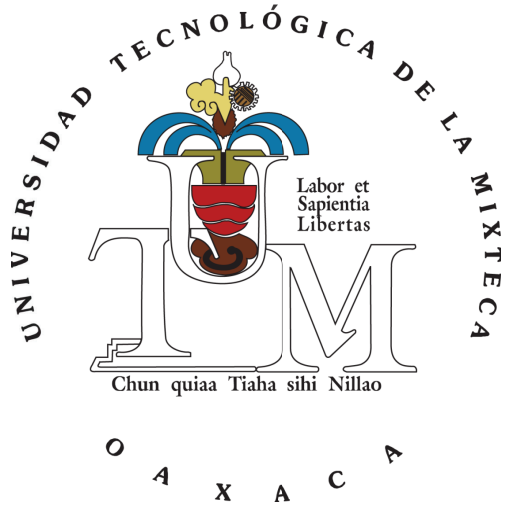


UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

"DISEÑO E IMPLEMENTACIÓN DEL SISTEMA DE CONTROL LATERAL PARA UN ROBOT MÓVIL AUTONOMOS MINI V2"

Tesis para obtener el título de:

INGENIERO MECÁNICO AUTOMOTRIZ

Presenta:
Kevin Fulgencio Guzmán Durán

Director de Tesis:
Dr. José Aníbal Arias Aguilar

Co-director de Tesis:
Dr. Oscar David Ramírez Cárdenas

H. Cd. de Huajuapan de León, Oaxaca, México, Febrero de 2023

Dedicatoria

Dedicada a mi madre Bricia, a mis hermanas Alexia, Sarahy y Annett, a mi familia y a mis amigos.

Agradecimientos

Quiero comenzar agradeciéndole a mi familia; mi madre Bricia Durán Arellano, mi hermana Alexia Guzmán Durán, Sarahy Dueñas Durán y Annett Dueñas Durán, que me apoyaron desde el inicio y siempre estuvieron a mi lado.

Agradezco a mis asesores Dr. José Anibal Arias Aguilar y el Dr. Oscar David Ramírez Cárdenas por su apoyo y conocimientos en la realización de este trabajo. Le agradezco su tiempo, sus enseñanzas y consejos, así como la confianza en mi capacidad para realizar este proyecto. Al Ing. Omar Roberto Cruz Ortega por recibirme en su laboratorio, gracias a su paciencia, aportes y asistencia pude alcanzar los resultados presentados en este trabajo.

A los sinodales, Dr. Rosebet Miranda Luna, Dr. Arturo Hernández Méndez y Dr. Fermín Hugo Ramírez Leyva, por su tiempo para la revisión de este trabajo, por sus consejos y enseñanzas.

Agradecimiento a la fundación Dr. Hood por el apoyo brindado, a mis amigos Eduardo, Guadalupe, Santiago, Hugo, Maxvell y demás amigos, su amistad es de lo más preciado que obtuve en esta licenciatura. Gracias por su apoyo, he aprendido mucho de ustedes y espero seguir haciéndolo.

Finalmente, a la Universidad Tecnológica de la Mixteca por permitirme desarrollar mis estudios de licenciatura y este trabajo de tesis. A los profesores que supieron compartir su enseñanza y sabiduría.

Resumen

Este documento detalla el diseño y la implementación de un sistema de control para el movimiento lateral de un robot móvil. Se propone una nueva forma para el control lateral, el cual se basa en el área de error de posición obtenida de forma visual por la cámara Intel Real Sense a bordo del robot AutoNOMOS mini V2. El área de error, es la superficie de que tan separado se encuentra el centro del robot con el centro de los carriles. Se describe el procedimiento para la detección de las líneas del carril, así también se describe un enfoque para obtener el área de error. Finalmente, se realiza el control lateral verificando su rendimiento en el simulador Gazebo y en una pista real, demostrando que el robot mantiene su posición dentro de su carril durante todo el trayecto.

Índice general

Índice de figuras

Capítulo 1

Introducción

La investigación sobre los vehículos autónomos se está expandiendo rápidamente. La consultora Gartner [1] predice que la tecnología de este tipo de automóviles creará miles de nuevos empleos y producirá millones de dólares para la industria del automóvil en el año 2050. Por ello, empresas como Tesla Motors, Waymo, Volkswagen y Mercedes Benz confían en el futuro desarrollo de este tipo de coches.

Los vehículos autónomos son sistemas complejos que pueden percibir su entorno y decidir cómo moverse sin intervención humana. Esta tecnología puede desglosarse en tres componentes fundamentales: percepción, planificación y control [2]. La visión por ordenador, los sensores LIDAR y RADAR, el Sistema de Posicionamiento Global (GPS) y otros sistemas de sofisticación comparable conforman el sistema de percepción.

La visión por ordenador es el núcleo de los algoritmos de detección, y las cámaras son el dispositivo que más se acerca a la forma en que las personas ven su entorno. Aunque la tecnología LIDAR y RADAR se utiliza activamente en el desarrollo de tecnologías de detección, las cámaras proporcionan una forma más eficiente y económica de recoger datos del entorno. La aplicación de comportamientos de conducción se ve directamente afectada por la identificación de carriles, que es un componente crítico del desarrollo de vehículos inteligentes. Estas funciones mejoran considerablemente la eficacia de la conducción y la seguridad del vehículo en el entorno. Por ejemplo, basándose en el carril detectado, es posible determinar una dirección de conducción efectiva para el vehículo inteligente y proporcionar la posición precisa del vehículo dentro del carril.

La detección automática de carriles para asistencia al conductor es un tema estudiado para el desarrollo de ADAS (Advanced Driver Assistance Systems) y otros sistemas de aplicación por su importancia en la seguridad de los conductores y peatones en vías vehiculares. Se han propuesto varios enfoques en los últimos años para resolver esta tarea y aumentar el reconocimiento de la tasa en comparación con los conductores humanos.

Recientemente se han propuesto diferentes algoritmos basados en modelos para la detección de carriles. En este sentido, hay cuatro fases principales en los procedimientos basados en modelos según [3]: extracción de características de carril, reducción de ruido, ajuste del modelo y detección de carril de vehículos.

Este documento describe el desarrollo de una metodología para la detección y mantenimiento dentro del carril para un robot móvil tipo Ackermann. Incluye las etapas de detección de carril utilizando el método de ventanas deslizantes. También la descripción de los controladores laterales Pure-Pursuit y Stanley. Por último se muestran los resultados obtenidos en dichos experimentos.

1.1. Antecedentes

Actualmente existen diferentes estrategias para controlar un vehículo autónomo. Además del controlador convencional PID (Proportional Integral Derivative), los controladores dinámicos comprenden el LQR (Linear Quadratic Regulator) y el MPC (Model Predictive Control) entre otros [4]. No obstante, basándose en el modelo cinemático simplificado de la bicicleta, existen diversas estrategias para controlar el movimiento lateral [5], [6], [7]. El objetivo es minimizar la diferencia entre un parámetro deseado y la condición actual, como la velocidad, la posición, la trayectoria, ángulo, etc.

Lombard et al. [8] proponen un nuevo enfoque geométrico para definir el ángulo de la rueda en función del estado del automóvil contribuyendo al problema de seguimiento de la trayectoria. Así lograron un análisis numérico del comando que resalta características como error de estado estable, tiempo de reacción, sobreimpulso, etc. En comparación con los enfoques geométricos más comunes. La Figura 1.1 muestra el error lateral (distancia ortogonal a la pista de referencia) medido como una función del tiempo para cada circuito. Se puede observar un error limitado a (1 ± 0.05) m cuando el coche circula por las curvas de la pista y un error de (0.2 ± 0.05) m en líneas rectas.

En la literatura sobre control lateral, se prefiere la parametrización natural, ya que representa intrínsecamente la forma de la curva en términos de cantidades directamente significativas para el modelo de control (por ejemplo, rumbo, radio de curvatura, etc.). Al respecto, Hammarstrand et al. [9] argumentan que los modelos basados en parametrizaciones de longitud de arco son más efectivos para representar la geometría de una carretera. Yi y col. [10] desarrollaron su control de crucero adaptativo siguiendo esta misma idea y discutieron las mejoras introducidas por un modelo clotoidal. Otras investigaciones sobre el control lateral suelen centrarse en los modelos de control adoptados, validando principalmente sus hallazgos en trayectorias predefinidas.

El controlador Stanley [11] fue el usado por la Universidad de Standford en el robot

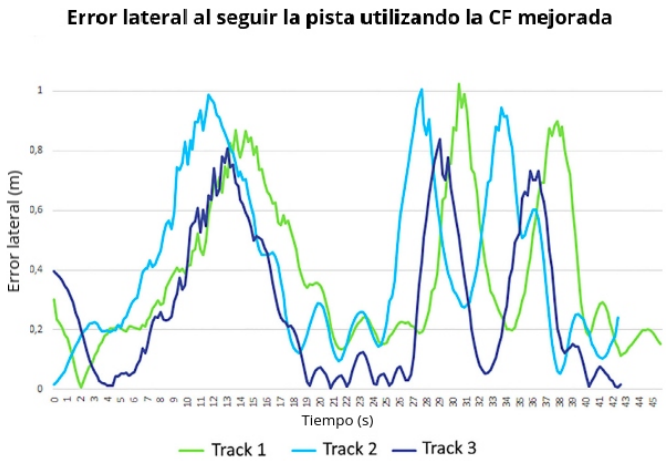


Figura 1.1: Error lateral medido durante el experimento en condiciones reales con el control lateral [8].

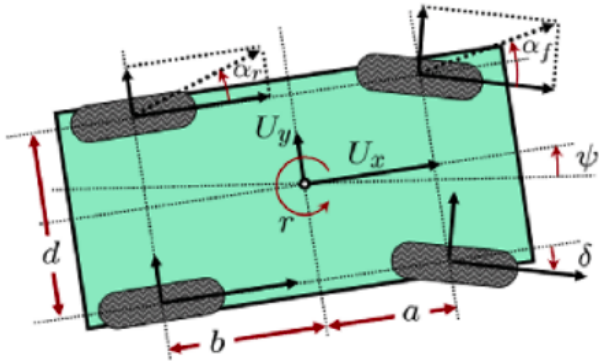


Figura 1.2: Estados del automóvil para el modelo dinámico [11].

Stanley [12] en el DARPA Grand Challenge. Se basa en dos principios, el "cross-track error" y en la alineación entre la trayectoria deseada y la del vehículo. El trabajo considera la orientación de las ruedas delanteras, en vez de la carrocería del vehículo, con respecto a la trayectoria deseada, lo que permite un control compartido del sistema, ver Figura 1.2.

Los artículos [13] y [14] propusieron leyes de control basadas en el control proporcional-integral-derivado (PID), mientras que el artículo [15] desarrolló un controlador lateral basado en el control difuso. También se han utilizado tecnologías de control, como H∞ [16], [17], regulador cuadrático lineal (LQR) [18] y control de deslizamiento [19], [20], todos estos, basados en el modelo cinemático tipo bicicleta.

El método de Lyapunov es una herramienta muy útil para diseñar un controlador de retroalimentación. Muchas de las técnicas de control por retroalimentación se basan en la

idea de una definición de la función de Lyapunov o, más específicamente, se basan en la derivada de la función de Lyapunov que garantiza la convergencia a un punto de equilibrio o un punto de estabilidad.

Norouzi et al. [21] propusieron un controlador robusto basado en Lyapunov mediante el uso de un algoritmo de optimización metaheurística para el control lateral de un vehículo autónomo. La ruta de cambio de carril doble se obtuvo utilizando una función polinomial (cuántica) de quinto grado y restricciones dinámicas. Y los errores de posición y orientación se extrajeron en función del modelo de bicicleta de vehículo de dos grados de libertad.

Tradicionalmente los estudios teóricos del control lateral de vehículos autónomos se aplican al mantenimiento de carril, sin embargo, los resultados se pueden aplicar a un área más amplia, como es el caso de cambio de carril, control de estacionamiento, control para evadir obstáculo, etc. El artículo [22] propone un esquema de cambio de carril para controlar un vehículo automatizado sin planificación de ruta y seguimiento. Utiliza un hiperplano variable en el tiempo para hacer frente al problema de la planificación de la trayectoria y el control del movimiento lateral en el control de cambio de carril.

Entre los controladores laterales basados en información obtenida por medio de la visión por computadora, se encuentra el propuesto por Huang et al. [23], usan el método de programación dinámica adaptativa y el algoritmo de control propuesto explota el conocimiento estructural de la cinemática del automóvil, así como los datos recopilados (imágenes) sobre la información del carril.

En un estudio más reciente que aborda el tema de la detección de líneas de carril, [24] presenta una técnica basada en ventanas deslizantes. Los autores demuestran que su técnica es mejor con respecto a otras propuestas que emplean la transformada de Hough o la extracción de características, que revelan problemas con la identificación de curvas. Demuestran el rendimiento de su algoritmo tanto en líneas rectas como curvas.

1.2. Planteamiento del problema

La maniobrabilidad lateral es uno de los aspectos más importantes de un vehículo autónomo. Es necesario seguir la línea central del carril para mantenerse en la carretera. Por otro lado, es una situación indeseable que el ángulo del volante varíe rápidamente o que se produzca un movimiento brusco del vehículo, lo cual hace que el pasajero se sienta incómodo e incluso la estabilidad del vehículo se vea afectada. Debido a esto, con ayuda del control lateral, el vehículo necesita que los movimientos sean suaves, precisos y seguros. Los controladores clásicos Stanley y Pure Pursuit utilizan una distancia de error como entrada, en cambio, se propone una forma alternativa de implementar el controlador lateral basado en el área de error.

1.3. Justificación

Entre los criterios de control más importantes se encuentran la rapidez y la precisión. El rendimiento del controlador varía según la selección de parámetros. Por lo tanto, es primordial seleccionar los parámetros óptimos en el controlador. Si un controlador está diseñado con un solo parámetro en lugar de varios parámetros, el programa para la sintonización de ganancia se vuelve simple y la velocidad de procesamiento se vuelve más rápida. Además, el ajuste de la ganancia del controlador se puede realizar de forma más fácil y precisa [11]. Como se puede observar en los antecedentes, existe una gran cantidad de artículos en los que se trabaja con un modelo geométrico, y específicamente utilizando los parámetros de ángulo de error y distancia de error. Es por ello que un nuevo enfoque geométrico podría proporcionar mejores resultados al tiempo que requiere una cantidad limitada de parámetros y ajustes. La contribución de este trabajo es proponer un nuevo enfoque geométrico para el control lateral, aprovechando los beneficios del enfoque geométrico tradicional y al mismo tiempo superando algunos de sus inconvenientes. Por otro lado, este trabajo pretende sentar las bases de futuras investigaciones con referencia a sistemas relacionados al control lateral de vehículos autónomos aplicando técnicas de aprendizaje profundo.

1.4. Hipótesis

Aplicando la metodología del parámetro de área de error obtenida a partir de técnicas de visión por computadora, es posible diseñar un sistema de control lateral para un robot móvil AutoNOMOS Mini V2.

1.5. Objetivos

1.5.1. General

Desarrollar un control lateral para el robot móvil AutoNOMOS Mini V2 con base en información visual sobre el posicionamiento relativo del vehículo dentro de su carril, de tal forma que el robot sea capaz de seguir una trayectoria de referencia sin conocimiento a priori del mapa, minimizando el error de posición y orientación, garantizando la permanencia dentro del carril y evitando giros bruscos del ángulo de dirección del robot.

1.5.2. Específicos

1. Definir un modelo de vehículo adecuado que cumpla con los requisitos necesarios que representen las características del robot móvil AutoNOMOS Mini V2.
2. Desarrollar un algoritmo para la detección de carril.
3. Determinar los parámetros óptimos para el diseño del control lateral de acuerdo a sus características físicas y mecánicas. Por ejemplo, configuración de tracción, dirección y grado de actuación, error de posicionamiento, ángulos de orientación, calibración de cámara, etc.
4. Diseñar y optimizar el control lateral del robot móvil, de tal forma que se minimice el área de error respecto de la referencia.

1.6. Metas

1. Desarrollar el modelo cinemático de la bicicleta en términos de área de error.
2. Diseñar los controladores laterales Pure-Pursuit y Stanley en base al modelo cinemático de la bicicleta.
3. Obtener las ecuaciones polinómicas que represente las lineas del carril.
4. Simular los controladores laterales Pure-Pursuit y Stanley en una plataforma de simulación robótica.
5. Implementación real en el robot móvil AutoNOMOS mini V2.

1.7. Metodología

Con base en las necesidades planteadas, se propone la siguiente metodología para diseñar un control lateral basado en visión computacional para un robot móvil AutoNOMOS mini V2, la cual consta de los siguientes pasos:

- **Fase inicial:** Revisar y comprender el funcionamiento de la plataforma AutoNOMOS Mini V2 y los elementos que conforman dicho robot móvil.
- **Identificación de requerimientos y necesidades:** Se consideran las limitaciones del robot móvil AutoNOMOS Mini V2 y las características de conducción deseadas, como la aceleración y ángulo de giro máximo.

- **Adquisición de imágenes:** Esta etapa comprende conocer las características de las imágenes y como realizar la captura cuando la plataforma esté en movimiento.
- **Detección de carriles:** En cada imagen de entrada se identifican los carriles presentes por medio de un detector.
- **Adquisición de datos:** El área de error es la base principal del control, en esta etapa se busca obtener este dato a partir de la detección de carriles.
- **Modelo matemático:** Definir las variables basadas en el modelo del vehículo cinemático, así como también desarrollar las ecuaciones que describan el comportamiento del sistema.
- **Estrategia de diseño de control:** Verificar técnicas de control y evaluar la técnica que más se adapte y brinde mejores resultados.
- **Simulación virtual:** Se implementan simulaciones en Gazebo-ROS para verificar el comportamiento del modelo y solucionar fallas cometidas.
- **Implementación real:** Se programan las ecuaciones en el simulador ROS, para posteriormente implementar el diseño en el robot móvil AutoNOMOS mini V2.

1.8. Delimitaciones

- La investigación y el planteamiento del problema supone que el control automático de un automóvil está dividido en control longitudinal y control lateral.
- La pista de pruebas con la que se pretende trabajar está en interiores, sin cambios de iluminación y se considerará como un entorno estático. Dicha pista tiene un fondo negro, líneas blancas y dos carriles.
- Se utilizan imágenes a color RGB obtenidas de la cámara Intel Real Sense 300.
- El robot móvil AutoNOMOS Mini V2 utiliza un sistema de dirección Ackerman actuado por un servomotor que permite obtener una dirección en el rango [42°, 138°].

1.9. Estructura de la tesis

Este trabajo de tesis está organizado por 5 capítulos, a continuación se exponen las ideas principales de cada uno: Capítulo 1 se presenta la introducción y los antecedentes de esta investigación, se describe el planteamiento del problema y su justificación, se especifican

los objetivos a cumplir para demostrar la hipótesis planteada y se describe la metodología seguida en el desarrollo. En el Capítulo 2 se describen los conceptos y bases teóricas necesarios para este trabajo de tesis. En el Capítulo 3 se describe el desarrollo de la metodología por etapas llevada a cabo para el diseño del control lateral. En el Capítulo 4 se muestran los resultados obtenidos del control lateral utilizado en dos pistas virtuales y una pista real. Por ultimo, en el capítulo 5 se dan a conocer las conclusiones generales obtenidas del trabajo desarrollado y se mencionan posibles trabajos futuros para mejorar el sistema propuesto.

Capítulo 2

Marco Teórico

El interés en los sistemas de asistencia al conductor basados en la visión computacional ha crecido enormemente durante la última década, y el asistente de mantenimiento de carril juega un papel importante en estos sistemas. El reconocimiento del entorno usando imágenes es parte importante de ADAS, por lo que las cámaras están presentes en estos sistemas. El desarrollo de técnicas para la detección de carril es parte importante del desarrollo de vehículos autónomos. Los prototipos a escala equipados con sensores como es el caso del AutoNOMOS Mini V2 permiten la investigación y desarrollo de estas técnicas. Tomando en cuenta los recursos ya existentes para realizar esta propuesta, a continuación se describen los conceptos necesarios para su realización.

2.1. Visión Computacional y procesamiento de imágenes

Para expertos en el área [25], el objetivo de la visión computacional es obtener información del entorno utilizando el procesamiento de imágenes para identificar objetos de interés y su posición en un ambiente. La visión computacional está estrechamente relacionada con el procesamiento de imágenes, que a pesar de compartir información en común, sus objetivos son diferentes. El procesamiento de imágenes se encarga del mejoramiento de la calidad de las imágenes para su posterior interpretación.

Por el contrario, la visión computacional se encarga de extraer características de una imagen para su descripción e interpretación. Actualmente existen múltiples aplicaciones practicas de la visión computacional, entre éstas podemos mencionar las siguientes:

- Robótica móvil y vehículos autónomos.

- Interpretación de imágenes aéreas y de satélite.
- Análisis e interpretación de imágenes médicas.
- Análisis de imágenes para astronomía.

2.2. Transformación de perspectiva

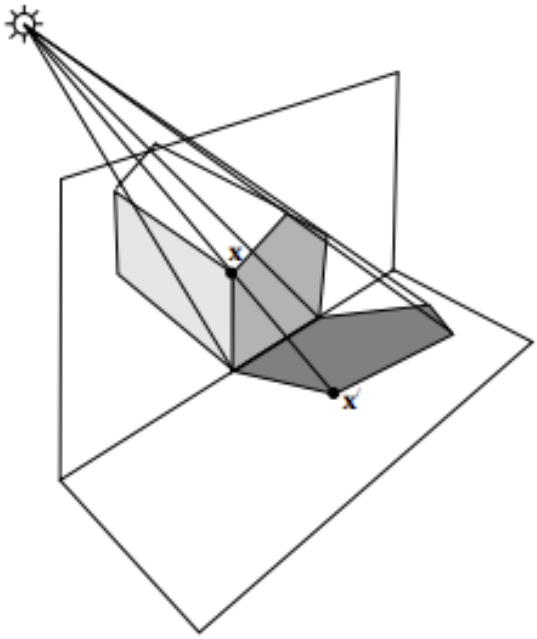


Figura 2.1: Ejemplo de una transformación proyectiva, $x = Hx$, que surge en las imágenes en perspectiva [26].

La transformación de perspectiva está asociada con el cambio en el punto de vista. Podemos cambiar la perspectiva de una imagen o vídeo dado, para obtener una mejor comprensión de la información requerida. Este tipo de transformación mantiene invariantes las propiedades de longitud (distancia entre dos puntos), ángulo (entre dos líneas) y área.

La técnica de mapeo de perspectiva inversa es el mapeo invertible de puntos y líneas en el plano proyectivo P^2, es decir, una transformación no singular lineal de coordenadas homogéneas [26]; y se define como:

"Un mapeo de $P^2 \rightarrow P^2$ es una proyección sí y sólo sí existe una matriz H_{3x3} no singular tal que para cualquier punto en P^2 representado por el vector x tenga un mapeo equivalente a Hx".

Por tanto, una transformación proyectiva puede escribirse como:

$$x' = Hx = \begin{bmatrix} A & t \\ V^T & v \end{bmatrix} \tag{2.1}$$

en donde A es una matriz $2x2$ no singular que representa la concatenación de matrices de rotación y escalamiento, t es un vector de traslación y $V^t = (v_1, v_2)$ el vector responsable de los efectos no lineales en la proyección. La Figura 2.1 muestra un ejemplo de una transformación proyectiva.

2.3. Interpolación polinomial

El problema matemático de la interpolación es el siguiente: Dado un conjunto de n pares de valores (xk, yk), encontrar una función $f(x)$ que cumpla $f(xk) = yk, k = 1, \cdots, n$. Existen diversos métodos para encontrar dicha función. Los más conocidos son los métodos que interpolan $f(x)$ por medio de un polinomio o una función racional (cociente de dos polinomios). En el caso de la interpolación polinómica dados n puntos, el grado más bajo del polinomio que pasa por los n puntos es $n-1$, a menos que dos o más puntos pertenezcan a un polinomio de grado más bajo. En este caso se trata de encontrar una combinación lineal de dos funciones $f1(x) = 1$ y $f2(x) = x$ que aproxima de la mejor manera posible una colección de datos determinada.

Definición 2.3 Dada una colección de datos, y dadas unas funciones $f1, \cdots, fm$, la solución de mínimos cuadrados para aproximar los datos con las funciones dadas es una función de la forma

$$f(x) = a_1 f(x) + \cdots + a_m f_m(x) \tag{2.2}$$

que minimiza el error cuadrático

$$\sum_{k=0}^{n} (f(x_k) - y_k)^2 \tag{2.3}$$

La elección de minimizar el error cuadrático (la elección de la potencia 2 en lugar de otra) obedece fundamentalmente a la facilidad de la resolución del problema en ese caso. Comenzamos por observar que, de existir una función de la forma 2.3 que hiciese cero el error tendríamos

$$a_1 f(x) + \cdots + a_m f_m(x) = y_k, k = 1, \cdots, n. \tag{2.4}$$

2.4. AutoNOMOS Mini V2

"AutoNOMOS Mini v2" es un modelo de vehículo a escala desarrollado para fines educativos por la Freie Universität Berlin. Puede ser controlado usando un teléfono móvil o puede ser programado para conducción en modo completamente autónomo. La computadora principal es una placa Odroid (XU4 64GB) que ejecuta Ubuntu Linux y el Sistema Operativo Robótico (ROS) [27]. La plataforma cuenta con distintos sensores:

- Escáner láser giratorio (RPLIDAR A2 360) que proporciona detección de obstáculos alrededor del vehículo.
- Un sistema estereoscópico tipo Kinect (Intel RealSense SR300) que se puede utilizar para detectar obstáculos y líneas de carril.
- Cámara de video ojo de Pez (ELP 1080) que está apuntando al techo y se puede usar para identificar marcadores, proporcionando una especie de GPS.
- Una Unidad de Medición inercial (MPU6050), la cual otorga mediciones de acelerómetros y giroscopios de la plataforma. Estas mediciones pueden ser usadas para complementar la odometría y la medición de la rotación del vehículo.

El AutoNOMOS Mini V2, mostrado en la Figura 2.2, tiene una geometría tipo Ackerman con tracción en las cuatro ruedas, cuyo tamaño es una décima parte de su contra parte real. Mide 40 cm de largo y 20 cm de ancho. Utiliza un motor sin escobillas de 24 $Watts$ para mover las ruedas y una batería Li-Po de 16.8 V de 4 celdas. El ángulo de dirección es controlado por un servomotor de 5 V.

Figura 2.2: AutoNOMOS mini V2 [25].

2.5. Intel Real Sense 300

La cámara encargada de aportar una imagen de la parte delantera del vehículo es una cámara Intel Realsense SR300 como la mostrada en la figura 2.3. Se trata de un modelo de cámara frontal, con profundidad 3D de corto alcance y un módulo 2D, combinando la detección de profundidad con una cámara RGB de 1080p.

Esta cámara es perfecta para desarrollar software y crear aplicaciones con detección de movimientos o reconocimiento facial. Algunos otros ejemplos de aplicaciones de esta cámara son el análisis de rostros y detección, escaneado y mapeado, segmentación de escenas, seguimiento de la mano, detección de gesto y realidad aumentada.

Figura 2.3: Cámara Intel Realsense RS300.

2.6. OpenCV

OpenCV es una biblioteca libre de visión computacional creada por Intel. Apareció por primera vez en 1999 y desde entonces ha sido utilizada en numerosas aplicaciones.

Al igual que ROS (Robot Operating System), tiene una licencia BSD (Berkeley Software Distribution) que permite su uso por motivos comerciales y científicos sin restricciones.

Entre sus aplicaciones más destacadas se encuentran el procesamiento de imágenes 2D y 3D, los sistemas de reconocimiento facial y de gestos, la identificación y el modelado de objetos y obstáculos, la segmentación y el reconocimiento, el seguimiento del movimiento y la realidad aumentada, entre otras. También permite la integración de sistemas de percepción de profundidad mediante el uso de dos cámaras. Es compatible con las plataformas Windows, Linux, MacOS, iOS y Android. Además, es compatible con C++, Java y Python.

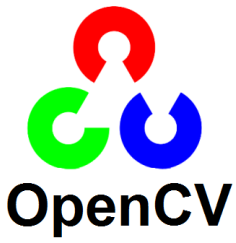


Figura 2.4: Logo de OpenCV.

2.7. Robot Operating System (ROS)

Robot Operating System (ROS) [28], es un entorno de trabajo para el desarrollo de software utilizado en robótica que provee la funcionalidad de un sistema operativo en un clúster heterogéneo. Inició su desarrollo en el año 2007 en los laboratorios de inteligencia artificial de la Universidad de Standford. De este modo, ROS se extiende a disciplinas de investigación, desarrollo de robots industriales, propósitos educativos y con fines de ocio. ROS proporciona bibliotecas y herramientas que, mediante una abstracción del hardware, permite el acceso a los componentes del mismo y el diseño de complejas aplicaciones software teniendo una idea superficial de cómo funciona el hardware.

El método principal para crear una red de comunicación entre los distintos componentes es proporcionar servicios que puedan ser consultados o definir conexiones publisher/subscriber entre los nodos. Las comunicaciones se realizan a través de mensajes de un tipo específico proporcionado por los paquetes generales o por paquetes individuales. Ubuntu es el sistema operativo que cuenta con soporte oficial. En nuestro caso, al trabajar con la distribución ROS Kinetic, la elección fue Ubuntu 16.04. Por otro lado, ROS permite trabajar en Python o C++ como lenguajes de programación, habiéndose seleccionado Python por sus cualidades y rendimiento.

Figura 2.5: Logo de ROS.

2.8. Simulador AutoNOMOS mini en Gazebo

Una de las característica importantes de ROS es que permite la simulación de los robots tomando en cuenta los parámetros físicos del robot y sus restricciones. Gazebo es un simulador 3D multi-robot con dinámica. Ofrece la posibilidad de simular con precisión y eficiencia, diversidad de robots, objetos y sensores en ambientes complejos interiores y exteriores. Gazebo genera, tanto la realimentación realista de sensores, como las interacciones entre los objetos físicamente plausibles, incluida una simulación precisa de la física de cuerpo rígido.

Para el desarrollo de este trabajo de tesis, se utilizó el modelo diseñado por el ITAM (Instituto Tecnológico Autónomo de México) [29], es un repositorio independiente para la simulación AutoNOMOS-Gazebo, el cual está disponible en línea. El robot que incluye este modelo cuenta con una cámara y un sensor Lidar, así como con movimiento y medición de orientación y odometría, ver Figura 2.6.

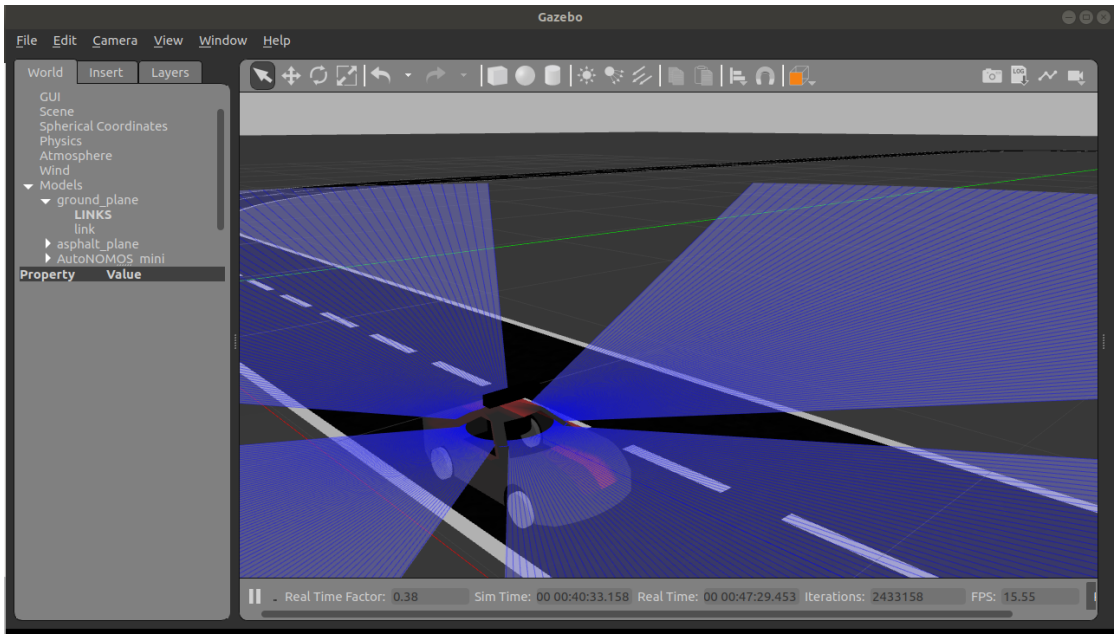


Figura 2.6: Modelo AutoNOMOS mini desarrollado por ITAM [29].

2.9. Modelo de bicicleta cinemática

El modelo de bicicleta es un modelo orientado al control adecuado de un vehículo de cuatro ruedas, donde las ruedas delanteras izquierda y derecha se combinan en una sola rueda direccional, y las ruedas traseras izquierda y derecha se combinan juntas en una sola rueda motriz. En este caso se tienen dos lados que definen al modelo de bicicleta, siendo la rueda direccional el lado variable, y la rueda motriz la parte fija. Para esta discusión, usaremos un segmento de línea como la ruta de referencia, que se muestra como una línea azul sólida en la Figura 2.7 del modelo de bicicleta cinemática. También es visible una línea roja punteada que es paralela al camino pero que pasa por el centro del eje delantero. A los efectos del control lateral, redefinimos nuestro rumbo en relación con el segmento de línea de ruta actual.

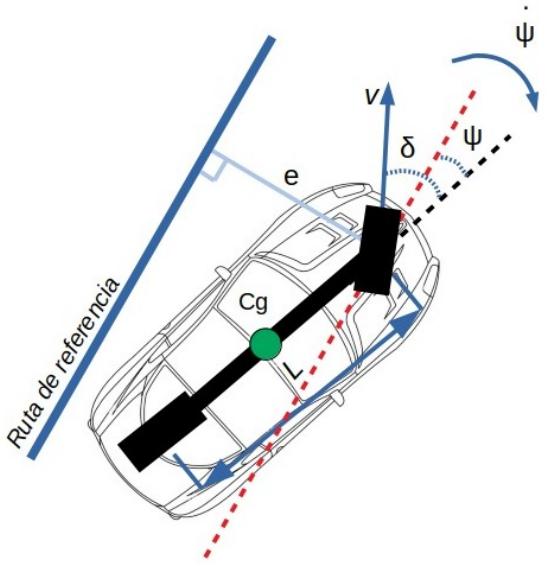


Figura 2.7: Modelo cinemático tipo bicicleta.

El lado variable se reutilizará para representar el ángulo de rumbo relativo del vehículo con respecto a la línea de trayectoria. La velocidad de la rueda delantera (v) y el ángulo de dirección con respecto a la dirección (δ) del rumbo no cambian. Se puede colocar un marco de referencia para el vehículo en el centro del eje trasero, en el centro del eje delantero o en el centro de gravedad según el diseño del controlador.

$$\dot{\psi}(t) = \frac{-v(t)\sin\left(\delta(t)\right)}{L} \tag{2.5}$$

Donde se tiene que:

- $v(t)$: Velocidad de la rueda delantera
- $\delta(t)$: Ángulo de dirección de rumbo
- L: Distancia entre ejes

El error de rumbo, es igual a la diferencia entre el rumbo de la ruta y el rumbo del vehículo en el punto de referencia a lo largo de la ruta. Es una medida principal de qué tan bien el vehículo está alineado y se mueve en la dirección de la ruta deseada. La tasa de error nos ayuda a comprender cómo evoluciona el error de rumbo ($\dot{\psi}$) con el tiempo y se puede calcular a partir de las ecuaciones del modelo de bicicleta cinemática. La tasa de error de rumbo también se puede expresar en términos de cualquiera de los tres puntos de referencia del vehículo.

Para los segmentos de línea recta, la tasa de cambio de rumbo deseada es cero y se puede eliminar. Esto se debe a que el rumbo de referencia no varía en el tiempo para una línea recta y, de hecho, es igual a cero, ya que hemos redefinido nuestro rumbo en relación con la dirección de la ruta actual.

$$\dot{e}(t) = v(t) \sin\left(\psi(t) - \delta(t)\right) \tag{2.6}$$

Donde se tiene que:

- $\dot{e}(t)$: Error de compensación
- $v(t)$: Velocidad de la rueda delantera
- $\psi(t)$: Ángulo de dirección de rumbo
- $\delta(t)$: Ángulo de dirección del vehículo

El otro tipo de error es un error de compensación llamado *cross-track error*. Este error, se define como la distancia entre el punto de referencia en el vehículo y el punto más cercano en la ruta deseada. Es la medida principal de qué tan cerca está la posición del vehículo de la posición deseada a lo largo del camino. Tanto el error de rumbo como el *cross-track error* deben converger a cero para que el vehículo siga correctamente la ruta deseada. La línea desde el punto de referencia del vehículo hasta el punto de referencia de la ruta es perpendicular a la ruta. La tasa de cambio del *cross-track error* se puede calcular extrayendo el componente lateral de la velocidad de avance. A partir de esta ecuación, podemos ver que a medida que aumenta la velocidad, el *cross-track error* cambia más rápidamente, lo que significa que se necesitan ángulos de dirección más pequeños para corregir errores de del mismo tamaño.

El seguimiento de trayectoria geométrica es cualquier controlador que rastrea una ruta de referencia usando solo la geometría de la cinemática del vehículo y la ruta de referencia. En el caso de los automóviles con geometría Ackermann, un controlador de seguimiento de trayectoria geométrica es un tipo de controlador lateral que ignora las fuerzas dinámicas en los vehículos y asume que la condición de no deslizamiento se mantiene en las ruedas.

2.10. Control Lateral

Una de las categorías de controladores son los controladores geométricos, los cuales se basan en la geometría, las coordenadas de la ruta deseada y los modelos cinemáticos del vehículo.

2.10.1. Control Lateral Stanley

El controlador Stanley es un controlador de seguimiento de ruta geométrica que es simple pero útil para robótica autónoma y automóviles autónomos. En la Figura 2.8 se puede ver las ligeras modificaciones a los términos relevantes basados en los supuestos de Stanley. El error transversal de la vía se mide en relación con el eje delantero, y el punto de referencia en la ruta no tiene una distancia de anticipación asociada. Primero, para eliminar el error de rumbo relativo a la trayectoria, el ángulo de dirección se establece directamente igual al rumbo.

$$\delta(t) = \psi(t) \tag{2.7}$$

Luego, para eliminar el error transversal, se agrega un control proporcional, cuya ganancia se escala por la inversa de la velocidad de avance:

$$\delta(t) = \arctan\left(\frac{ke(t)}{v_f(t)}\right),\ \delta(t) \in [\delta_{min}, \delta_{max}] \tag{2.8}$$

Finalmente, el comando del ángulo de dirección se mantiene dentro de los ángulos de dirección mínimo y máximo, que generalmente son simétricos alrededor de 0.

El controlador Stanley escala sus ganancias por la velocidad de avance y también tiene la tangente inversa de la señal de control proporcional. La ley de control final simplemente combina estos tres elementos para establecer el ángulo de dirección del automóvil de la siguiente manera.

$$\delta(t) = \psi(t) + \arctan\left(\frac{ke(t)}{v_f(t)}\right), \delta(t) \in [\delta_{min}, \delta_{max}] \tag{2.9}$$

Donde se tiene que:

- $\delta(t)$: Ángulo de dirección del vehículo
- $\psi(t)$: Ángulo de dirección de rumbo
- k: Ganancia
- $e(t)$: Error de rumbo relativo a la trayectoria
- $v_f(t)$: Velocidad de avance

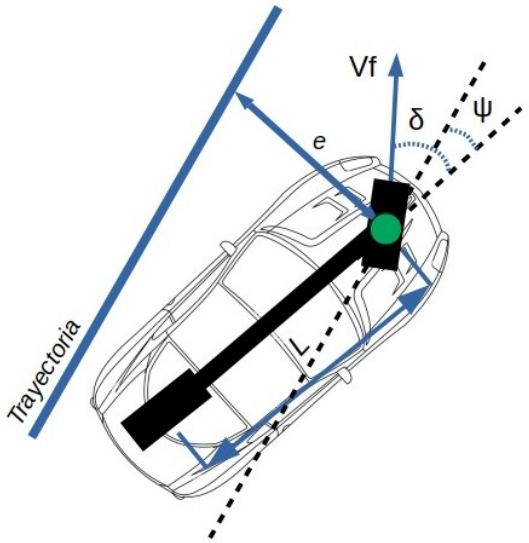


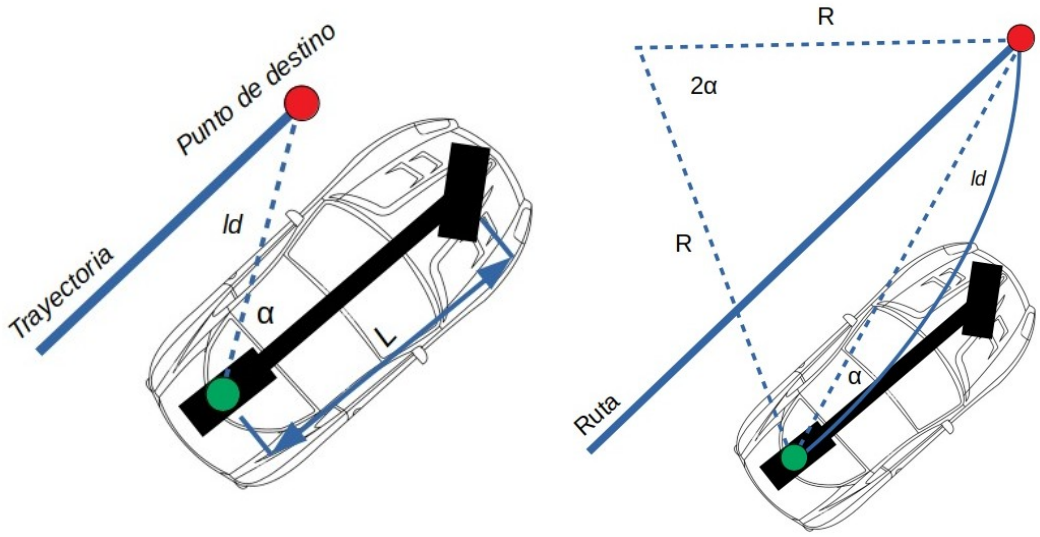
Figura 2.8: Esquema del modelo cinemático Stanley.

2.10.2. Control Lateral Pure-Pursuit

En el método de control Pure Pursuit, la idea central es colocar un punto de referencia en la ruta a una distancia fija por delante del vehículo, y así calcular los comandos de dirección necesarios para cruzarse con este punto utilizando un ángulo de dirección constante. A medida que el vehículo gira hacia el camino para seguir esta curva, el punto continúa

avanzando, reduciendo el ángulo de dirección y llevando suavemente el vehículo hacia el camino.

En este método, el centro del eje trasero se utiliza como punto de referencia en el vehículo, y definimos la línea que conecta el centro del eje trasero con el punto de referencia objetivo como una línea de distancia fija l_d, conocida como *look-ahead distance*, que es la línea discontinua en la Figura 2.9a. El ángulo entre el rumbo de la carrocería del vehículo y la línea de avance se denomina alfa (α). El punto objetivo de la trayectoria, el centro del eje trasero y el centro instantáneo de rotación forman un triángulo con dos lados de longitud R y uno de longitud l_d. Se define el arco que lleva el punto de referencia del vehículo al punto de destino en la ruta y es la parte del círculo ICR que cubre el ángulo de 2α. El ángulo 2α se puede derivar utilizando identidades trigonométricas estándar, ver Figura 2.9b.



(a) Esquema del modelo cinemático Pure-Pursuit.

(b) Arco de referencia esquema Pure-Pursuit

Figura 2.9: Esquema del modelo cinemático Pure-Pursuit.

Con base en la ley de los senos e identidades trigonométricas, se escribe la siguiente ecuación:

$$\frac{l_d}{2\sin(\alpha)\cos(\alpha)} = \frac{R}{\cos(\alpha)} \tag{2.10}$$

Reorganizando:

$$R = \frac{l_d}{2\sin(\alpha)} \tag{2.11}$$

Por tanto, podemos obtener la curvatura:

$$k = \frac{1}{R} = \frac{2\sin(\alpha)}{l_d} \tag{2.12}$$

El *cross-track error* (e) se define aquí como la distancia lateral entre el vector de rumbo y el punto de destino, por lo que:

$$\sin(\alpha) = \frac{e}{l_d} \tag{2.13}$$

Y de ahí se puede calcular el valor de la dirección deseada:

$$\delta = \arctan\left(\frac{2L\sin(\alpha)}{l_d}\right), \quad \delta \in [\delta_{min}, \delta_{max}] \tag{2.14}$$

Donde se tiene que:

- δ: Ángulo de dirección
- L: Longitud entre ejes
- α: Ángulo entre rumbo y línea de avance
- l_d: Distancia de anticipación

Es importante tener en cuenta que el controlador Pure-Pursuit con un valor fijo de l_d conduce a un controlador de curvatura que no tiene en cuenta la velocidad del vehículo. Esto significa que el ángulo de dirección seleccionado sería el mismo independientemente de si el vehículo circula a 10 km/h o 100 km/h, lo que da lugar a aceleraciones laterales muy diferentes. Un controlador ajustado para alta velocidad sería demasiado lento a baja velocidad y uno ajustado para baja velocidad sería peligrosamente agresivo a altas velocidades.

Capítulo 3

Desarrollo

En este capítulo se presenta el proceso realizado para desarrollar el sistema de control lateral. El sistema consta de dos pasos clave: el uso del algoritmo de ventanas deslizantes para obtener el modelo que representa las líneas del carril y la obtención del área de error de posición con el objetivo de minimizar la distancia del centro del robot y el centro del los carriles. Finalmente, se describe la implementación tanto en simulación como en el robot móvil AutoNOMOS Mini V2.

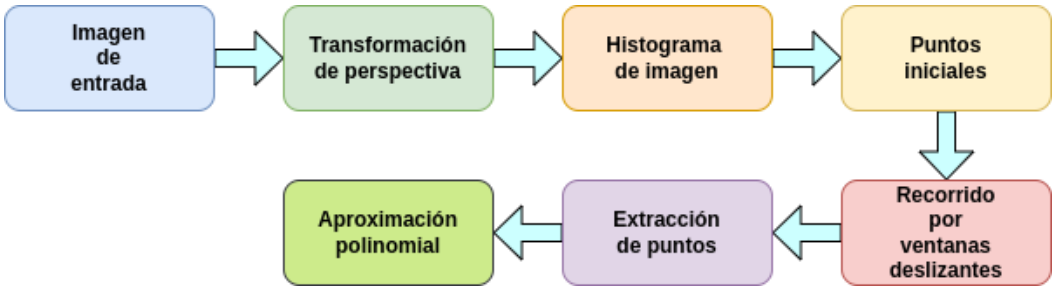


Figura 3.1: Etapas de la detección de carriles utilizando ventanas.

3.1. Detección de líneas de carril

Esta sección describe el desarrollo de un algoritmo de segmentación de imágenes para identificar líneas de carril en una imagen. Este enfoque se basa en la búsqueda de una concentración de píxeles blancos a lo largo del eje x de la imagen y en un procedimiento de ventana deslizante derivado del concepto de Haque et al [30]. En general, el diagrama de la Figura 3.1 representa la simplicidad gradual de la evolución de este procedimiento.

Para este estudio, el algoritmo se modificó para tener en cuenta las imágenes tomadas por la cámara Intel Real Sense.

La canalización toma una imagen como entrada y genera un modelo matemático de los límites de los carriles como salida. La imagen es capturada por una Intel Real Sense, una cámara fijada en la parte superior del robot. El modelo de límite de carril es un polinomio de segundo grado:

$$y(x) = c_0 + c_1 x + c_2 x^2 \tag{3.1}$$

Tanto x como y se miden en metros, definen un sistema de coordenadas en la carretera como se muestra en la Figura 3.2.

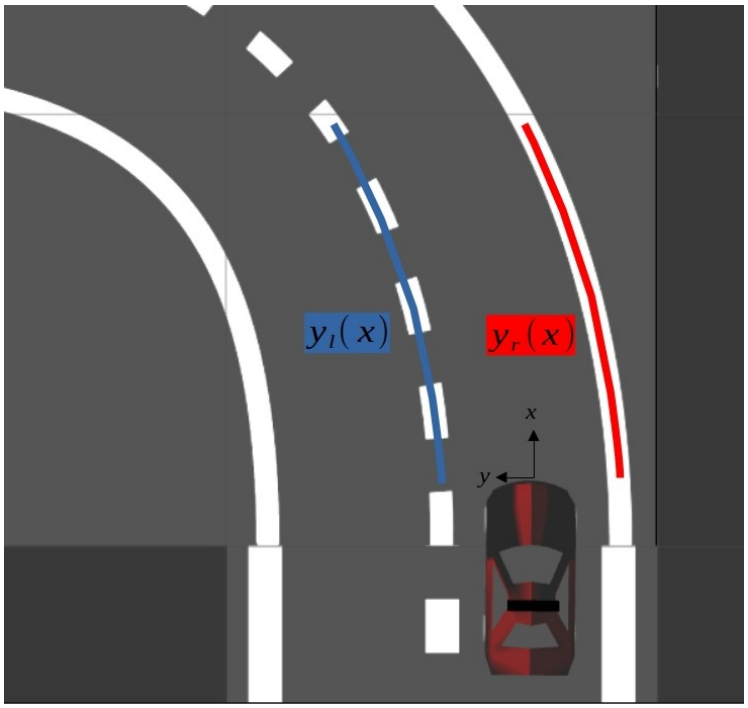
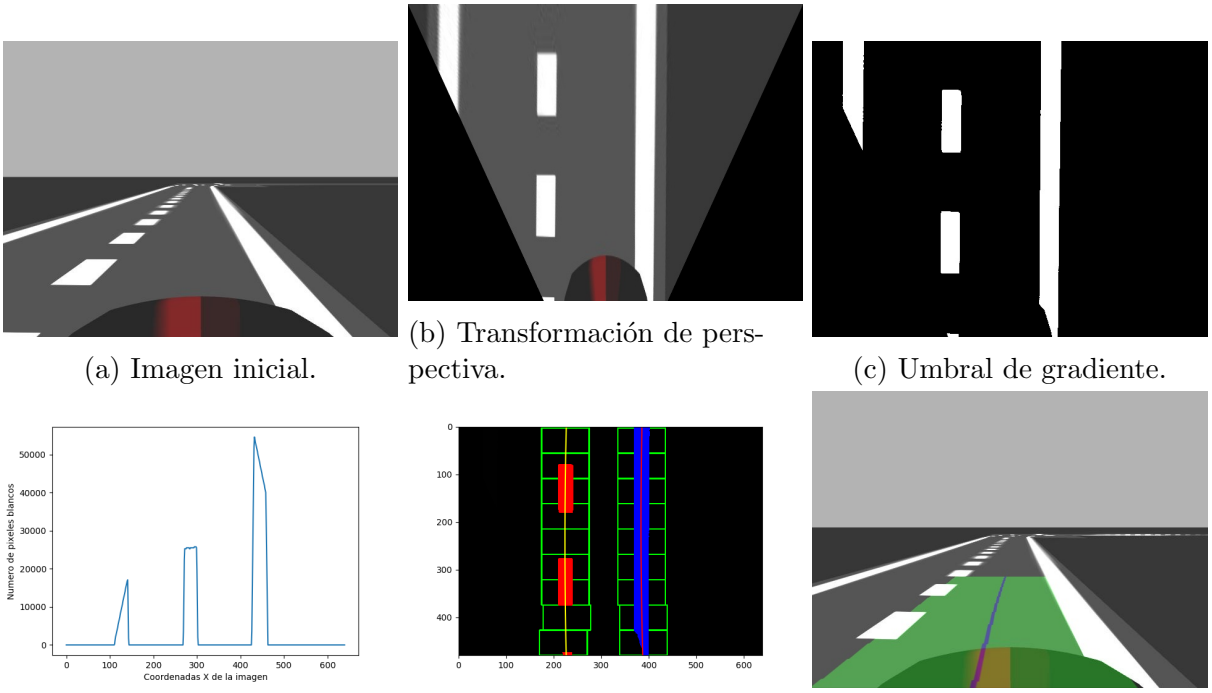


Figura 3.2: Sistema de coordenadas viales. La perspectiva se conoce como vista de pájaro.

3.1.1. Imagen de entrada

El robot móvil AutoNOMOS Mini V2 cuenta con la cámara RGBD Intel RealSense SR300 montada sobre el robot móvil a 16.5 *cm* sobre el suelo; con su eje óptico paralelo al suelo y mirando hacia el frente. La principal fuente de información para el seguimiento de

carriles se basa en la información visual extraída de las imágenes que proporciona la cámara frontal del vehículo, siendo relativa a la forma de la carretera. Las imágenes capturadas por este sensor son como la mostrada en la Figura 3.8a con dimensiones de 640 x 480 píxeles.



(a) Imagen inicial.

(b) Transformación de perspectiva.

(c) Umbral de gradiente.

(d) Histograma de píxeles blancos.

(e) Búsqueda de píxeles por ventana deslizante.

(f) Imagen final, ilustra el área alusivo al carril.

Figura 3.3: Procesamiento detección de carriles.

3.1.2. Transformación de perspectiva

La corrección de perspectiva [31] se usa para transformar una imagen de entrada en una imagen de vista superior 2D. Para centrarse en la parte de las líneas del carril en la imagen, movemos nuestro punto de vista a la mejor perspectiva de la calle. Esto se hace para estirar la imagen y facilitar localizar con precisión las líneas del carril y obtener el ajuste polinómico referente a cada línea.

Con el proceso de transformación de perspectiva se obtiene solo la sección de interés, es la porción de la imagen que contiene la información más relevante en cuanto a las líneas de los carriles. La distorsión de la perspectiva de la vista de la cámara frontal hace que los carriles paralelos y rectos converjan en un punto de fuga. Esta transformación facilita el procesamiento de los algoritmos de detección de carriles, ya que las líneas rectas paralelas se representan como líneas verticales. La ventaja de crear una vista de pájaro es que la relación

de la imagen es consistente y el ancho de la línea de carril es básicamente el mismo, lo que puede simplificar el procesamiento de cálculo del reconocimiento de línea de carril. Por supuesto, el proceso de crear la vista de pájaro en sí sacrificará cierta eficiencia.

La transformación de perspectiva requiere matrices de puntos de origen y destino. Estos puntos se pueden determinar inspeccionando manualmente una imagen con líneas paralelas conocidas. Una vez que se determinan los puntos de origen y destino, la transformación se puede aplicar a las imágenes de entrada. La Figura 3.8b es un ejemplo de la imagen de salida deformada.

El principio de reconocimiento de líneas de carril se basa principalmente en la diferencia de color entre la línea de carril y el carril. Una vez obtenida la imagen con información relevante para la detección de carriles, el siguiente paso es resaltar los píxeles de interés por medio de una operación de umbralización. Dado que es de interés eliminar el fondo de la imagen para conservar solo los píxeles que corresponden al nivel de gris de las líneas de los carriles, se realiza la umbralización de la imagen. Este umbral es el punto (o valor) en el cual el histograma de una imagen se divide en dos picos, el resultado después de aplicar el umbral de gradiente se ilustra en la Figura 3.3c.

3.1.3. Histograma de imagen

En esta sección, lo que se busca es localizar con precisión el inicio de la línea del carril. De la imagen con vista de pájaro, se obtiene su respectivo histograma en la dirección X y se ubica el pico máximo como el punto de inicio de búsqueda para las líneas del carril, ver Figura 3.3d.

La Figura 3.3c muestra la vista de tres carriles en vista superior, la Figura 3.3d su respectivo histograma. Sin embargo, lo que se busca es obtener solamente los dos carriles que se encuentran enfrente del robot, el algoritmo realiza una comparación y devuelve 2 coordenadas X en la imagen que tengan el mayor número de píxeles blancos.

Algoritmo 1 Histograma de imagen

Entrada: Imagen con corrección de perspectiva
Salida: Puntos de coordenadas X iniciales
1 Conteo de píxeles blancos en el eje horizontal (Histograma).
2 Calculo de máximos del histograma.
3 Filtrar máximos del histograma.
4 Puntos de coordenadas X iniciales.

3.1.4. Búsqueda por ventana deslizante

Una ventana deslizante es una región rectangular de ancho y alto fijos que se "desliza" a través de una imagen (ver Figura 3.4). Un argumento importante es el tamaño de paso, este tamaño indica cuántos píxeles se va a "saltar" en ambas direcciones (x, y). Normalmente, no es recomendable recorrer todos y cada uno de los píxeles de la imagen (es decir, tamaño de paso = 1), ya que esto sería computacionalmente costoso. Cuanto más pequeño sea el tamaño de paso, más ventanas necesitará examinar, en cambio, cuanto mayor sea el tamaño de paso, menos ventanas necesitará examinar; sin embargo, si bien esto será computacionalmente más eficiente, es posible que no detecte objetos en las imágenes si el tamaño de paso se vuelve demasiado grande.

El primer paso para detectar con precisión la marca de carril es identificar la región de máxima probabilidad de su existencia. La región de máxima probabilidad se puede determinar observando el histograma de la vista de pájaro, que produce dos picos distintos, uno para el carril izquierdo y otro para el derecho. Ahora, los carriles se pueden identificar utilizando el enfoque de búsqueda de ventana deslizante. El algoritmo de la ventana deslizante utiliza este conocimiento para identificar y seguir las líneas de carril desde la parte inferior hasta la superior de la imagen. La Figura 3.3e muestra el resultado de la búsqueda por ventana deslizante. Se visualizan los resultados de la ventana deslizante dibujando un rectángulo en la parte central donde se a detectado un píxel blanco.

Proceso de búsqueda: En primer lugar, se determina el tamaño de la ventana de búsqueda (anchura y altura). A continuación, se utiliza el punto inicial obtenido por el histograma como punto base de la búsqueda actual. Si bien la primer ventana de inicio se toma a partir del punto inicial de cada línea, cada ventana siguiente se define con base en la ubicación del píxel con la menor coordenada en Y de la ventana anterior. La anchura y la altura se determinan dividiendo el tamaño de la imagen por el número de ventanas de búsqueda. En este caso, existen nueve ventanas. En segundo lugar, se calculan las estadísticas del histograma horizontal y vertical de cada ventana de búsqueda de forma independiente. El algoritmo 2 cuenta el número de píxeles con nivel de gris distinto de cero en la región del cuadro de búsqueda y elimina los cuadros con menos de 50 píxeles distintos de cero. A continuación, calcula el valor medio de las coordenadas de los píxeles distintos de cero como centro del fotograma de búsqueda actual. Los parámetros de la curva de carril correspondientes a la búsqueda actual se obtienen ajustando estos puntos centrales con un polinomio de segundo orden.

Usando el método de ventanas deslizantes, se detectan los píxeles en una imagen que son los límites de carril. Se asocian los píxeles de límite de carril a puntos en la carretera, $(x_i, y_i),\ i = 0, 1, 2, ..., n$. Posteriormente, se puede crear una lista de tuplas $(x_i, y_i, p_i(left))$, ya que se conocen las coordenadas de la carretera de cada píxel (x_i, y_i).

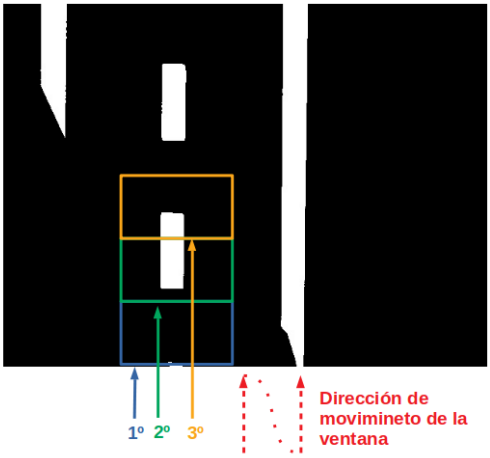


Figura 3.4: Técnica de ventana deslizante.

En los casos donde se detectan líneas que son pequeñas y no representan una línea de carril completa, se puede filtrar esta lista y descartar todas las tuplas donde $p_i(left)$ es pequeño. La lista filtrada de (x_i, y_i) se puede introducir en un método para el ajuste de polinomios, lo que dará como resultado un polinomio $y(x) = c_0 + c_1x + c_2x^2$ describiendo el límite del carril izquierdo. El mismo procedimiento se repite para el límite derecho.

3.1.5. Ilustrar Carril

Después de conocer los polinomios característicos de cada línea de carril, se realiza la operación de deformación para garantizar una visualización precisa de la imagen. Utilizando los mismos parámetros que al cambiar a la vista superior, se utiliza la función de transformación de perspectiva inversa para volver a la imagen original. La Figura 3.3f demuestra los resultados en la imagen final. La línea azul denota el centro del carril, mientras que la región verde delimita el área completa del carril.

3.2. Área de error

La cámara montada en la parte superior del robot se utiliza para estimar la posición lateral del vehículo con respecto a su carril y para estimar la curvatura de la carretera. Debido a la altura a la que se encuentra la cámara, el campo de visión de la misma solo captura el camino que se encuentra a una distancia menor o igual a L_h, tal y como se ilustra en la Figura 3.5 marcada en color verde.

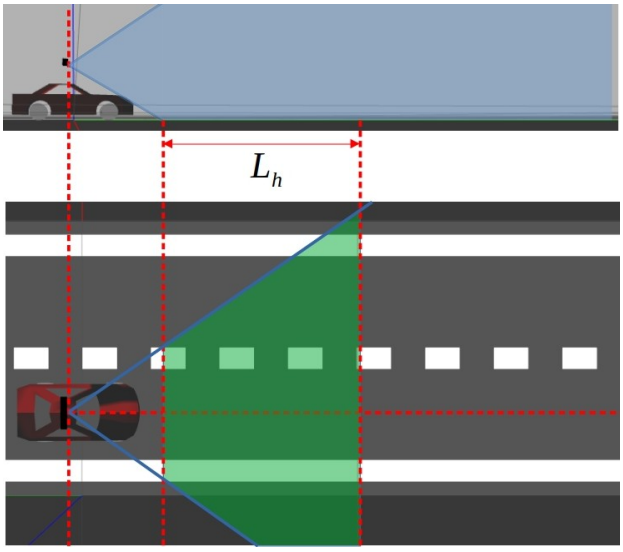


Figura 3.5: Campo de visión de la cámara a bordo del robot móvil.

Una vez identificado el polinomio característico que representa cada línea de carril, se determinan la línea central del carril y la línea de referencia que se origina del centro del robot móvil. La técnica para determinar la línea que representa el centro del robot móvil tiene en cuenta las coordenadas de la imagen y asume que la cámara está situada en el centro del robot, lo que hace que la línea central del robot móvil inicie siempre en el centro de la imagen de entrada. Esta línea central del robot, toma las mismas características (curvatura) que la línea central del carril, pero con la diferencia de que su punto de origen se encuentra en la parte media de la imagen, como se muestra en la Figura 3.6.

(a) Carril recto. (b) Carril en curva.

Figura 3.6: Ilustración de la línea central del robot (rojo) y la línea central de los carriles (azul).

Las líneas negras de la Figura 3.7 indican las líneas del carril, la línea azul representa

la línea central del carril (el centro de los dos carriles), la línea roja discontinua representa la ubicación de referencia y el campo de visión de la cámara (L_h) estará definido por el tamaño de la imagen.

Los valores positivos en el área de error indican que el vehículo se desplaza hacia la derecha, mientras que los valores negativos indican que el vehículo se desplaza hacia la izquierda. Como ilustración, el área de error se representa visualmente en la Figura 3.7, donde el área de error negativa está representada por la parte sombreada en verde.

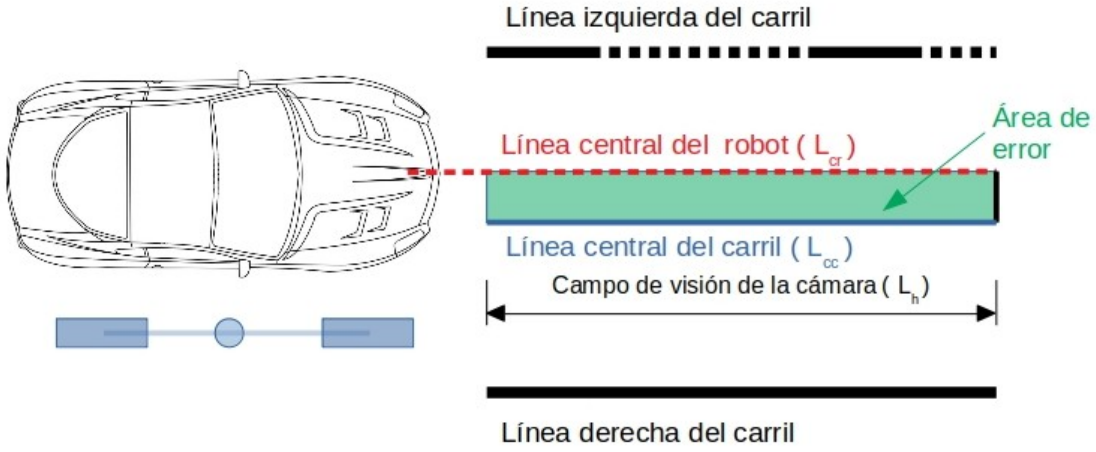


Figura 3.7: Definición del área de error.

El área de cada lado de carril se encuentra en forma de imagen, como se muestra en la Figura 3.8, la diferencia de esas dos áreas dará como resultado el área de error. Se evalúa la distribución de los píxeles en la anchura y la altura de la imagen. Esta región se determina mediante una función de OpenCV. Para determinar el área de error en un sistema métrico decimal, el número de píxeles se multiplica por un factor de conversión, como en la ecuación 3.2.

$$E_{area\_cm} = E_{area\_px} \left(\frac{Altura_{carril} \cdot Ancho_{carril}}{Altura_{px} \cdot Ancho_{px}} \right) \tag{3.2}$$

Donde se tiene que:

- E_{area_cm}: Área de error de posición (cm)
- E_{area_px}: Área de error de posición (px)
- $Altura_{carril}$: Altura del carril (cm)
- $Ancho_{carril}$: Ancho del carril (cm)

- $Altura_{imagen}$: Altura de la imagen (px)
- $Ancho_{imagen}$: Ancho de la imagen (px)

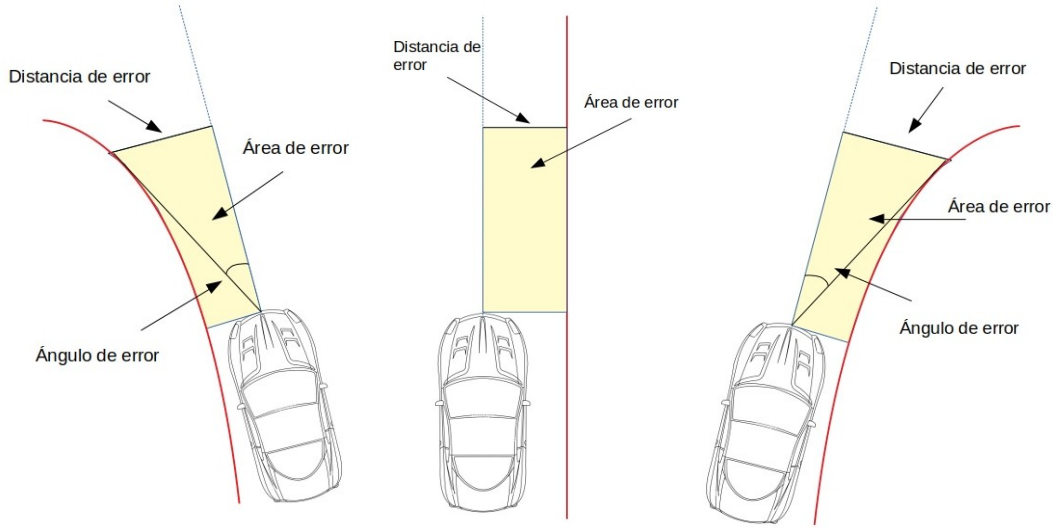
(a) Carril recto. (b) Carril en curva.

Figura 3.8: Ilustración de área de la línea central del robot (rojo) y área de la línea central de los carriles (naranja).

3.3. Ley de Control

El método convencional de un control lateral utiliza la distancia de error y ángulo de error para calcular la posición del robot, y posteriormente determinar el ángulo de dirección para mantener la ruta deseada. Sin embargo, se sugiere un parámetro más eficiente y simple para el controlador lateral. Este parámetro usa solo el área de error y puede lograr no necesitar los dos parámetros clásicos.

El motivo por el que se propone el área de error como parámetro alternativo, se basa en que el área de error tiene información sobre la distancia de error y ángulo de error en varias situaciones. La Figura 3.9 muestra tres situaciones representativas que ocurren durante el mantenimiento de la ruta deseada. La Figura 3.9a muestra la situación donde la distancia de error y el ángulo de error existen simultáneamente y podemos encontrar que el área de error también existe. Si aumenta la distancia de error o el ángulo de error, también aumenta el área de error. La Figura 3.9b muestra la situación cuando solo existe la distancia de error y por ende también el área de error. Finalmente, la Figura 3.9c muestra una situación similar a la Figura 3.9a, pero con una dirección opuesta. Como se observa, los parámetros de ángulo de error y distancia de error pueden ser reemplazados por el parámetro del área de error.



(a) (b) (c)

Figura 3.9: Definición del área de error durante el mantenimiento de la ruta deseada, a) Existe distancia y ángulo de error, b) Existe distancia de error, c) Existe distancia y ángulo de error. Nótese que, en todos los casos, existe el área de error.

Conociendo el estado deseado en el próximo periodo de carril, es posible calcular las acciones de control necesarias para que el robot móvil siga la trayectoria de referencia con un buen rendimiento. Se asume que el robot móvil se mueve en un plano horizontal sin deslizarse. A partir de las ecuaciones referentes al control Stanley y Pure-Pursuit definidas en el capitulo anterior, se sustituyen los parámetros de error por el área de error. Lo que nos da las siguientes ecuaciones 3.3 y 3.4, para controlar el ángulo de dirección del robot móvil (δ).

Control Stanley

$$\delta = K_1 E_{area} + K_2 \arctan\left(\frac{K_3 E_{area}}{V}\right) \tag{3.3}$$

Control Pure-Pursuit

$$\delta = K_1 \arctan\left(\frac{2L \sin\left(E_{area}\right)}{l_d}\right) \tag{3.4}$$

La ecuación 3.3 es una fórmula del controlador Stanley. K_1, K_2, K_3, son valores de ganancia y V es una velocidad longitudinal del robot. La ecuación 3.4 es una fórmula del controlador Pure Pursuit. K_1 es un valor de ganancia, L es la distancia entre ejes del robot y l_d es la distancia de anticipación. Para ambas formulas, E_{area} es el área de error, el

área de que tan separado se encuentra el centro del robot y la línea central de los carriles que delimitan la pista. Esta área de error debe converger a cero para que el vehículo siga correctamente la ruta deseada.

El diagrama de bloques de la Figura 3.10 representa al sistema que se desea controlar.

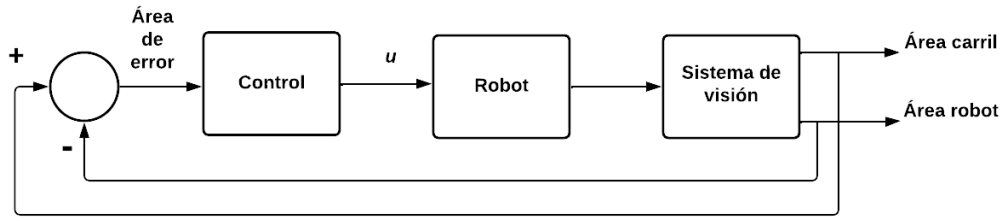


Figura 3.10: Diagrama de bloques del sistema en lazo cerrado.

Algoritmo 2 Búsqueda por Ventana Deslizante

Entrada: Imagen binaria
Salida: Límites del carril

1: Obtener anchura (w) y altura (h) de la imagen
2: Calcular el histograma de la imagen (H_{img}) de píxeles blancos por columna
3: Encontrar el punto máximo en H_{img} para la mitad derecha (r_{bx})
4: Encontrar el punto máximo en H_{img} para la mitad izquierda (l_{bx})
5: Defina el número de ventanas a usar (n_w)
6: Defina la altura de la ventana como: $w_h = |h/n_w|$
7: Defina el ancho de la ventana w_w
8: Defina la coordenada x actual de la línea izquierda como: $l_{cx} = l_{bx}$
9: Defina la coordenada x actual de la línea derecha como: $r_{cx} = r_{bx}$
10: Defina el número mínimo de píxeles para centrar la ventana de búsqueda (m_p)
11: **Para** $c_w = 1 \cdots n_w$ **hacer**
12: Establecer el margen inferior de la ventana como: $b_w = h - c_w * w_h$
13: Establecer el margen superior de la ventana como: $t_w = h - (c_w + 1) * w_h$
14: Establecer margen izquierdo de la ventana izquierda como: $l_{lm} = l_{cx} - w_w$
15: Establecer margen izquierdo de la ventana derecha como: $l_{rm} = l_{cx} + w_w$
16: Establecer margen derecho de la ventana izquierda como: $r_{lm} = r_{cx} - w_w$
17: Establecer margen derecho de la ventana derecha como: $r_{rm} = r_{cx} + w_w$
18: Obtener todos los puntos no nulos (p_x, p_y) que: $(p_y \geq b_w) \wedge (p_y < t_w) \wedge (p_x \geq l_{lm}) \wedge (p_x < l_{rm})$ y añadirlos a l_p
19: Obtener todos los puntos no nulos (p_x, p_y) que: $(p_y \geq b_w) \wedge (p_y < t_w) \wedge (p_x \geq r_{lm}) \wedge (p_x < r_{rm})$ y añadirlos a r_p
20: Defina el número de elementos en l_p como n_{lp}
21: Defina el número de elementos en r_p como n_{rp}
22: Fijar el centro de la ventana derecha (wr_x, wr_y) como: $wr_x = r_{cx}$ y $wr_y = b_w + ((t_w - bw)/2)$
23: Añadir (wr_x, wr_y) para R_{centro}
24: Fijar el centro de la ventana izquierda (wl_x, wl_y) como: $wl_x = l_{cx}$ y $wl_y = b_w + ((t_w - bw)/2)$
25: Añadir (wl_x, wl_y) para L_{centro}
26: **Si** número de elementos en $l_p > m_p$ **entonces**
27: l_{cx} = media de todos p_x en l_p
28: **Fin Si**
29: **Si** número de elementos en $r_p > m_p$ **entonces**
30: r_cx = media de todos p_x en r_p
31: **Fin Si**
32: **Fin Para**
33: **Return** L_{centro}, R_{centro}

Capítulo 4

Resultados experimentales

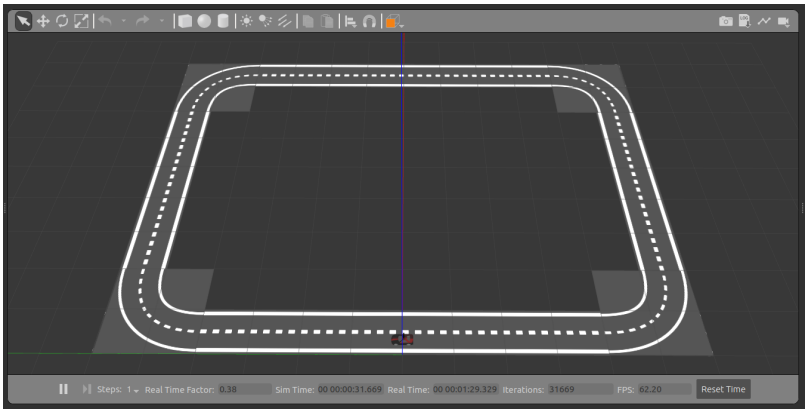
En este capítulo se describen los resultados de la metodología propuesta para el desarrollo del control lateral, mostrada en la Figura 3.1. Dicha metodología fue puesta a prueba en escenarios virtuales y uno real. En el siguiente enlace [32], es posible consultar los vídeos de las pruebas realizadas para cada sección.

4.1. Resultados en escenarios virtuales

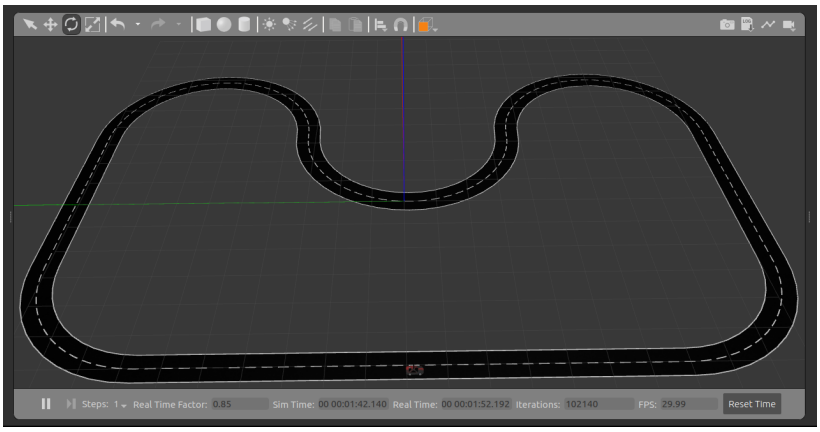
4.1.1. Condiciones de implementación en escenarios virtuales

Para examinar la funcionalidad de la técnica sugerida en diversas condiciones, simulamos el método propuesto en dos pistas. En todas las circunstancias, la velocidad constante del robot es de 0.6 m/s. A continuación se detallan los resultados de cada una de estas simulaciones. El objetivo principal del robot móvil es permanecer en el carril; para la prueba, el vehículo se sitúa primero en el carril con un rumbo directo respecto a la dirección del mismo (véase la Figura 4.1).

La ley de control se codificó en ROS como un nodo, y las pruebas se llevaron a cabo en el simulador Gazebo utilizando pistas como las que se muestran en la Figura 4.1. Debido a las características del robot móvil del simulador, el ángulo de dirección u es tal que si $-23^o \leq u < 0^o$, el vehículo girará a la derecha, y si $0^o < u \leq 23^o$, el vehículo girará a la izquierda. La anchura de las vías de la carretera es de 80 cm, por lo que cada carril mide 40 cm. Según los parámetros del modelo matemático dados en el capítulo anterior, la longitud del robot es $L = 0.4\ m$.



(a) Pista 1: Circular.



(b) Pista 2: Con curvas pronunciadas.

Figura 4.1: Pistas para el proceso de validación del control por medio de la simulación.

4.1.2. Control Stanley

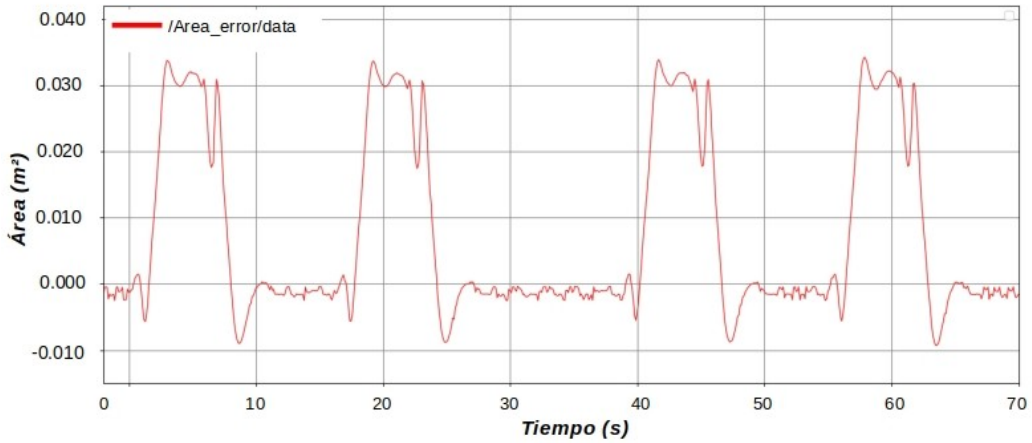
Para evaluar el rendimiento del controlador Stanley de la ecuación 2.9 con θ sustituido por el área de error (E_{area}) como en la ecuación 3.3, la prueba se realiza en dos pistas diferentes con los valores de ganancia idénticos K_1, K_2 y $K_3 = 2$.

La ley de control empleada en el método Stanley es la siguiente:

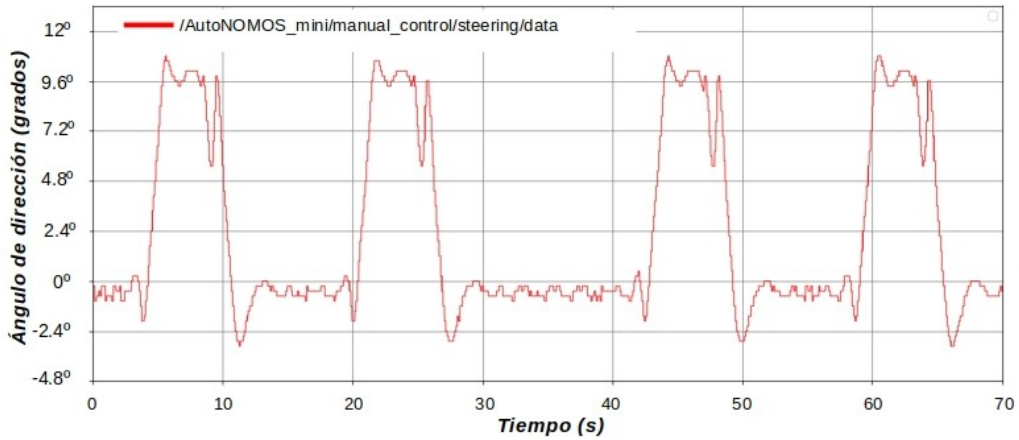
$$u = K_1 E_{area} + K_2 \arctan\left(\frac{K_3 E_{area}}{V}\right) \tag{4.1}$$

Pista 1:

Utilizando el control Stanley, se obtuvieron las señales mostradas en la Figura 4.2. La Figura 4.2a corresponde al área de error, mientras que la Figura 4.2b representa el cambio de ángulo de dirección medido en grados. Como se puede observar existe una relación proporcional entre las dos señales así que una estrategia de control lineal es posible. Utilizando este controlador, el área de error (E_{area}) se mantuvo entre 0.035 y -0.01 m^2 mientras que el ángulo de dirección (u) estuvo entre 11^o y -2.5^o.



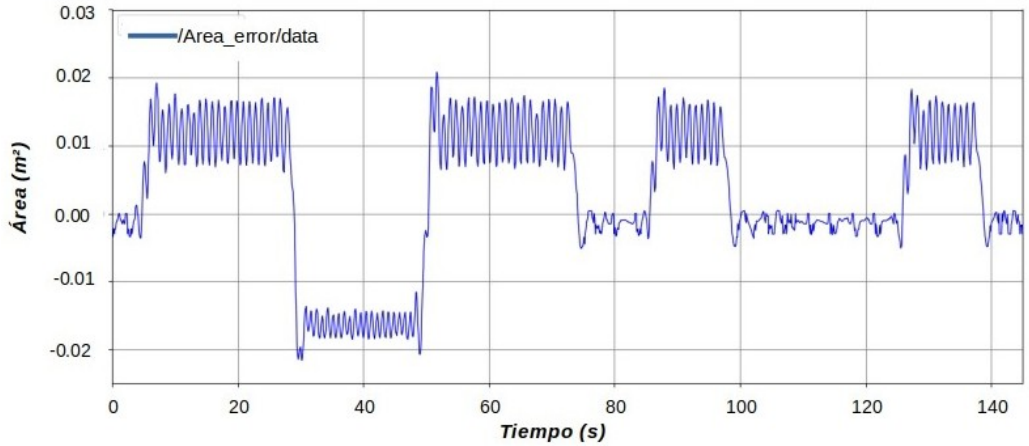
(a) Área de error.



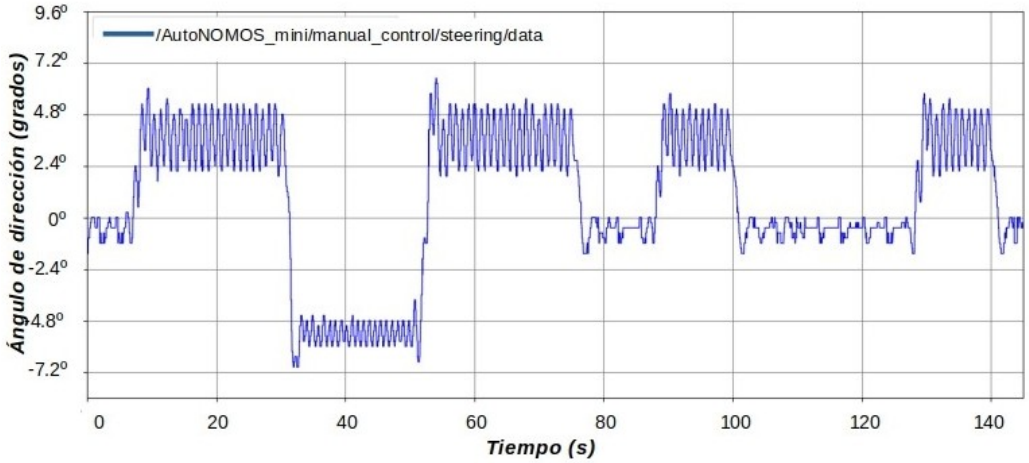
(b) Ángulo de dirección del robot.

Figura 4.2: Control Stanley: Evolución del área de error en el tiempo. Si se cumple que $-0.04 \leq E_{area} \leq 0.04$ el automóvil se mantendrá dentro de su carril.

Pista 2:



(a) Área de error

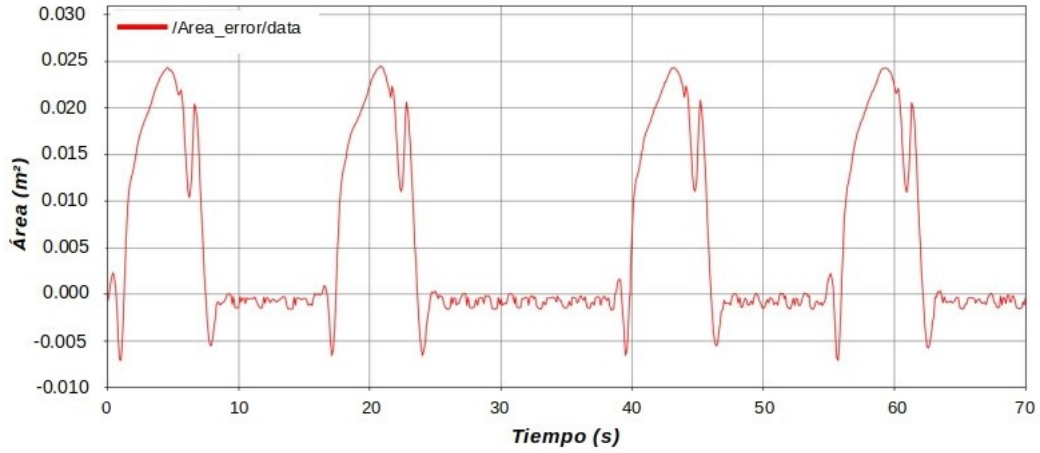


(b) Ángulo de dirección del robot.

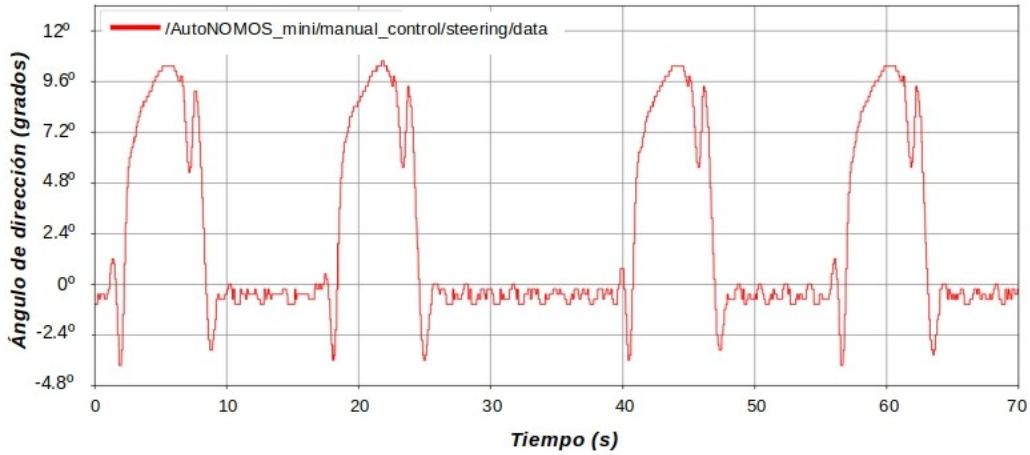
Figura 4.3: Control Stanley: Evolución del área de error y ángulo de dirección con respecto al recorrido de la pista 2.

Por otro lado, para el recorrido en la pista 2 (mostrada en la Figura 4.1b). El gráfico de la Figura 4.3a representa la evolución del área de error en el tiempo, mientras que el gráfico de la Figura 4.3b representa el ángulo de dirección del robot. Como se ve en la Figura 4.3b, el error de posición aumenta cuando el vehículo hace una curva y baja hasta casi cero cuando el vehículo viaja en línea recta. Dado que $-0.04 \leq E_{area} \leq 0.04$, el robot permanece dentro de su carril durante todo su recorrido.

4.1.3. Control Pure-Pursuit



(a) Área de error.



(b) Ángulo de dirección del robot.

Figura 4.4: Control Pure-Pursuit: Evolución del área de error y el ángulo de dirección en el recorrido de la pista 1.

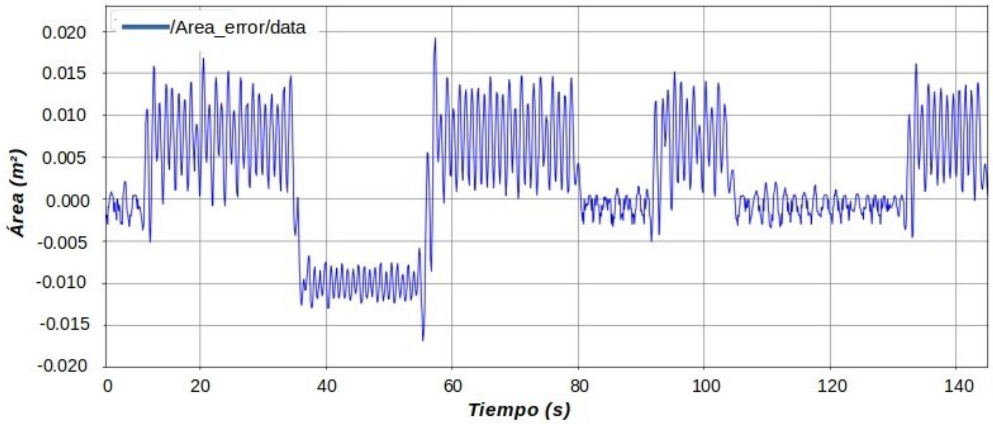
Para el controlador lateral Pure-Pursuit de la ecuación 3.4, la ganancia asignada a este controlador es: $K_1 = 2$, con los valores: $L = 0.4\ m$ y $l_d = 0.2\ m$. La ley de control se muestra en la ecuación 4.2. A diferencia de los resultados anteriores, la señal de área de error disminuyó de amplitud, la señal tiene una región comprendida en 0.025 y $-0.009\ m^2$.

$$u = K_1 \arctan\left(\frac{2L\sin\left(E_{area}\right)}{l_d}\right) \tag{4.2}$$

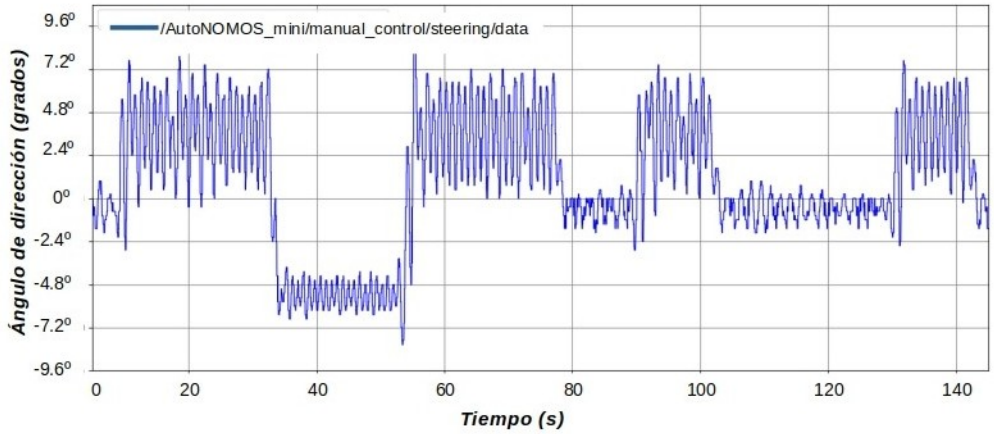
Pista 1:

Para la prueba realizada en la pista 1, el control cinemático muestra resultados similares al caso de la pista 1 con el control Stanley. Las señales se muestran en la Figura 4.4. Por otro lado, los resultados del área de error y ángulo de dirección mostradas en las Figura 4.4 presentan oscilaciones ligeramente inferiores a las presentadas en la pista 1 con el control Stanley.

Pista 2:



(a) Área de error.



(b) Ángulo de dirección del robot.

Figura 4.5: Control Pure-Pursuit: Evolución del área de error y el ángulo de dirección en el recorrido de la pista 2.

Las Figuras 4.5a y 4.5b muestran las salidas de área de error y ángulo de dirección durante todo el recorrido. Aun en el caso más crítico en este escenario, donde el área de error máximo es $-0.017\ m^2$ se puede apreciar como los valores de la acción de control varían de forma suave sin generar giros bruscos y el área de error lateral se mantiene acotado con un error máximo de $-0.020 < E_{area} < 0.017\ m^2$.

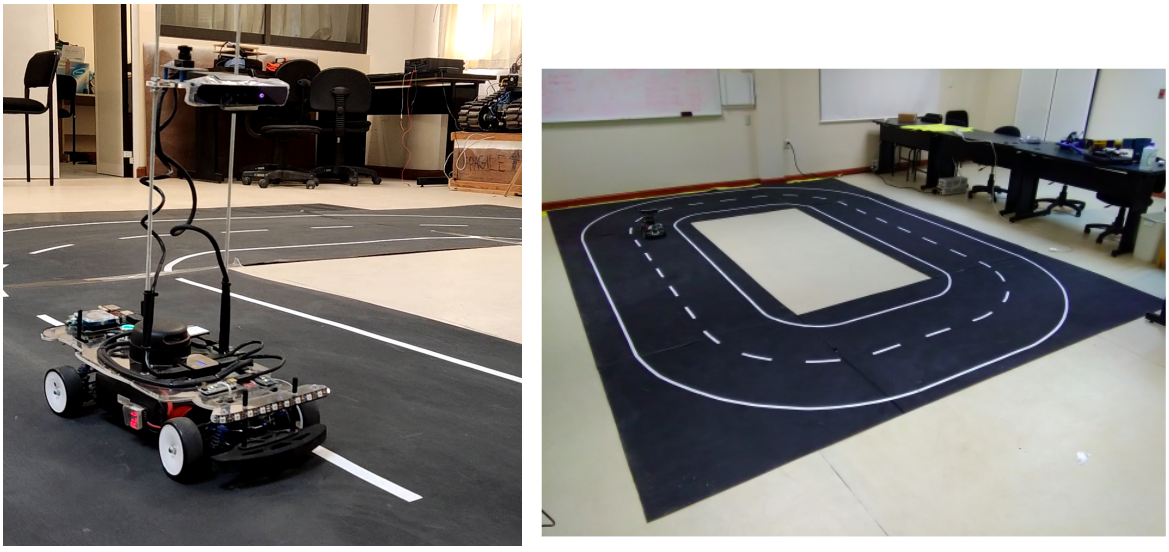
Los resultados obtenidos corroboran las bondades del controlador propuesto, mostrando que el robot móvil sigue la trayectoria de referencia de una manera precisa. La próxima sección se presentará la implementación en el robot móvil AutoNOMOS mini V2. Debido a que se implementan dos controles, es importante observar las diferencias en el comportamiento de cada uno de ellos, esto con el objetivo de identificar el control con el mejor desempeño. El control Stanley presenta un área de error máximo de $0.035\ m^2$ con un grado de giro máximo de 11^o, mientras que el control Pure-Pursuit de $0.025\ m^2$ y 10^o, ambos en la pista 1. Comparando el valor de estos dos controles y apoyados de su respectiva gráfica, se observa que el control Stanley presenta mayor error y mayor grado de giro. Con base a esto, se concluye que el control con mejor desempeño es el control lateral Pure-Pursuit.

4.2. Resultados en la pista real

4.2.1. Condiciones de implementación en la pista real

En la simulación, los controladores Stanley y Pure Pursuit han logrado mantener el robot móvil dentro de las líneas que delimitan el carril. El control Pure Persuit refleja un área de error y ángulo de dirección menor en comparación con el control Stanley. El consenso parece ser que el controlador Stanley no es tan suave, estable o robusto como Pure Pursuit. El control Pure Pursuit puede ajustar la distancia de anticipación, o distancia del punto de la trayectoria de referencia más cercano al vehículo y el punto objetivo. Finalmente el control funciona bien en casos de baja velocidad. Por lo tanto, se ha optado por realizar la implementación en el robot móvil real con un solo controlador, el control lateral Pure-Pursuit. La ganancia del controlador es $K_1 = 5$, esta ganancia fue obtenida de acuerdo a las respuestas en la simulación.

El vehículo tiene una distancia entre las ruedas delanteras y traseras $L = 27\,cm$. La pista mostrada en la Figura 4.6b tiene un ancho de 80 cm por lo que cada carril mide 40 cm. Todas las curvas de la pista tienen un radio externo de 1.8 m y un radio interno de 1.0 m. El ensayo realizado es con una velocidad mantenida entre 0.6 y 0.75 m/s. El periodo de muestreo que se presenta en las gráficas de resultados, equivale al tiempo en que el robot da una vuelta completa en su respectiva pista.



(a) (b)

Figura 4.6: Implementación: (a) Modelo robot móvil AutoNONOS Mini V2. (b)Pista de pruebas ubicada en el Laboratorio de Robótica Inteligente.

El tópico de ROS que conduce al robot está programado de tal modo que, si el ángulo de dirección u cumple que $-23.97^o \leq u < 0^o$, el vehículo girará a la derecha; por otro lado si $0^o < u \leq 22.34^o$ el giro es hacia la izquierda. La distancia de anticipación se establece en $l_d = 40\ cm$. Por lo tanto, se usa como ley de control:

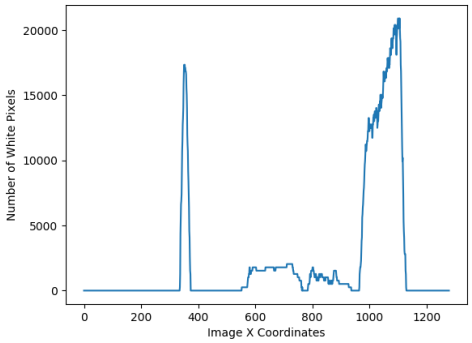
$$u = K_1 \arctan\left(\frac{2L\sin\left(E_{area}\right)}{l_d}\right) \tag{4.3}$$

La Figura 4.7 muestra el método para la obtención de la líneas de carril en la implementación real.

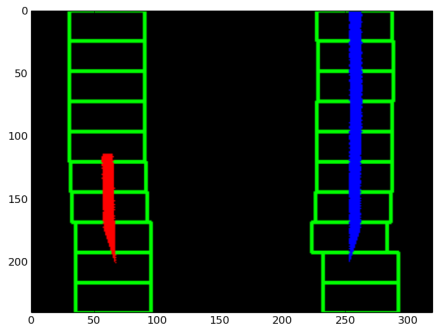
4.2.2. Control Pure-Pursuit

Con ayuda del controlador Pure Pursuit, el robot móvil recorrió la pista sin salirse de su carril, y las señales que se obtuvieron durante su segunda vuelta se ilustran en los resultados. Las señales en la Figura 4.8 muestran el área de error y la señal de ángulo de dirección. En primera instancia, el robot se situó en una parte recta de la pista, realiza un tramo recto, seguido de una curva a la izquierda. El área de error muestra un comportamiento positivo al entrar a la curva, con cambios suaves y una reducción del área de error en tramo recto.

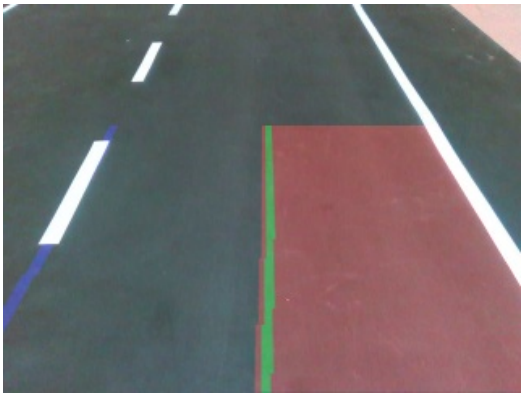
(a) Transformación de perspectiva.



(b) Histograma.



(c) Ventana Deslizante.

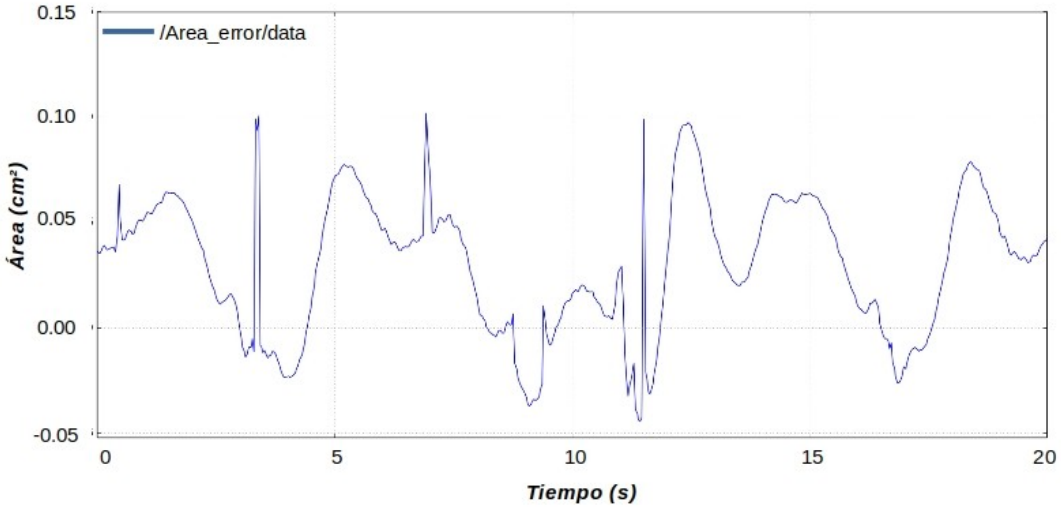


(d) Área de error.

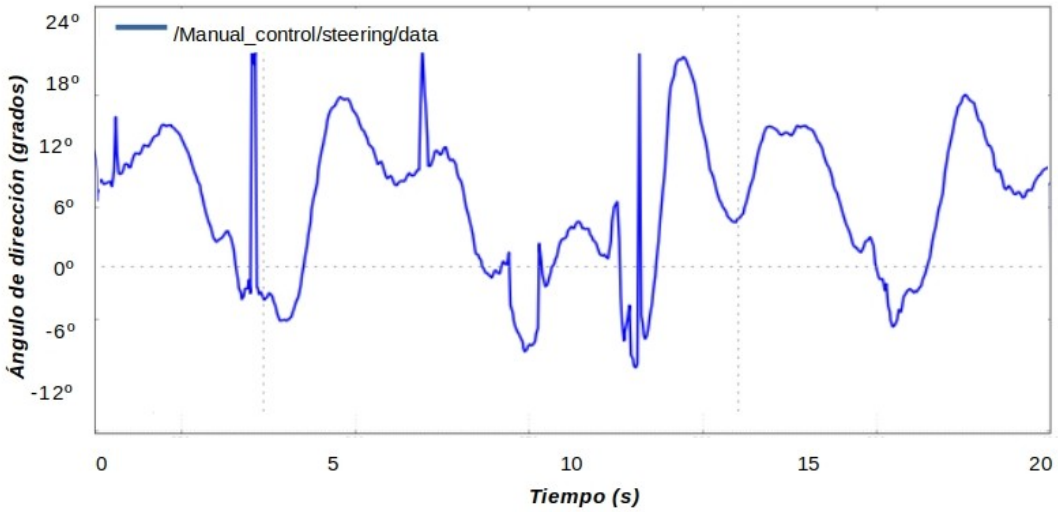
Figura 4.7: Proceso de extracción de características del carril.

En cuanto a la acción de control, se puede observar que esta varía de forma suave y sin generar cambios bruscos en la dirección. El valor máximo del área de error se alcanza en las curvas y es de 0.10 cm^2, mientras que el ángulo de dirección máximo se sitúa en 22.3^o. Esta área de error es pequeña comparada con la distancia que existe entre las ruedas del robot móvil (0.20 cm). Al seleccionar la distancia de anticipación (l_d) igual a 40 cm, este valor compensa el retardo que se obtiene al realizar el proceso de detección de las líneas del carril y la obtención de área de error.

Así pues el controlador diseñado fue lo suficientemente robusto como para mantener el vehículo dentro de su carril en todo momento y por lo tanto, la metodología empleada ofrece una solución sencilla y fácil de implementar al problema de la conducción autónoma de un vehículo tipo Ackerman. Sin embargo, dicha robustez está limitada a la condición inicial (en un inicio el robot debe ver las lineas del carril), de otro modo, el robot se quedará con una velocidad 0 m/s.



(a) Área de error $[cm^2]$.



(b) Ángulo de dirección [Grados].

Figura 4.8: Pista real: Resultados.

4.3. Dificultades encontradas

Si bien el objetivo general se cumplió, no fue una tarea fácil. A continuación se presentan los problemas más significativos de las diferentes áreas que se superaron en el desarrollo de este trabajo de investigación.

1. Las primeras simulaciones realizadas con los controladores no contemplaban arranques o terminaciones suaves en las trayectorias, lo que tuvo como consecuencia que al momento de implementarlos experimentalmente, se observara que el robot tenía arranques bruscos y frenados forzados. Debido a esto, se optó por mantener el robot móvil a una velocidad comprendida entre 0.6 y 0.75 m/s.

2. En el momento de la implantación, las variaciones inducidas por el reflejo del sol en el laboratorio de robótica donde se colocó la pista de pruebas hicieron que la cámara detectara líneas de carril falsas, imposibilitando la adquisición de un valor preciso del área de error.

Capítulo 5

Conclusiones y trabajos a futuro

5.1. Conclusiones

En este trabajo de tesis se presentó un método de control lateral basado en visión por computadora. Este método consiste en el uso del área de error de posición como la variable de entrada al controlador. Es importante recalcar que el área de error de posición como atributo es algo novedoso, pues en la literatura no existe evidencia de trabajos que hagan uso de este tipo de propiedad para la tarea de control lateral. Tras realizar los experimentos necesarios se concluyó que efectivamente el uso del área de error como atributo permite al control lateral alcanzar el objetivo propuesto que se basa en recorrer una pista sin salirse de las lineas que lo delimitan. En contraste con controladores ocupados para el mismo objetivo, nuestro trabajo al utilizar sólo el área de error llega a ser fácilmente adaptable a otros modelos con características similares.

Puede observarse que el sistema de control propuesto depende del procesamiento de la imagen de entrada, sin embargo, es independiente de los demás sensores. El algoritmo para la detección de carriles ha logrado la detección de todos los límites de carril visibles, incluyendo la detección de puntos falsos positivos debido al reflejo de la luz en la pista real. Se requiere realizar mejoras en el algoritmo de detección de carril para algunas situaciones concretas, como puede ser en curvas cerradas, que el ángulo de visión de la cámara no pueda detectar.

Por otro lado, haber hecho uso de librerías y herramientas de desarrolladores de ROS y OpenCV ha facilitado el desarrollo de las soluciones, ya que se cuenta con unas funciones específicas para cada aplicación y una comunidad que brinda posibles consultas.

Desde el punto de vista del hardware, el empleado ha presentado una importante limitación al no tener suficiente capacidad de cómputo, especialmente en las tareas que empleaban

el tratamiento de imágenes y su posterior procesado. Ello ha llevado a aumentar la mirada hacia adelante de la cámara, de tal forma que las señales que se envían tarde, puedan ser usadas en el momento preciso. Pero, a pesar de ello, el robot móvil pudo circular por la pista sin salirse del carril.

Una aclaración importante sobre el comportamiento del robot móvil a lo largo de las pruebas es que el campo de visión restringido de la cámara durante el seguimiento de la trayectoria provoca la incapacidad de reconocer las líneas de carril en algunas curvas, lo que lleva al control a adoptar un curso en línea recta.

Una característica atractiva de este tipo de controles es que su implementación es simple y efectiva. A través de los experimentos realizados se puede concluir que el área de error entre las trayectorias real y deseada es muy pequeña, con respecto a las dimensiones del robot. La capacidad del robot móvil para conducir depende de su velocidad: en velocidades bajas, el robot puede girar de manera efectiva y segura, a diferencia de velocidades altas.

Con la realización de este trabajo de tesis, se obtuvieron conocimientos nuevos en el área de procesamiento de imágenes, visión por computadora, control, robótica y programación en lenguaje Python.

5.2. Trabajos a futuro

Tomando en cuenta las restricciones y desventajas que presenta el método de control, se han identificado algunas tareas que se desprenden de este trabajo sobre las cuales se pretende explorar a futuro. Estas tareas se presentan a continuación:

En cuanto a la identificación de líneas, los resultados obtenidos para tramos rectos y curvos han sido positivos, por lo que el futuro debería centrarse en ampliar el método a cruces, rotondas y, quizás, circunstancias más difíciles. La aproximación polinómica arroja resultados precisos, pero la categorización de las líneas de antemano podría ser crucial. Por otro lado, se recomienda potenciar la aproximación "traking", que trata de predecir la trayectoria designada por las líneas de la carretera, lo que resulta especialmente beneficioso en escenarios como las curvas y las rotondas. También se proponen situaciones con añadidos que condicionan la identificación de las líneas, como los pasos de peatones, utilizando estos condicionantes de detección de señales. Por último, en esta tesis sólo se tiene en cuenta el control lateral. Sin embargo, es posible ampliar las variables de decisión, dotando al controlador de más factores para tener un control total, es importante combinar los controles lateral (giro) y vertical (velocidad). Para aumentar la autonomía del robot, es recomendable diseñar aplicaciones de alto nivel que consideren retos como la detección de obstáculos, el reconocimiento de señales de tráfico, el mapeo y la localización.

Bibliografía

[1] M. J. Walker, "Top 10 strategic technology trends for 2017," in *Intelligent Things. Gartner*, 2017.

[2] J. Naranjo, C. Gonzalez, R. Garcia, T. de Pedro, and R. Haber, "Power-steering control architecture for automatic driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 4, pp. 406–415, 2005.

[3] A. bar hillel, R. Lerner, D. Levi, and G. Raz, "Recent progress in road and lane detection: A survey," *Machine Vision and Applications*, vol. 25, 04 2014.

[4] B. Varma, N. Swamy, and S. Mukherjee, "Trajectory tracking of autonomous vehicles using different control techniques(pid vs lqr vs mpc)," in *2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)*, 2020, pp. 84–89.

[5] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 3, pp. 566–580, 2007.

[6] H. Peng and M. Tomizuka, "Vehicle lateral control for highway automation," in *1990 American Control Conference*, 1990, pp. 788–794.

[7] S.-H. Lee and C. C. Chung, "Multilevel approximate model predictive control and its application to autonomous vehicle active steering," in *52nd IEEE Conference on Decision and Control*, 2013, pp. 5746–5751.

[8] A. Lombard, J. Buisson, A. Abbas-Turki, S. Galland, and A. Koukam, "Curvature-based geometric approach for the lateral control of autonomous cars," *Journal of the Franklin Institute*, vol. 357, 07 2020.

[9] L. Hammarstrand, M. Fatemi, F. García-Fernández, and L. Svensson, "Long-range road geometry estimation using moving vehicles and roadside observations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 8, pp. 2144–2158, 2016.

[10] S. G. Yi, C. M. Kang, S.-H. Lee, and C. C. Chung, "Vehicle trajectory prediction for adaptive cruise control," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 59–64.

[11] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing," in *2007 American Control Conference*, 2007, pp. 2296–2301.

[12] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, and P. Mahoney, "Stanley: The robot that won the darpa grand challenge." *J. Field Robotics*, vol. 23, pp. 661–692, 01 2006.

[13] H. Zhang and J. Wang, "Vehicle lateral dynamics control through afs/dyc and robust gain-scheduling approach," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 1, pp. 489–494, 2016.

[14] G. Han, W. Fu, W. Wang, and Z. Wu, "The lateral tracking control for the intelligent vehicle based on adaptive pid neural network," *Sensors*, vol. 17, p. 1244, 05 2017.

[15] J. Yang and N. Zheng, "An expert fuzzy controller for vehicle lateral control," in *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*, 2007, pp. 880–885.

[16] X. Huang, H. Zhang, G. Zhang, and J. Wang, "Robust weighted gain-scheduling h_∞ vehicle lateral motion control with considerations of steering system backlash-type hysteresis," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 5, pp. 1740–1753, 2014.

[17] C. Latrach, M. Kchaou, A. El Hajjaji, and A. Rabhi, "Robust h fuzzy networked control for vehicle lateral dynamics," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, 2013, pp. 905–910.

[18] W. Rui, S. Yi-Ming, L. Mei-Tong, and Z. Hao, "Research on bus roll stability control based on lqr," in *2015 International Conference on Intelligent Transportation, Big Data and Smart City*, 2015, pp. 622–625.

[19] S.-H. Lee and C. C. Chung, "Predictive control with sliding mode for autonomous driving vehicle lateral maneuvering," in *2017 American Control Conference (ACC)*, 2017, pp. 2998–3003.

[20] G. Tagne, R. Talj, and A. Charara, "Higher-order sliding mode control for lateral dynamics of autonomous vehicles, with experimental validation," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 678–683.

[21] A. Norouzi, M. Masoumi, A. Barari, and S. F. Sani, "Lateral control of an autonomous vehicle using integrated backstepping and sliding mode controller," *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics*, vol. 233, no. 1, pp. 141–151, 2019.

[22] J. S. Kim, S.-H. Lee, and C. C. Chung, "Lane change control with optimal time-varying sliding mode in automated driving vehicle," in *2020 American Control Conference (ACC)*, 2020, pp. 430–435.

[23] M. Huang, M. Zhao, P. Parikh, Y. Wang, K. Ozbay, and Z.-P. Jiang, "Reinforcement learning for vision-based lateral control of a self-driving car," in *2019 IEEE 15th International Conference on Control and Automation (ICCA)*, 2019, pp. 1126–1131.

[24] J. He, S. Sun, D. Zhang, G. Wang, and C. Zhang, "Lane detection for track-following based on histogram statistics," in *2019 IEEE International Conference on Electron Devices and Solid-State Circuits (EDSSC)*, 2019, pp. 1–2.

[25] L. E. Sucar and G. Gómez, *Visión computacional*. Instituto Nacional de Astrofísica, Optica y Electrónica. Puebla, México, 2011.

[26] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[27] F. U. B. Dahlem Center for Machine Learning Robotics, "Automodelcarwiki," https://github.com/automodelcar/automodelcarwiki/wiki/, 2016, [Online; Último acceso agosto-2022].

[28] ROS, http://wiki.ros.org/es, 2021, [Online; Último acceso agosto-2022].

[29] R. ITAM, "Ek_autonomos_sim," https://github.com/ITAM-Robotica/EK_AutoNOMOS_Sim, 2019, [Online; Último acceso agosto-2022].

[30] M. Haque, M. Islam, K. Alam, H. Iqbal, and M. Shaik, "A computer vision based lane detection approach," *International Journal of Image, Graphics and Signal Processing*, vol. 11, pp. 27–34, 03 2019.

[31] Z. Liu, S. Wang, and X. Ding, "Roi perspective transform based road marking detection and recognition," in *2012 International Conference on Audio, Language and Image Processing*, 2012, pp. 841–846.

[32] K. Guzmán, "Control lateral," https://github.com/guzduran/Lateral-Control_error-area.git, 2022, [Online; Último acceso diciembre-2022].