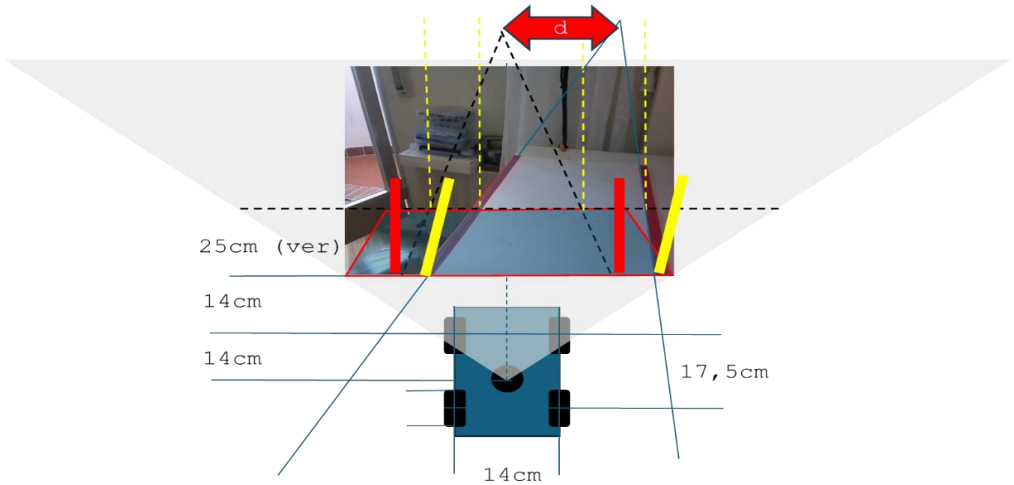


*Figura 28: Offstet del foco 1*

En este trabajo se propone simplificar los cálculos dado que por la velocidad baja del vehículo vamos a estar centrados en el campo cercano [referencia repositorio]. El siguiente gráfico muestra una propuesta de error en base a los valores d1 y d2, que son los desplazamientos con respecto a los límites del carril al final de la región de interés

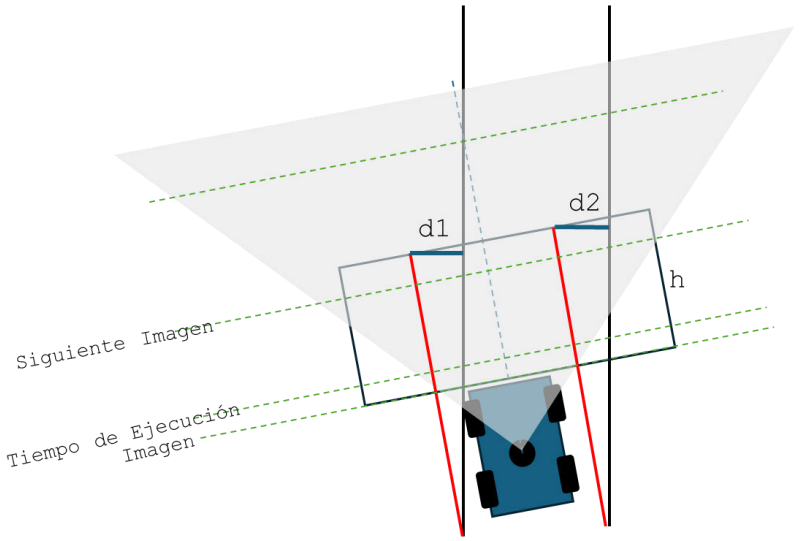


*Figura 29: Offset del foco 2*

O lo mismo, pero en perspectiva, tal y como lo capta la cámara:

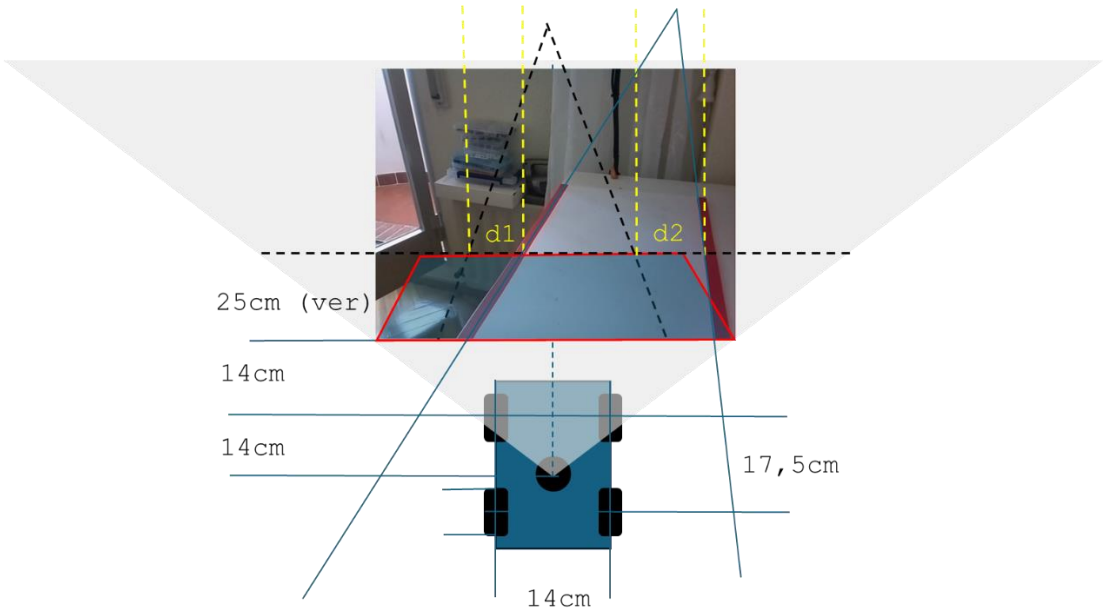


*Figura 30: Offset del foco 3*

### 3.5.6 Análisis de trayectoria

En este apartado se va a realizar un análisis de cómo a través de la dirección del robot puedo mejorar la posición de este con respecto al carril, o lo que es lo mismo, minimizar el error que se obtiene en cada fotograma (frame).

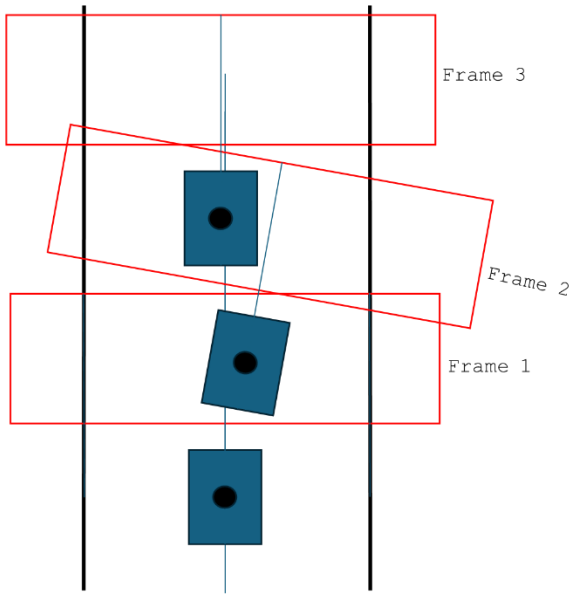


*Figura 31: Esquema de la trayectoria 1*

En la siguiente figura se muestran el movimiento del robot en dos frames cuando se aplica un ángulo (θ) a la dirección del robot. Para ello aplicaremos la aproximación del modelo geométrico descrito más arriba (tan(δ) = L/R)

Calculamos los valores de giro (α) y desplazamiento lateral (d) del robot entre dos frames

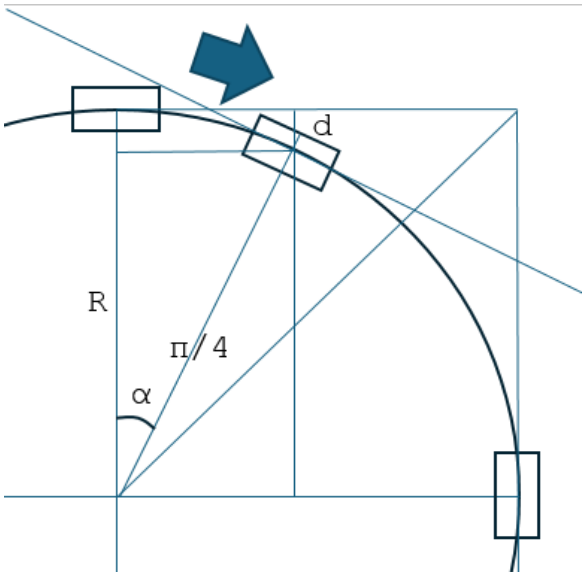


*Figura 32: Esquema de la trayectoria 2*

*tan(ϑ) = L/R = 17,5/R* (siendo L la distancia entre ejes del robot y R el radio de curvatura)

*R = 17,5/ tan(ϑ)* (θ máximo es π/4)

*R(min)* = *17,5 cms* y *R(max) =* ∞

Distancia recorrida entre *frames (l): 3,6 cms*

*α (rad) = 3,6/R (cm)*

*α (max) = 0,07 π ≈ 8º* (en 1 seg aprox puede corregir 45º)

*d (cms) = R(1-cos α) = R(1-cos(3,6/R)) = 17,5 (1 – cos(3,6/17,5))*

*d (max) = 0,37 cm (en un seg aprox puede corregir 1,8 cms)*

Estos valores son las correcciones que se pueden hacer en términos de ángulo y distancia al centro del carril. Si aplicamos el ángulo máximo de la dirección del robot (±30º o ±π/4):

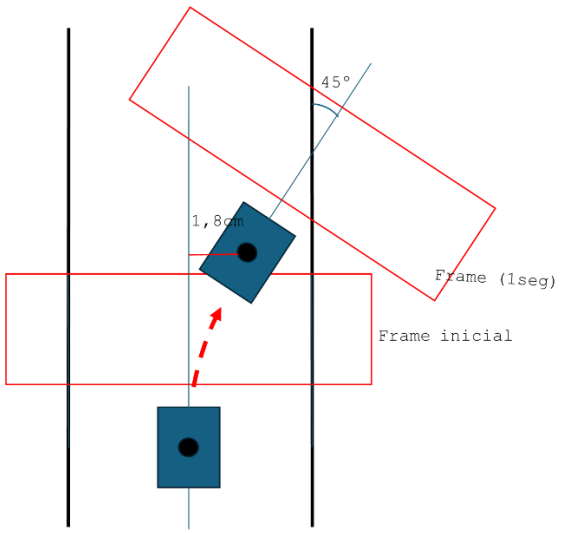


*Figura 33: Esquema de la trayectoria 3*

# 4 Metodología

Dentro de este apartado se realiza el diseño del sistema de seguimiento y la implementación, partiendo del primer proyecto del robot [11].

## 4.1 Diseño del Sistema de Seguimiento

### 4.1.1 Infraestructura previa

Toda la implementación del sistema de detección de carriles como el control de Robocar se encuentra en el repositorio https://github.com/rubenhigorg/robocar, en la rama TFG_2.

Para el diseño del sistema de seguimiento, se parte de una infraestructura Software implementada en ROS2 [11], donde existen una serie de nodos sobre los que se implementa este sistema. Aquí se detallan brevemente los nodos con su funcionalidad:

- **Camera_node**: Se encarga de recoger los frames de la cámara y publicarlos en el topic /camera_image. Teniendo en cuenta que la velocidad del robot en modo autónomo es de 18 cm/segundo, con una frecuencia de 3 frames por segundo es suficiente para hacer un seguimiento del carril óptimo.
- **Teleop_twist_joy**: Este nodo se encarga de recoger los datos de un mando inalámbrico de PS3 para publicarlos en el topic /Joy, que enviará mensajes del estado de todos los botones y Joysticks.
- **Car_control_node**: Se encarga de controlar el robot, tanto la velocidad como la dirección, en función de los datos que recoge del topic /Joy. Este nodo es modificado para incorporar el modo autónomo.

También están implementados otros nodos que pueden servir para mejorar el algoritmo de conducción autónoma, aunque no se utilizan para este proyecto:

- **Energy_node**: publica información sobre la energía, incluyendo el voltaje de tres baterías y la corriente, en un tópico llamado 'energy' cada segundo. Podría servir para reducir el consumo de la conducción autónoma bajando la velocidad del robot.
- **Imu_node**: publica datos de aceleración y velocidad angular obtenidos de un sensor MPU6050 en un tópico llamado 'imu' a una tasa de 2 Hz (cada 0.5 segundos). Utiliza la biblioteca MPU6050 para interactuar con el sensor y publica los datos redondeados a dos decimales. Puede ser útil para detectar derrapes y evitar que el coche deslice.
- **Distance_Node**: utiliza los tres sensores ultrasónicos HC-SR04 para medir distancias en las direcciones izquierda, derecha y centro, y un sensor de parada de emergencia (KY032). Publica los datos de distancia y el estado del sensor de emergencia en el topic /ultrasound_data. Es útil para detectar obstáculos en el carril y que el robot actúe en consecuencia.

**Car_control_node:** La calidad de los frames es de 640x480 píxeles. Esta calidad es más que suficiente para la detección de líneas:

*Figura 34: Frame centrado en el carril*

El formato de los frames enviados a través del topic /camera_image es bgr8, formato ampliamente utilizado para el procesamiento de frames por OpenCV.

## 4.2 Selección de Componentes

En este trabajo solo se presentan la cámara, la cual se ha tenido que cambiar debido a las deficiencias de la implementada en el primer proyecto del robot [11].

### 4.2.1 Cámara

Inicialmente se instaló una cámara para Raspberry clónica, los resultados fueron negativos por la baja velocidad de obturación, en el momento que el robot se desplazaba las imágenes de los frames) aparecían movidas.

Como solución se ha montado una cámara adicional de tipo webcam en el mástil del robot. El fabricante de la cámara es NexiGo y las características son:

- Cámara web Full HD 60 fps 1080p: la cámara web NexiGo Full HD está equipada con una resolución de alta definición de 1920 x 1080p.
- Balance de blancos automático y control de exposición automática.
- Proporciona imágenes claras y vídeo de alta definición a 60 fps, incluso cuando se graba en ajustes de poca retroiluminación la cámara se ajusta de forma inteligente para producir la mejor imagen posible.
- Conexión tipo USB.


*Figura 35: Cámara NexiGo*

### 4.2.2 Sensor de velocidad

Para el control de la velocidad del robot se ha introducido un sensor en el eje de una de las ruedas traseras basado en un opto-acoplador. Lo que hace este sensor es detectar los impulsos de un led que pasan por una rejilla (ver figura)

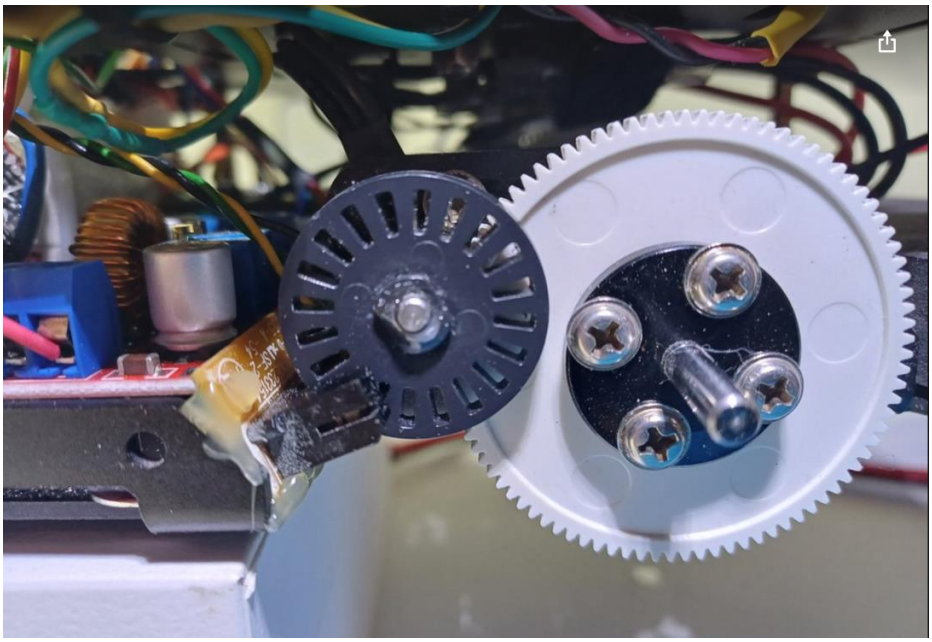

*Figura 36: Sensor de velocidad*

Los impulsos son generados por el opto-acoplador según el siguiente circuito

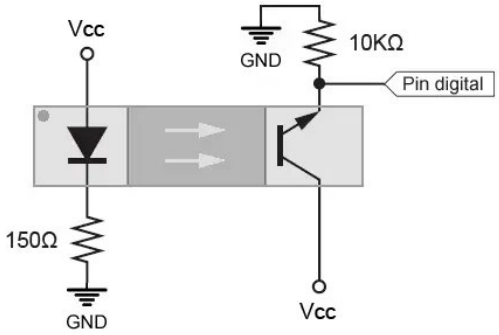


*Figura 37: Circuito opto-acoplador*

El conteo de los impulsos lo realiza un Arduino Nano y la información se entrega a la Raspberry en formato I2C para que se pueda integrar en ROS2.

## 4.3 Implementación

Aquí se procede a implementar, según el diseño descrito anteriormente, toda la lógica y pruebas para la detección y seguimiento de carriles.

### 4.3.1 Clase ImageProcessor

Toda la lógica de procesamiento de frames se implementa en la clase ImageProcessor, que tiene los siguientes métodos:

- *get_line_markings (frame)*: Aísla los carriles del frame, devolviendo un frame binario con las líneas detectadas.
- *doBlur (frames, iterations, kernelSize)*: Este método aplica un filtro de desenfoque gaussiano al frame pasado por parámetro.
- *doRegionOfInterest (frame)*: Devuelve una máscara con la región de interés seleccionada. Utiliza dos atributos declarados en la propia clase: self.roiX y self.roiY, que determinan el porcentaje seleccionado para la región en el eje de coordenadas.
- *findLanes (frame, lines, drawAll=False)*: Itera sobre las líneas obtenidas por la transformada de Hough y filtra las líneas para ubicar el carril izquierdo y derecho.
- *drawPoly (frame, poly, color, width=3)*: Esta función pinta sobre el frame pasado por parámetro la línea definida por el polinomio *poly*.
- *process (frame)*: principal método de la clase. Se encarga de utilizar todos los métodos de la clase más el algoritmo de Canny y la transformada de Hough para obtener las líneas, almacenándolas en los atributos de clase self.right y self.left.

```python
def process(self, frame):
    """
        Main pipeline for detecting lanes on a frame.
    """
    binary_frame = self.get_line_markings(frame)
    blured = self.doBlur(binary_frame, iterations=3, kernelSize=7)
    canny = cv2.Canny(blured, threshold1=20, threshold2=40)
    roi = self.doRegionOfInterest(canny)
    houghLines = cv2.HoughLinesP(
        roi,
        rho = 1,
        theta = np.pi / 180,
        threshold = 20,
        lines = np.array([]),
        minLineLength = 5,
        maxLineGap = 60
    )
    lanes = self.findLanes(frame, houghLines, drawAll=True)
    # self.drawPoly(lanes, self.left.poly, self.left.color, width=3)
    # self.drawPoly(lanes, self.right.poly, self.right.color, width=3)

    return lanes
```

*Figura 38: Código método process*

### 4.3.2 Notebook

Para la implementación del algoritmo diseñado, se utiliza un Jupyter Notebook para el procesamiento y posterior análisis de los frames recogidos por la cámara, ubicado en pruebas/computer_vision/video.ipynb.

El notebook consiste en probar todas las funcionalidades de la clase ImageProcessor. Para ello, en vez de ejecutar el código de la clase, se pone directamente para hacer las correcciones pertinentes.

Primero se cargan todas las librerías:

```python
%matplotlib inline
import cv2
import numpy as np
from matplotlib import pyplot as plt
from line import Line
from processor import ImageProcessor
from tracker import Tracker
```

*Figura 39: Importación de librerías*

Y se instancian las variables necesarias para todo el procesamiento de la imagen, como se muestra a continuación:

```
# Config camara raspberry en /dev/video0
camera = cv2.VideoCapture('/dev/video0', cv2.CAP_V4L)
camera.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
processor = ImageProcessor((640, 480), 20)

# Define the average positions for the left and right lanes.
averageLeft = np.poly1d(np.array([-0.3756, 292.7]))
averageRight = np.poly1d(np.array([0.4277, 348.6]))
base_path = "/home/rhiguera/robocar/pruebas/computer_vision/log"
rutas_imagenes = [f"{base_path}/frame_{i}.jpg" for i in range(1, 181)]
```

*Figura 40: Inicialización de variables*

El procesamiento de la imagen va a consistir en lo siguiente:

#### *4.3.2.1 Conversión a Frames binarios*

El primer paso en el procesamiento de imágenes es convertir cada frame capturado por la cámara del coche en una imagen binaria. Esta conversión implica resaltar los carriles y eliminar el resto de la información innecesaria.

- Conversión a formato HSV: Primero, se convierte la imagen a formato HSV con OpenCV para filtrar
- Aplicación de un filtro de color: Para esto, existe un script en el repositorio en la ruta pruebas/computer_vision/calibration/hsv_calibration.py, que sirve para ajustar la zona a resaltar con un panel de control:

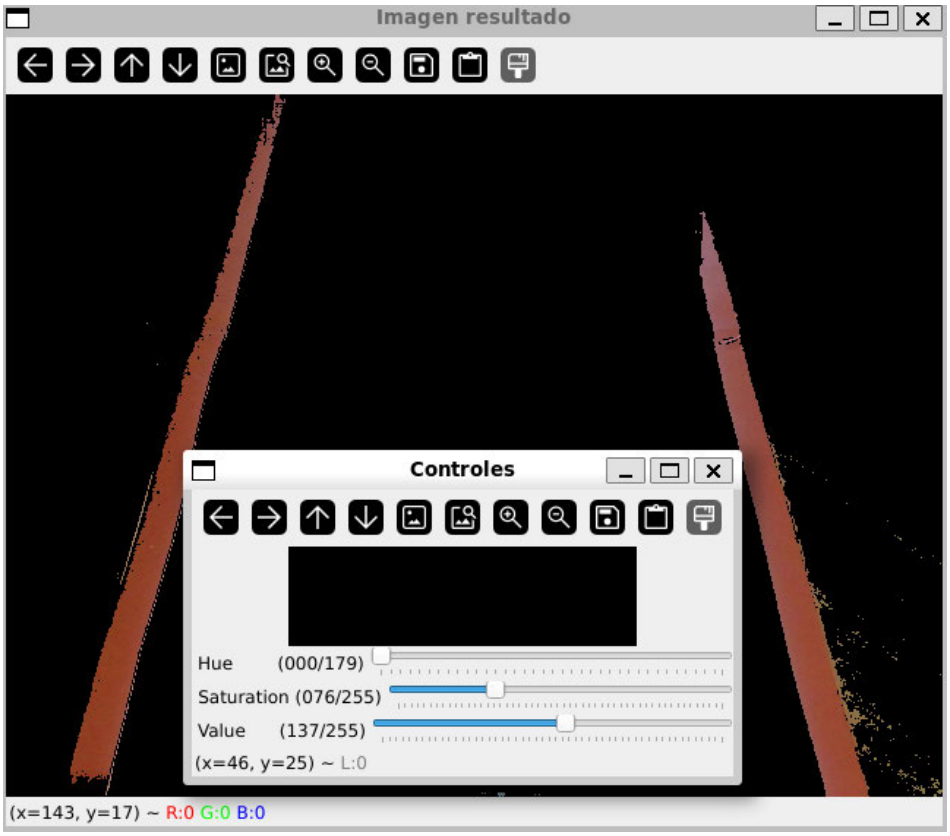


*Figura 41: Aplicación de filtro de color*

- Umbralización: Con la imagen ya filtrada, bastaría con aplicar la umbralización para que descarte únicamente los frames negros.

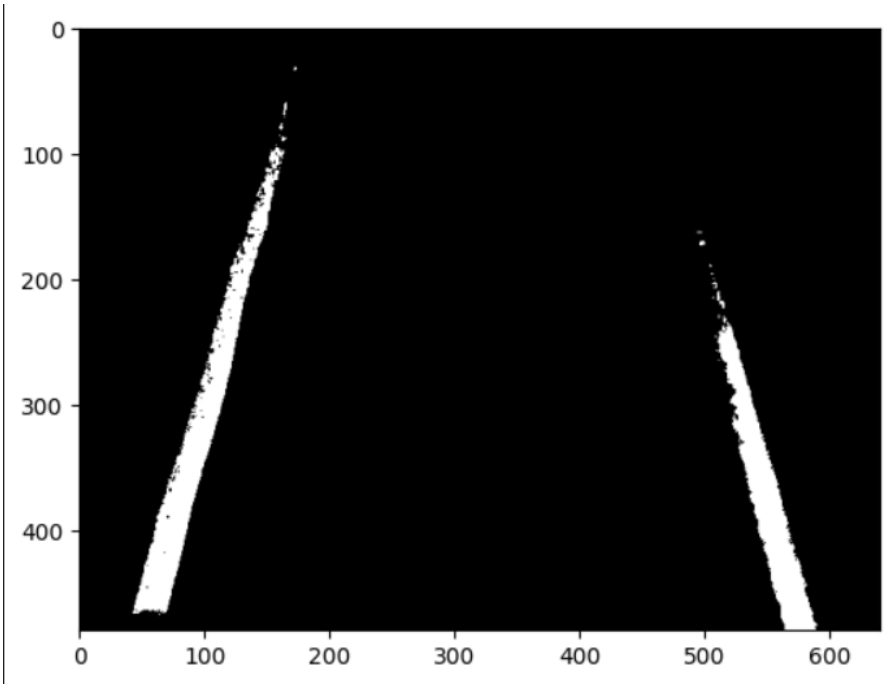


*Figura 42: Frame binario*

#### *4.3.2.2 Detección de Bordes usando el Algoritmo de Canny*

El siguiente paso es detectar los bordes en la imagen binaria. Utilizamos el algoritmo Canny para esto, ya que es un método eficaz para encontrar bordes en una imagen.

- Reducción de Ruido: Aplicamos un filtro Gaussian Blur para suavizar la imagen y reducir el ruido, lo que ayuda a mejorar la detección de bordes.
- Detección de Bordes: Aplicamos el algoritmo Canny, que detecta bordes en la imagen calculando el gradiente de intensidad. El resultado es una imagen donde los bordes son claramente visibles.

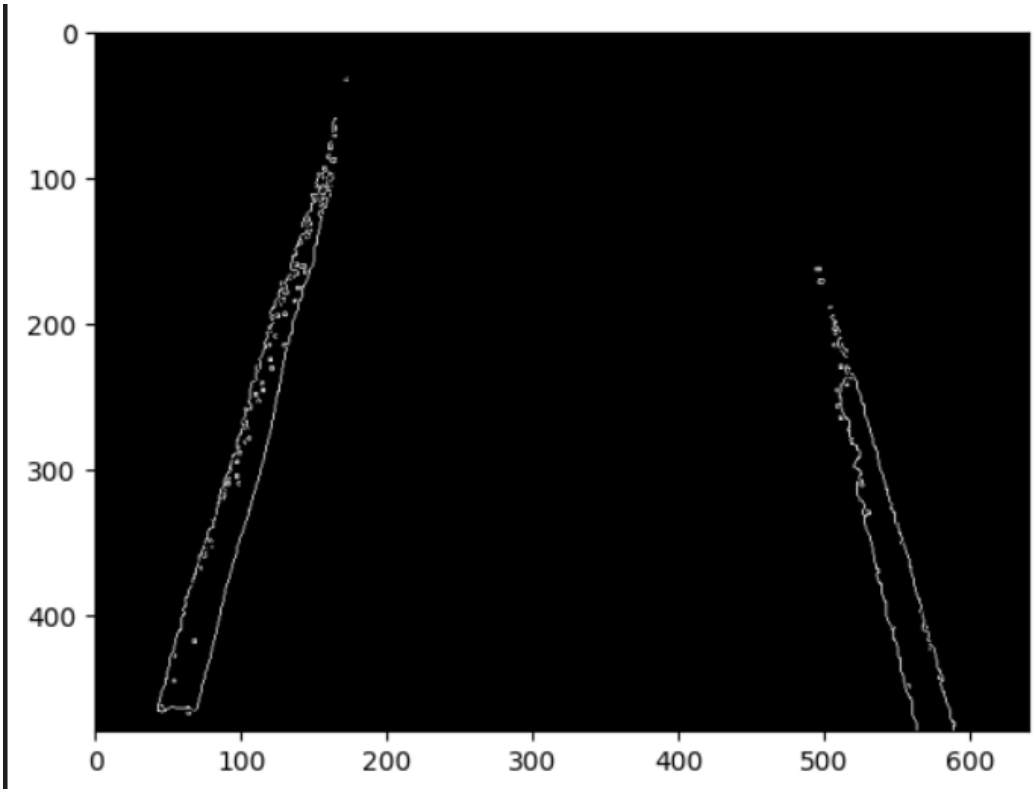


*Figura 43: Filtro de Canny*

#### *4.3.2.3 Selección de la Región de Interés*

Antes de proceder a la detección de líneas, se selecciona la región de interés del frame sobre la que analizar los carriles. En este caso, se hace a través de dos atributos de la clase ImageProcessor: self.roiX y self.roiY.

```
roi = processor.doRegionOfInterest(frames_array[0])
plt.imshow(roi)
plt.show()
✓ 0.2s
```

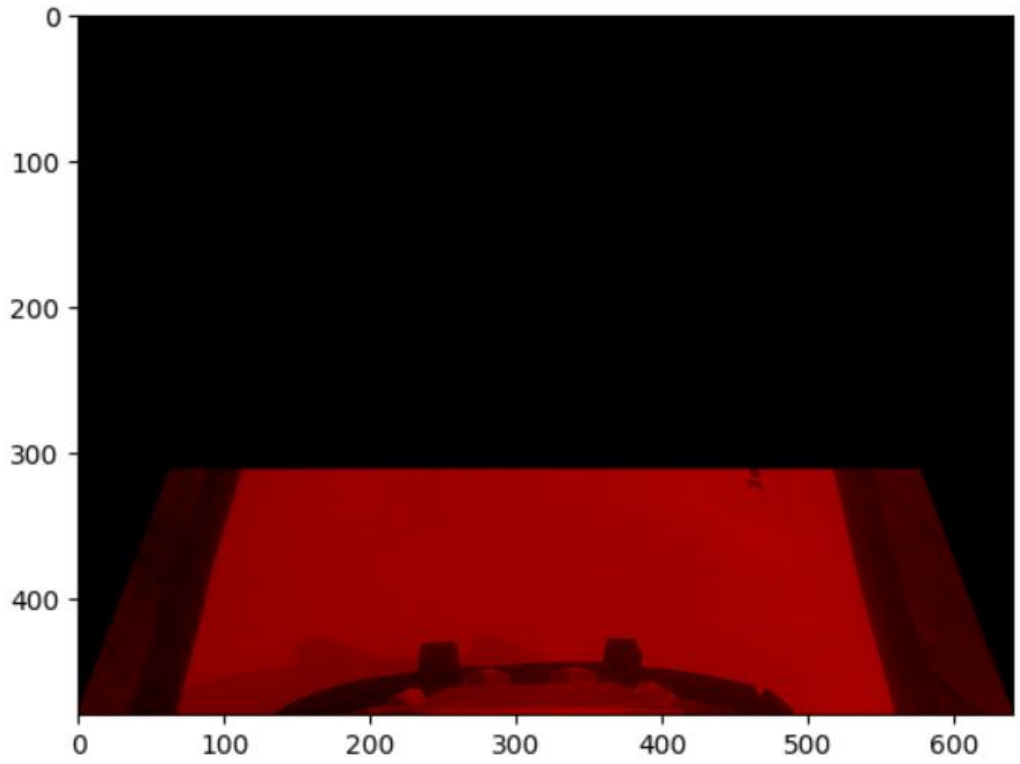


*Figura 44: Región de Interés*

#### *4.3.2.4 Detección de Líneas con la Transformada de Hough*

Finalmente se utiliza la Transformada de Hough para detectar líneas en la imagen de bordes. Este método es ideal para detectar líneas rectas en una imagen.

- **Transformada de Hough**: La función cv2.HoughLinesP en OpenCV se usa para detectar líneas en la imagen. Esta función devuelve una lista de líneas detectadas, donde cada línea está representada por sus puntos de inicio y fin.
- **Filtrado de Líneas**: Filtramos las líneas detectadas para separar las líneas del carril izquierdo y derecho basándonos en distintos parámetros que dependen del escenario.

#### *4.3.2.5 Obtención de carriles en función del escenario*

*Tabla 3: Obtención de carriles*

| Escenario | Algoritmo de filtrado |
|---|---|
| Detección de dos líneas | El extremo más cercano al robot de la línea detectada debe estar en el cuadrante correspondiente. Si está en el cuadrante izquierdo del frame, será el carril izquierdo, y viceversa. |

| | |
|---|---|
| Detección de un único carril | La pendiente de la línea detectada indicará si es carril izquierdo o derecho. Si la pendiente es negativa, será carril derecho y si la pendiente es positiva será carril izquierdo |
| No se detectan carriles | Este caso no debería ocurrir, aunque es posible que debido al entorno y otras circunstancias, algún frame salga borroso y sea imposible detectar las líneas. Se hace una estimación con el filtro de Kalman para, por lo menos, seguir la curvatura según los frames anteriores |

#### *4.3.2.6 Obtención de carriles centrados del coche*

Para un posterior cálculo del ángulo de dirección con el que debe girar el coche, hay que obtener las rectas centradas. Esto es, tomar un fotograma donde se visualicen los carriles rectos y completamente centrados:

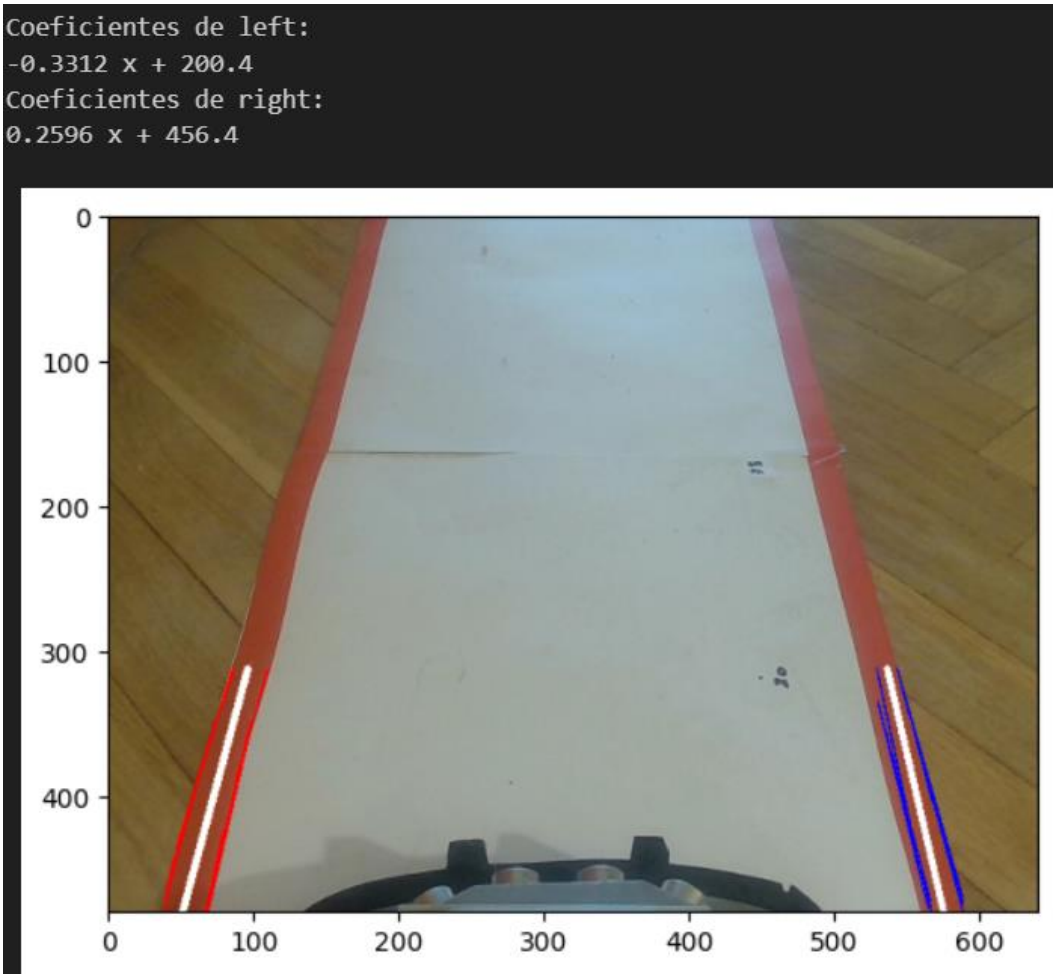


*Figura 45: Carriles centrados*

En este frame, se pueden ver los coeficientes de ambas curvas y las líneas pintadas en blanco. Las líneas más finas azules y rojas son todas las líneas detectadas por la transformada de Hough,

con las cuales después se calcula la media para sacar las rectas finales. Estos valores de las rectas se guardarán en los atributos averageLeft y averageRight de la clase ImageProcessor.

#### *4.3.2.7 Obtención del Offset de los carriles*

Para calcular el error, se sigue el mismo procedimiento para detectar líneas. Una vez tengamos las nuevas líneas que definen los carriles, se calcula la diferencia entre los puntos más altos de la región de interés entre los carriles centrados y las rectas detectadas en el frame:

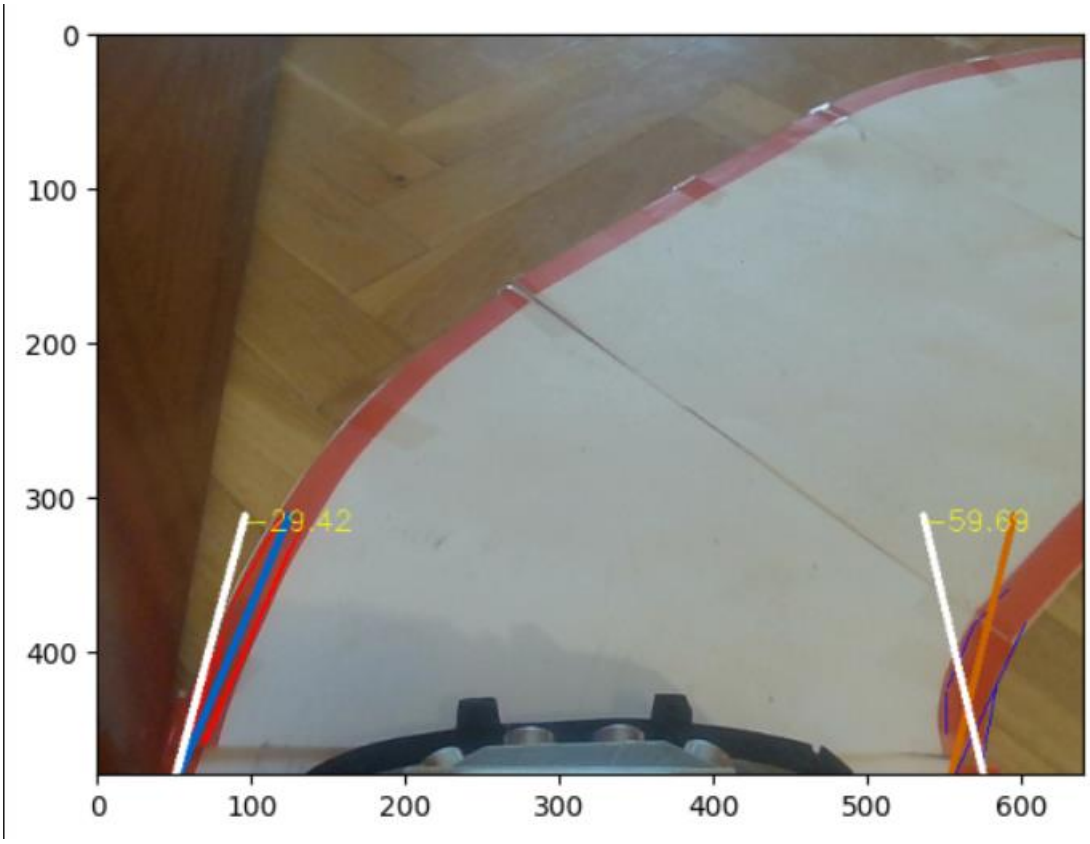


*Figura 46: Frame con Offset*

Como se puede observar, para un mayor entendimiento del cálculo del Offset, se hace sobre un frame donde existe una curva a la derecha. Se obtienen los resultados -29.42  -59.69 para los carriles izquierdo y derecho, respectivamente.

```
# Calculates the error on both lanes.
y0 = processor.roiY[1] * processor.h
leftError = averageLeft(y0) - processor.left.poly(y0)
rightError = averageRight(y0) - processor.right.poly(y0)
averageError = (leftError + rightError) / 2
```

*Figura 47: Cálculo del error*

Este error, posteriormente, servirá para indicar al robot el ángulo de giro que debe realizar para seguir el carril.

#### *4.3.2.8 Obtención del ángulo de giro con Controlador PID*

Se implementa un controlador Proporcional-Integral-Derivativo para obtener el ángulo de giro necesario para mantener el robot alineado con los carriles. Para ello, se inicializan variables clave: el error promedio sumError y el error previo prevError, donde se usa la variable previamente calculada averageError.

Se definen las constantes PID kp, ki y kd para los componentes Proporcional, Integral y Derivativo, respectivamente. Los términos PID se calculan de la siguiente manera:

- Proporcional (P): Se calcula como kp * averageError, donde kp es la constante proporcional.
- Integral (I): Se actualiza sumando averageError * ki * self.dt a self.sumError, donde ki es la constante integral. Este término acumula el error a lo largo del tiempo.
- Derivativo (D): Se calcula como (averageError – self.prevError) * kd / self.dt, donde kd es la constante derivativa. Este término responde a la tasa de cambio del error, ayudando a no hacer giros bruscos en las ruedas

#### *4.3.2.9 Obtención de carriles con Kalman*

La obtención de la estimación de los carriles se hace a través del script tracker.py, que define una clase Tracker. Esta clase implementa un filtro de Kalman 1-D para rastrear una línea. El filtro de Kalman es utilizado aquí para estimar y predecir la posición y la velocidad de una línea a lo largo del tiempo, basándose en mediciones ruidosas. La implementación utiliza la biblioteca filterpy para el filtro de Kalman y numpy para operaciones matemáticas. A continuación se describen los atributos principales usados por el filtro de Kalman:

- **dt**: Intervalo de tiempo entre actualizaciones, utilizado para modelar el ruido del proceso
- **dim_x=4**: Dimensiones del estado (posición y velocidad en dos dimensiones).
- **dim_z=2**: Dimensiones de la medición (pendiente y -x intercept de la línea).
- **x**: Estado inicial (posición y velocidad inicializadas a 0).
- **P**: Matriz de covarianza inicial, configurada con una gran incertidumbre inicial (uncertaintyInit).
- **Q**: Matriz de covarianza del ruido del proceso, modelada como ruido blanco discreto.
- **R**: Matriz de covarianza del ruido de medición, con valores bajos indicando mediciones relativamente precisas.
- **H**: Matriz de función de medición, mapea el estado al espacio de medición.
- **F**: Matriz de transición de estado, modela la dinámica del sistema.

**Método *add (self, poly)***: Actualiza el filtro de Kalman con una nueva medición (un polinomio que representa la línea) y devuelve el polinomio filtrado. Utiliza el método predict para estimar el estado actual antes de la medición, y luego update para corregir la estimación con la nueva medición. La medición es un polinomio con coeficientes que representan la pendiente (m) y el intercepto (b). El método devuelve un nuevo polinomio que representa la línea estimada después de la corrección, tal y como se muestra en el siguiente frame:

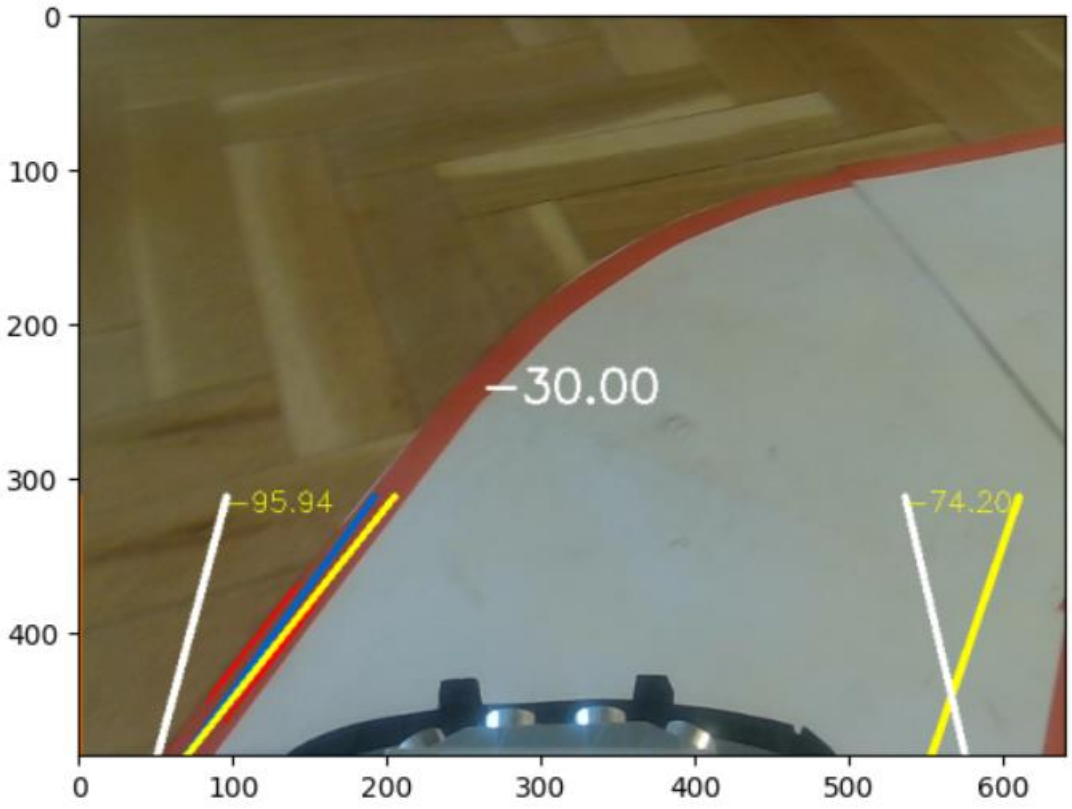


*Figura 48: Frames con Kalman*

Las líneas amarillas representan las rectas obtenidas con el filtro de Kalman. Un detalle importante es que en este escenario, el carril derecho no ha sido detectado. Sin embargo, con el filtro de Kalman se ha conseguido una estimación cercana al acierto. El carril izquierdo por otro lado, tanto el detectado como el estimado siguen una línea muy similar.

### 4.3.3 Resumen del algoritmo de obtención de carriles

*# Inicializamos el filtro de Kalman para las mediciones:*

*left_tracker = Tracker ()*

*right_tracker = Tracker ()*

*average_left = getAverageLeft ()*

*average_right = getAverageRight ()*

*# Obtenemos carriles con los frames que vayan llegando:*

*binary_frame = Filtrar_imagen_color (frame)*

*canny = bordes_canny (binary_frame)*

*lines = transformada_hough (canny)*

*left, right = filtrado_lineas (lines)*

*# Añadimos líneas al modelo de Kalman y obtenemos la estimación:*

*kalman_left = leftTracker.add (left)*

*Kalman_right = rightTracker.add (right)*

*If not exists (left): # Si no existe el carril Izquierdo en el frame actual, cogemos el de Kalman*

*left = kalman_left*

*If not exists (right):*

*right = kalman_right*

*# Calculamos el error a partir de las líneas obtenidas*

*y0 = getTopROI ()*

*Left_error = average_left (y0) – left (y0)*

*Right_error = average_right (y0) – right (y0)*

### 4.3.4 Integración del Algoritmo en ROS2

Una vez implementado todo, se tiene que integrar en el framework de ROS2. Para ello hay que hacer lo siguiente:

- Crear un nuevo nodo que recoja los frames, los procese, y publique el resultado (ángulo de giro) en un nuevo topic. Lo llamaremos /lane_info.
- Modificar el nodo car_control_node de manera que admita conducción autónoma, siguiendo el siguiente diagrama de estados:

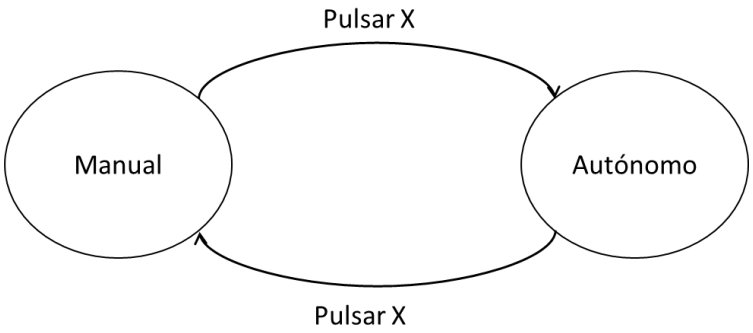


*Figura 49: Diagrama de estados del robot*

En modo autónomo, el robot sólo escuchará del topic /lane_info. La velocidad es constante de 18 cm/s.

En la siguiente página se muestra el diagrama de nodos actualizado con la implementación de conducción autónoma.

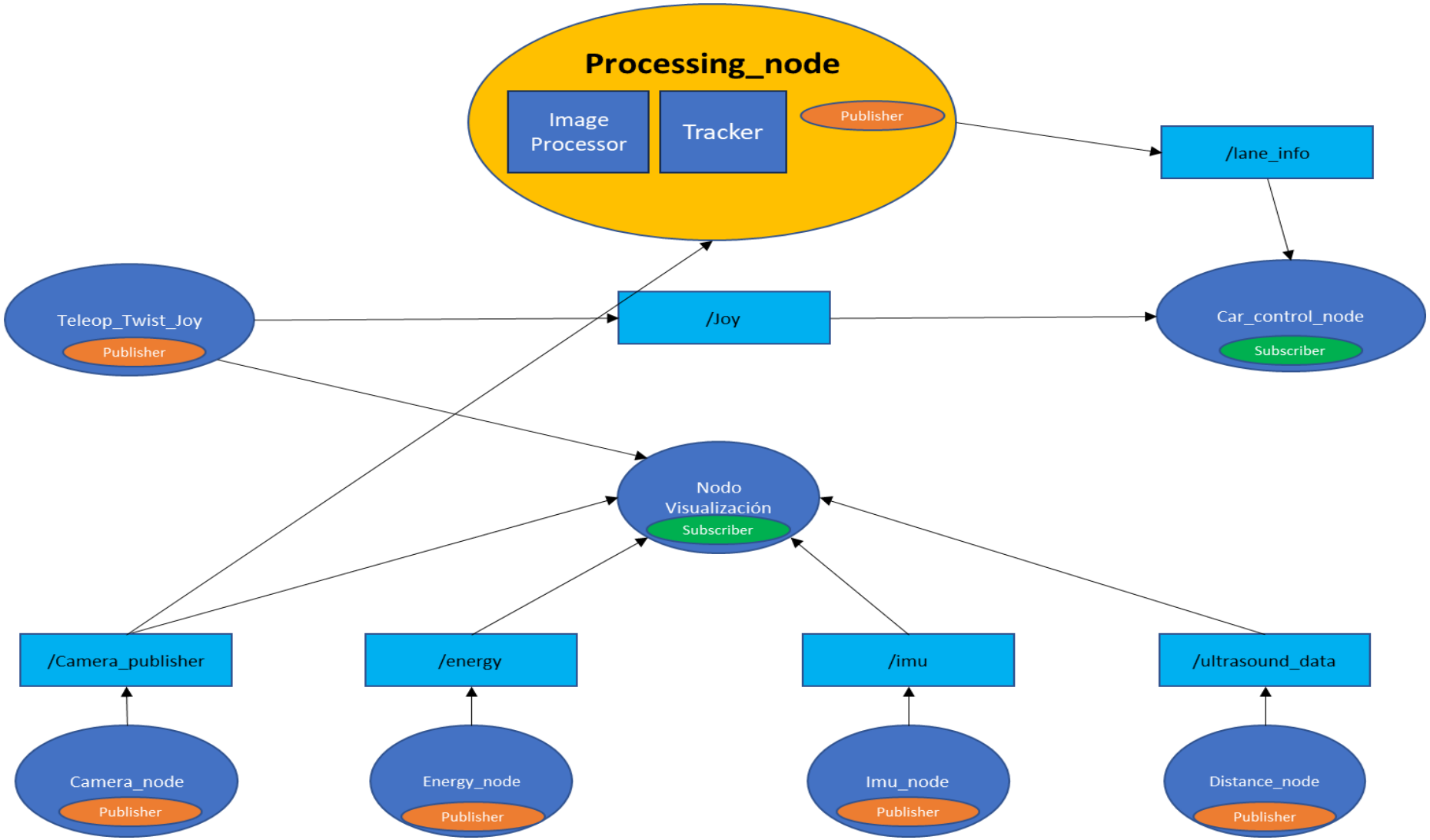


*Figura 50: Diagrama de nodos ROS*

## 4.4 Pruebas y Validación

### 4.4.1 Preparación de las pruebas

Para las pruebas de este modelo, se utiliza cartulinas DIN-A3 y cinta aislante roja, según se muestra en la siguiente imagen:


*Figura 51: Trazada del circuito*

Las pruebas consisten en conducir el coche manualmente de manera que siga los carriles como debería hacer en la modalidad de conducción autónoma. De esta manera, a medida que va avanzando el robot, se van guardando frames de ejemplo de una vuelta completa al circuito. Estos frames se guardan en la ruta /pruebas/computer_vision/log. Para ello, hay que poner a punto el escenario, realizando lo siguiente:

- Modificar camera_node, de manera que guarde en la ruta ya mencionada los frames que vaya enviando al topic /camera_image. Los frames tendrán de nombre frame_x.jpg.
- Ejecutar nodo teleop_twist_joy, camera_node y car_control_node, nodos imprescindibles para realizar las pruebas.
- Subir los resultados al repositorio

### 4.4.2 Pruebas

En este apartado se exponen distintos frames según las distintas curvas mostradas en la imagen anterior, mostrando un frame de inicio y fin para cada una de las distintas curvas o rectas definidas. En las siguientes páginas se muestran, para cada punto amarillo definido, cómo se comporta el coche.

#### *4.4.2.1 Recto*

**Comienzo**

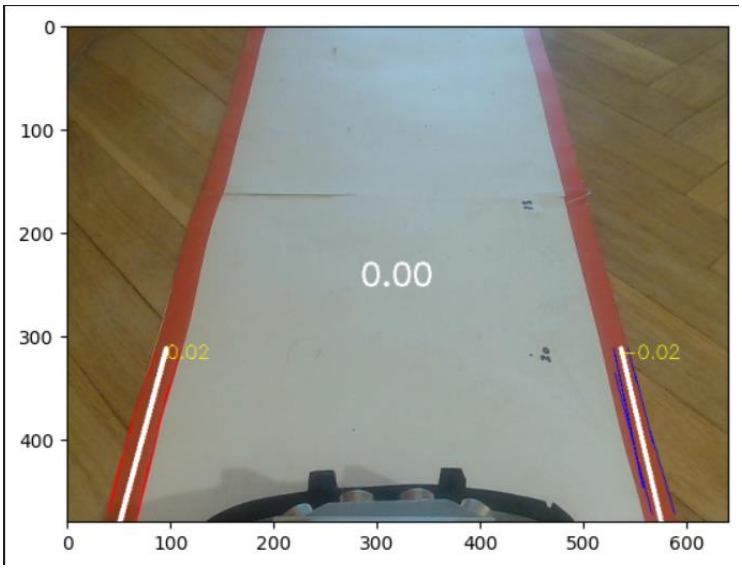


*Figura 52: Comienzo frame recto*

**Fin**

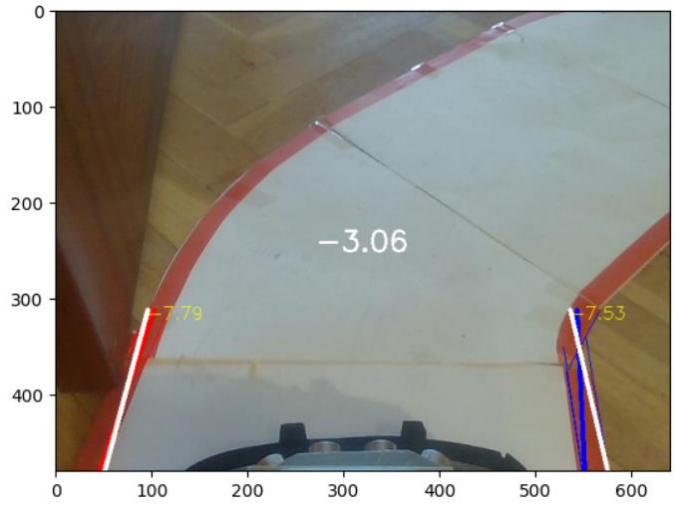


*Figura 53: Fin frame recto*

#### *4.4.2.2 Curva Derecha Máxima*

**Inicio de la curva**

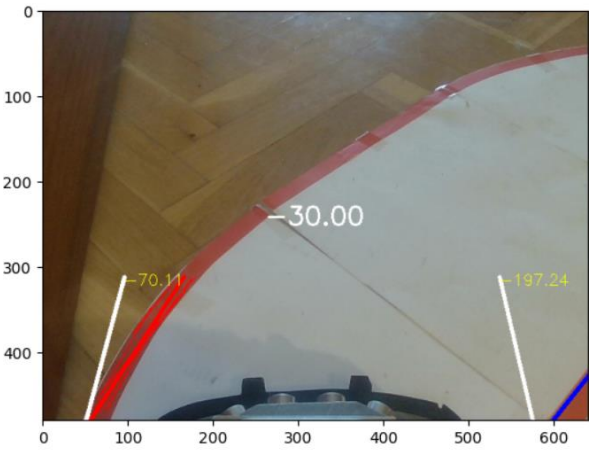


*Figura 54: Inicio frame curva derecha*

A continuación se muestra un frame que verifica la resiliencia del algoritmo, ya que no detecta carril izquierdo ni derecho:

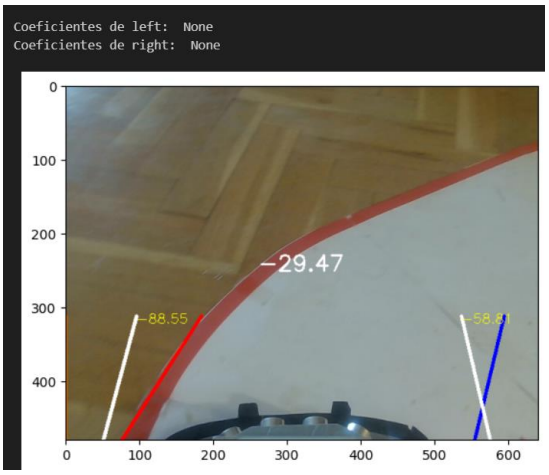


*Figura 55: Frame sin líneas detectadas*

Sin embargo, como está usando las rectas obtenidas por el filtro de Kalman, tiene una estimación precisa (el carril derecho es cierto que está bastante desviado).

**Fin de la curva**

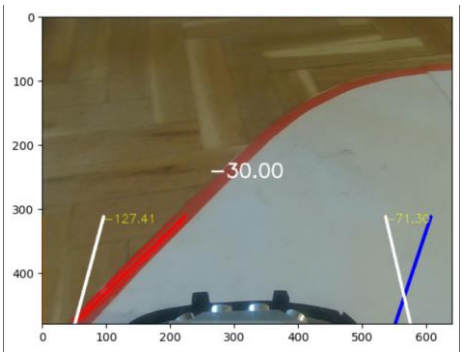


*Figura 56: frame de fin de curva derecha*

#### *4.4.2.3 Curva Derecha Leve*

**Inicio de la curva**

Lo que en principio era una curva leve, según se ve en los siguientes frames es una curva donde el coche tiene que realizar de nuevo un giro máximo a la derecha. Este resultado se debe a que el circuito es bastante estrecho y en realidad, aunque parezcan curvas más ligeras, el coche sólo tiene 30º de ángulo de giro, lo cual es bastante poco.

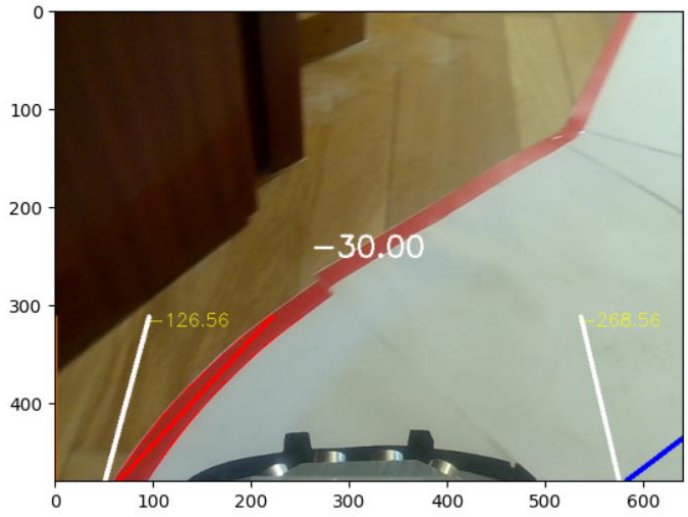


*Figura 57: Inicio frame curva leve*

**Fin de la curva**

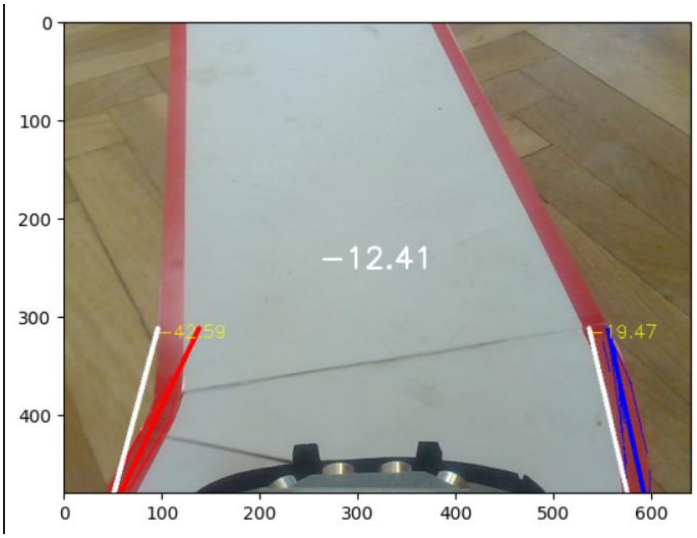


*Figura 58: Fin frame curva leve*

#### *4.4.2.4 Curva Izquierda Leve*

**Inicio de la curva**

Justo el siguiente frame pertenece al de la curva leve a la izquierda:

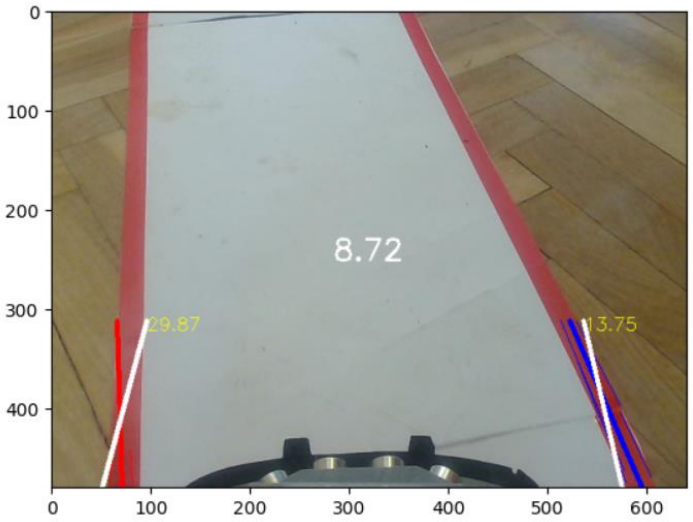


*Figura 59: Inicio frame curva izquierda*

**Fin de la curva**

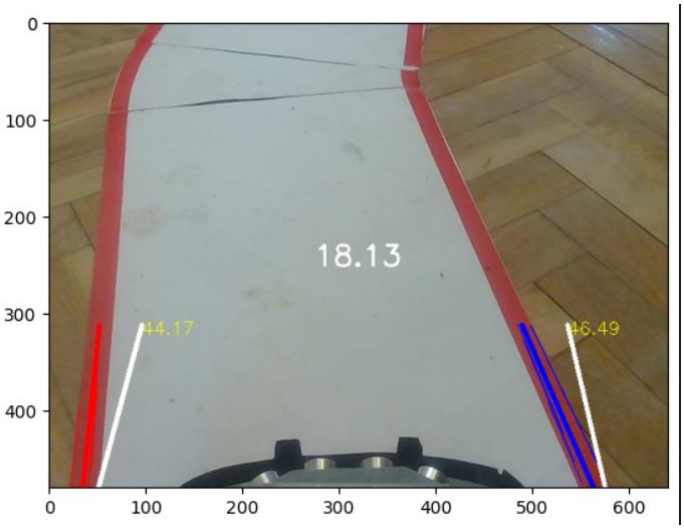


*Figura 60: Fin frame curva izquierda*

### *4.4.2.5 Curva Derecha*

**Inicio de la curva**

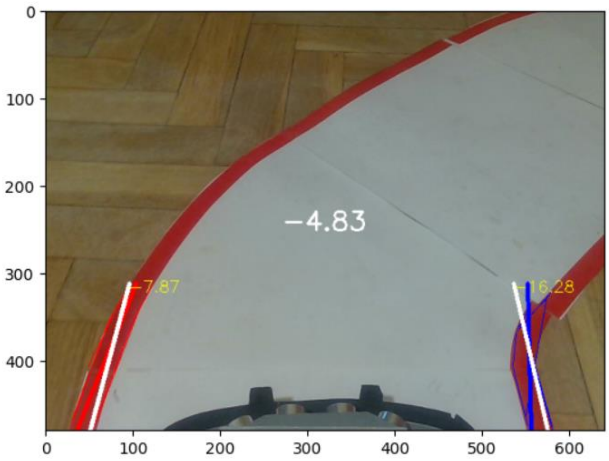


*Figura 61: Curva derecha*

**Dentro de la curva**

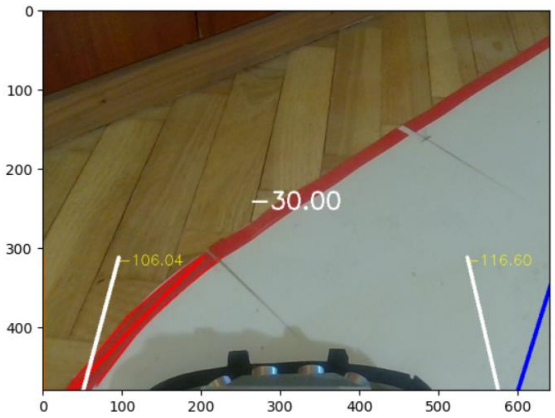


*Figura 62: frame dentro de curva derecha*

**Fin de la curva**

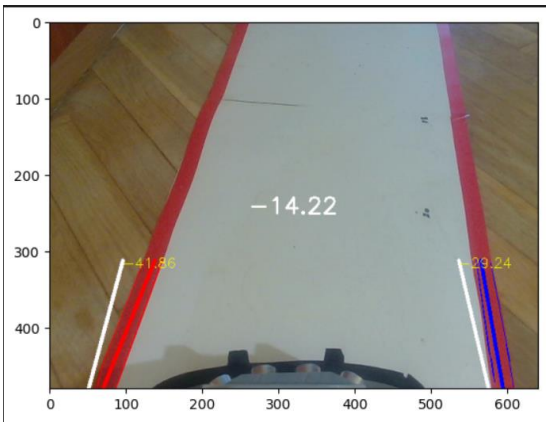


*Figura 63: Fin frame curva derecha 2*

# 5 Resultados

## 5.1 Resultados de la Implementación

Tras las pruebas en el circuito, el sistema, aunque capaz de identificar y seguir carriles en condiciones ideales, enfrentó problemas en segmentos del circuito donde la iluminación cambiaba significativamente. En particular, en ciertas curvas y tramos rectos con cambios abruptos de luz, la detección de carriles fue inconsistente, afectando la precisión y la estabilidad del seguimiento. Se ha mitigado dentro de lo posible todo esto, añadiendo un foco con luz artificial cerca del circuito.

Durante las pruebas, se notó que el sistema no detectaba los carriles correctamente en varios frames debido a las condiciones de iluminación no óptimas. Sin embargo, el uso del filtro de Kalman permitió hacer estimaciones bastante precisas de los carriles, lo que ayudó a mantener el seguimiento del camino aunque visualmente no se detectaran los carriles en momentos puntuales. Esto demuestra una capacidad de resiliencia del sistema, pero también destaca su dependencia de condiciones lumínicas adecuadas para funcionar eficazmente.

## 5.2 Análisis de Datos

Las pruebas fueron meticulosamente diseñadas para evaluar cómo el sistema manejaba diferentes segmentos del circuito, incluyendo tramos rectos y curvas bajo variadas condiciones de iluminación.

### 5.2.1 Precisión en la detección de carriles

- **Rectas:** En partes del circuito con iluminación uniforme, el sistema mantuvo al robot centrado en el carril con un margen de error bajo.
- **Curvas:** El robot tuvo dificultades en curvas, especialmente cuando giraba hacia zonas donde la iluminación cambiaba drásticamente, lo que afectaba la detección de carriles.

### 5.2.2 Estabilidad del seguimiento

A pesar de no detectar carriles en algunos frames debido a la variación de iluminación, el filtro de Kalman proporcionó una estimación bastante correcta de la posición del carril, permitiendo al sistema mantener un seguimiento relativamente estable.

### 5.2.3 Adaptabilidad a condiciones cambiantes

El sistema demostró ser menos fiable en condiciones de iluminación no óptimas, donde los cambios significativos de luz afectaron de forma negativa a la capacidad del robot para detectar y seguir los carriles adecuadamente.

## 5.3 Discusión de Resultados

El análisis de los datos recogidos mostró que, mientras el sistema tiene la capacidad de seguir carriles con una precisión razonable bajo condiciones controladas, su rendimiento decrece notablemente con la variabilidad de la iluminación. Esta sensibilidad a la luz plantea dudas sobre la fiabilidad del sistema en aplicaciones prácticas fuera de un entorno controlado.

La dependencia del filtro de Kalman para compensar la detección visual insuficiente subraya la necesidad de mejoras técnicas. Aunque este enfoque es útil para mitigar problemas de detección momentáneos, confiar excesivamente en la estimación puede llevar a errores en situaciones donde los cambios ambientales son más pronunciados y menos predecibles.

Para aumentar la robustez del sistema, sería beneficioso integrar técnicas avanzadas de procesamiento de imágenes que puedan adaptarse mejor a las variaciones de iluminación. Además, mejorar la calibración de la cámara y considerar el uso de tecnologías de sensores más sofisticadas podría proporcionar una detección más consistente en diferentes condiciones ambientales.

# 6 Aspectos Éticos, Legales y Profesionales

## 6.1 Aspectos Éticos

La ética en el desarrollo de vehículos autónomos es fundamental para garantizar la seguridad y el bienestar de los usuarios y de la sociedad en general. Este proyecto se ha desarrollado teniendo en cuenta los siguientes principios éticos:

### 6.1.1 Seguridad

Se ha priorizado la seguridad en el diseño y la implementación del sistema autónomo. Todos los componentes y algoritmos se han probado rigurosamente para minimizar riesgos y garantizar un funcionamiento seguro en todas las condiciones.

### 6.1.2 Privacidad de los Datos

La recolección y procesamiento de datos se realiza respetando la privacidad de los individuos. Los datos obtenidos de los sensores se utilizan exclusivamente para mejorar el desempeño del vehículo y no se comparten con terceros sin consentimiento.

### 6.1.3 Transparencia y Responsabilidad

El desarrollo del proyecto se ha llevado a cabo con transparencia, documentando todas las etapas del proceso. Esto incluye la publicación de resultados y la explicación clara de las decisiones tomadas durante el desarrollo.

## 6.2 Aspectos Legales

El desarrollo de vehículos autónomos debe cumplir con una serie de normativas y regulaciones legales para asegurar que su operación es lícita y segura. Este apartado sigue la línea del primer proyecto de este robot [11].

### 6.2.1 Cumplimiento Normativo

Debido a la naturaleza de tipo prototipo, así como de que se trata de un robot a menor escala de un vehículo autónomo, no resulta necesario alinearse con las las regulaciones locales respecto al uso de vehículos autónomos. Por este motivo se limita la operatividad del sistema a recintos privados en entornos controlados.

### 6.2.2 Propiedad Intelectual, Licencias y Permisos

Se ha respetado la propiedad intelectual de tecnologías y componentes utilizados en el proyecto. Se ha obtenido todas las licencias y permisos necesarios para la construcción y prueba del sistema. Los sensores, herramientas software y otros elementos tecnológicos han sido empleados conforme a las licencias de uso correspondientes, evitando infracciones.

## 6.3 Aspectos Profesionales

En cuanto a los aspectos profesionales, el proyecto se ha llevado a cabo con el máximo nivel de profesionalidad posible y adherencia a buenas prácticas de la industria:

### 6.3.1 Calidad y Rigurosidad Técnica

Se ha implementado una metodología de trabajo estructurada y rigurosa, que incluye el desarrollo en espiral incremental. Esto ha permitido la integración y prueba continuas de los componentes del sistema, garantizando su robustez y fiabilidad.

### 6.3.2 Colaboración y Comunicación

El proyecto ha fomentado un entorno de trabajo colaborativo, con reuniones periódicas y una comunicación efectiva entre todos los miembros del equipo. Esto ha facilitado la resolución de problemas y la mejora continua del proyecto.

### 6.3.3 Desarrollo Sostenible

Se ha procurado que el desarrollo del proyecto sea sostenible, utilizando componentes y tecnologías que minimicen el impacto ambiental. Además, se ha promovido el uso responsable de recursos y la gestión eficiente del presupuesto.

Este enfoque integral en los aspectos éticos, legales y profesionales no solo asegura la viabilidad y legalidad del proyecto, sino que también contribuye a su aceptación social y éxito a largo plazo.

# 7 Conclusión y Líneas Futuras

## 7.1 Conclusiones del Estudio

El estudio de la implementación del sistema de seguimiento de carriles en un robot autónomo ha ofrecido una comprensión detallada de su funcionamiento y eficacia en un entorno controlado. Aunque el sistema ha demostrado ser capaz de mantener la trayectoria del robot dentro de los carriles bajo condiciones de iluminación estables, ha revelado deficiencias significativas bajo variaciones de luz, un factor crítico en entornos reales.

Es importante reconocer que existen numerosas soluciones alternativas en el ámbito de los sistemas de seguimiento de carriles y vehículos autónomos. Cada una de estas soluciones presenta diferentes enfoques, desde sistemas basados en visión computacional avanzada hasta aquellos que integran sensores LIDAR o fusiones de múltiples tecnologías de detección. La implementación actual debe ser vista como una de muchas posibles, y la evaluación comparativa con otras tecnologías es esencial para determinar su viabilidad y eficacia relativa.

Además, la precisión de los sensores y la calibración son elementos críticos que afectan directamente la capacidad del sistema para realizar un seguimiento fiable. Mejoras en estas áreas, especialmente en la robustez del sistema bajo diversas condiciones de iluminación, serían beneficiosas.

En conclusión, aunque este estudio proporciona una perspectiva valiosa sobre el uso de filtros de Kalman y la detección basada en cámaras para el seguimiento de carriles, queda claro que una evaluación más amplia de diferentes soluciones tecnológicas es necesaria. Solo mediante la implementación y prueba de diversas tecnologías se podrá establecer si esta rama de implementación es la más adecuada, o si otras alternativas pudieran ofrecer resultados superiores en aplicaciones prácticas.

## 7.2 Líneas Futuras

A raíz de las evaluaciones realizadas en el estudio del sistema de seguimiento de carriles para el robot autónomo, se han identificado diversas oportunidades para mejorar la implementación y eficacia del sistema. Considerando las limitaciones actuales y el potencial tecnológico disponible, se proponen las siguientes líneas futuras para el desarrollo y optimización del sistema:

- **Mejora de la calidad de la cámara**: Uno de los problemas identificados es la calidad moderada de la cámara utilizada, que puede limitar la precisión en la detección de carriles bajo diversas condiciones de iluminación y distancia. Una actualización a una cámara de mayor resolución y mejor rendimiento en condiciones de baja luz podría mejorar significativamente la capacidad del sistema para detectar y seguir carriles con mayor precisión.
- **Actualización de la Unidad Central de Procesamiento**: La Raspberry Pi 4, aunque adecuada para tareas de procesamiento básico, se queda corta en comparación con alternativas más potentes como la NVIDIA Jetson Nano. La Jetson Nano ofrece una capacidad de procesamiento significativamente mayor, lo que podría traducirse en mejoras en la velocidad y la eficiencia del procesamiento de imágenes y datos sensoriales. Este cambio no solo permitiría implementar algoritmos más complejos de visión artificial y aprendizaje automático, sino que también mejoraría la respuesta del sistema en tiempo real.
- **Integración de sensores adicionales**: Actualmente, el robot está equipado con cuatro sensores de ultrasonidos y un sensor infrarrojo de emergencia. Estos sensores podrían utilizarse de manera más integrada para mejorar la percepción del entorno del robot.

Por ejemplo, los sensores podrían usarse para detectar obstáculos cercanos o para mejorar la precisión del seguimiento de carriles al proporcionar datos adicionales sobre el entorno inmediato del robot.

- **Aumento de la capacidad de procesamiento con la Jetson Nano**: Con la incorporación de la Jetson Nano, sería posible explorar técnicas más avanzadas de inteligencia artificial, como redes neuronales profundas, que requieren un gran poder de cómputo. Esto podría incluir la implementación de sistemas de visión mejorados que sean capaces de interpretar de manera más efectiva y en tiempo real los complejos entornos de conducción.

# 8 Anexos

## 8.1 Códigos de Programación

Todo el código de este proyecto se encuentra en la rama TFG_2 del repositorio:

https://github.com/rubenhigorg/robocar/tree/TFG_2

## 8.2 Esquemas del Robot

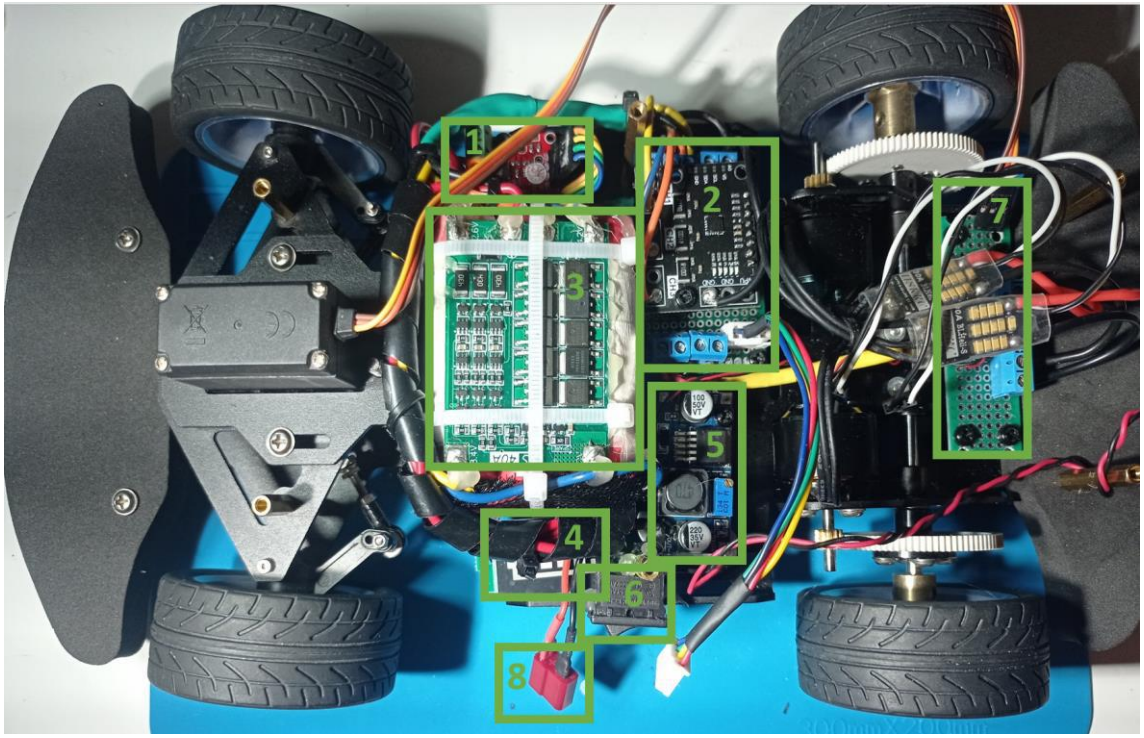


*Figura 64: Fotografía del sistema de baterías*

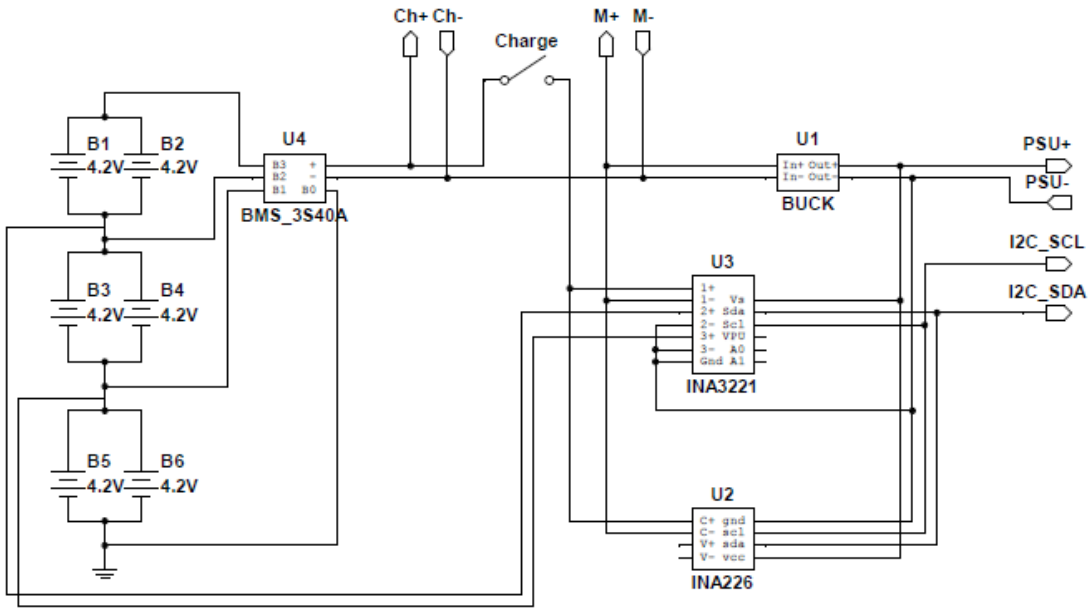


*Figura 65: Esquemáticos de baterías y sensores*

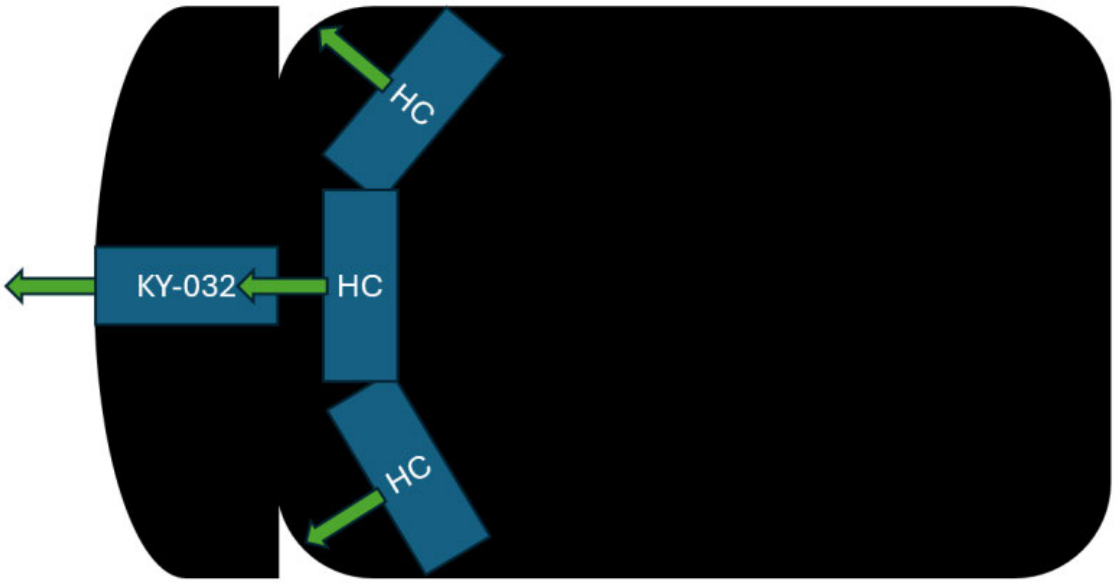


*Figura 66: Esquema de sensores de distancia*

*Figura 67: Fotografía del robot*

# 9 Índice de Figuras

# 10 Índice de tablas

# 11 Bibliografía


[1] T. B. Nguyen, «Evaluation of Lane Detection Algorithms based on an Embedded Platform,» Chemnitz, 2017.

[2] S. Ulbrich, «Towards a Functional System Architecture for,» p. 16, 2017.

[3] J. C. &. T. M. M. McCall, « Video-based lane estimation and tracking for driver assistance: Survey, system, and evaluation.,» *IEEE Transactions on Intelligent Transportation Systems, 7,* pp. 20-37, 2006.

[4] S. S. N. &. S. P. Waykole, Review on Lane Detection and Tracking Algorithms of Advanced Driver Assistance System, Sustainability, 2021.

[5] C. Lee y J. Moon, *Robust Lane Detection and Dtracking for Real-Time Applications,* IEEE Trans. Intell. Transp. Syst., 2018.

[6] P. Wang y C. Chan, «A reinforcement learning based approach for automated lane change maneuvers. In,» IEEE, Changshu, China, 2018.

[7] H. Saleem, «Steering Angle Prediction Techniques for Autonomous Ground Vehicles: A Review,» IEE Access.

[8] Z. Yang, «Accurate and robust vanishing point detection method in unstructured road scenes,» IEEE, 2019.

[9] A. B. G. E. a. M. B. M. AbdElrahman, «Vision-based road tracking of wheeled mobile robot,» Proc. Int. Conf. Aerosp. Sci. Aviation Technol., 2013.

[10] J. M. Snider, «Automatic Steering Methods for Autonomous Automobile Path Tracking,» *Robotics Institute, Carnegie Mellon University,* 2009.

[11] K. R. y. R. Higuera, *Diseño y Construcción de Robocar,* Madrid, 2024.

[12] T. Litman, «Autonomous Vehicle Implementation Predictions,» *Victoria Transport Policy Institute,* p. 48, 2022.