Simulación y planificación de rutas para vehículos autónomos en entornos urbanos a escala inspirados en la Ciudad de Guatemala

Luis Pedro Garrido Jurado

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Simulación y planificación de rutas para vehículos autónomos en entornos urbanos a escala inspirados en la Ciudad de Guatemala

Trabajo de graduación presentado por Luis Pedro Garrido Jurado para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2024

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Simulación y planificación de rutas para vehículos autónomos en entornos urbanos a escala inspirados en la Ciudad de Guatemala

Trabajo de graduación presentado por Luis Pedro Garrido Jurado para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2024

Vo.Bo.:

(f) _____

M. Sc. Carlos Esquit

Tribunal Examinador:

(f) _____

M.Sc. Carlos Esquit

(f) _____

M. Sc. Miguel Enrique Zea Arenales

(f) _____

Ing. Kurt Emmanuel Kellner

Fecha de aprobación: Guatemala, 13 de febrero de 2025.

Prefacio

Al finalizar esta etapa tan significativa de mi vida, deseo expresar mi más profundo agradecimiento a las personas que han estado a mi lado, brindándome su apoyo incondicional.

En primer lugar, a mi mamá, Evelyn Jurado Álvarez, por su amor y respaldo inquebrantables. Sin su apoyo constante y su ejemplo de fortaleza, este logro no habría sido posible. A mi hermano, Jorge Augusto Garrido Jurado, por su compañía, ánimo y apoyo a lo largo de este camino. Ambos han sido pilares fundamentales en mi vida, y a ellos les debo gran parte de este triunfo.

A mi asesor, MSc. Miguel Enrique Zea Arenales, mi más sincero agradecimiento por su dedicación y esfuerzo continuo. Su guía y compromiso fueron clave en el desarrollo de esta tesis, y sus enseñanzas quedarán siempre presentes.

A mis amigos de la carrera, quienes compartieron conmigo momentos inolvidables y me brindaron su amistad en los momentos más desafiantes. A través de cada salida, clase, proyecto y conversación, ustedes fueron una fuente invaluable de motivación.

Quiero también recordar a mi padre, Jorge Alberto Garrido Molina, quien, aunque ya no está conmigo, fue la persona que desde el principio me motivó a convertirme en ingeniero. Su memoria sigue siendo una gran inspiración en mi vida.

A mi querida perrita Jackie, quien me acompañó durante tantas jornadas de estudio, brindándome su lealtad y amor hasta el último día.

Finalmente, extiendo mi agradecimiento al resto de mi familia, abuela, hermano, tías, tío, primas y primos, quienes siempre han estado allí, ofreciendo su apoyo y cariño en cada etapa de mi vida.

A todos ustedes, gracias. Este logro es también suyo.

Índice

Lista de figuras

Lista de cuadros

Resumen

Este estudio desarrolla un sistema de simulación, planificación de rutas y control para vehículos autónomos en entornos urbanos, adaptado a las condiciones viales de la Ciudad de Guatemala. La creciente importancia de los vehículos autónomos para mejorar la seguridad y eficiencia del transporte motiva la investigación, enfrentando desafíos únicos de infraestructura y tráfico local.

Se digitalizó un entorno vial a escala mediante un grafo direccional que replica las carreteras guatemaltecas. Se implementaron algoritmos de búsqueda de rutas optimizados por distancia, con un tiempo de procesamiento promedio de 1.8 milisegundos. Se desarrollaron los controladores PID con acercamiento exponencial y Lyapunov, evaluados mediante simulaciones virtuales y pruebas físicas con el sistema OptiTrack para monitoreo preciso.

Los resultados de la simulación de múltiples vehículos mostraron que el controlador Lyapunov tiene una alta precisión al tener una semejanza media de trayectoria de 99.6 % y 99.2 % en simulación virtual y física respectivamente; mayores al PID que obtuvo 93.4 % y 91.8 %. Sin embargo, el PID demostró mayor estabilidad al mantenerse debajo de la velocidad máxima el 100 % del tiempo, comparado con el Lyapunov (80.7 % y 86.1 %). El controlador Lyapunov manejó hasta dos vehículos de manera estable, mientras que PID gestionó hasta tres con mayor eficiencia.

En conclusión, el estudio avanza en el desarrollo de vehículos autónomos para entornos urbanos complejos, ofreciendo una plataforma adaptada a la Ciudad de Guatemala. Los hallazgos destacan la importancia de seleccionar el controlador adecuado según las necesidades de precisión y estabilidad, facilitando la implementación segura y eficiente de tecnologías autónomas en contextos viales desafiantes.

Abstract

This study develops a simulation, route planning, and control system for autonomous vehicles in urban environments, adapted to the traffic conditions of Guatemala City. The growing importance of autonomous vehicles in improving transportation safety and efficiency motivates this research, addressing unique challenges related to local infrastructure and traffic.

A scaled urban environment was digitized using a directional graph that replicates Guatemalan roads. Route search algorithms optimized for distance were implemented, with an average processing time of 1.8 milliseconds. PID controllers with exponential approach and Lyapunov were developed and evaluated through virtual simulations and physical tests using the OptiTrack system for precise monitoring.

Simulation results with multiple vehicles showed that the Lyapunov controller achieved high precision, with an average trajectory similarity of 99.6 % and 99.2 % in virtual and physical simulations respectively, compared to the PID controller's 93.4 % and 91.8 %. However, the PID controller demonstrated greater stability by maintaining below maximum speed 100 % of the time, compared to Lyapunov's 80.7 % and 86.1 %. The Lyapunov controller managed up to two vehicles stably, while the PID managed up to three more efficiently.

In conclusion, this study advances the development of autonomous vehicles for complex urban environments by offering a platform adapted to Guatemala City. The findings highlight the importance of selecting the appropriate controller based on precision and stability needs, facilitating the safe and efficient implementation of autonomous technologies in challenging traffic contexts.

CAPÍTULO 1

Introducción

El presente trabajo se enfoca en el desarrollo de un sistema de simulación, planificación de rutas y control para vehículos autónomos en entornos urbanos a escala, con especial atención en las condiciones viales específicas de la Ciudad de Guatemala. Se busca replicar las complejidades del entorno vial guatemalteco para proporcionar una plataforma que permita analizar y validar algoritmos de control que faciliten la navegación eficiente y segura de vehículos autónomos, emulando las características del tráfico real. Este proyecto integra tecnologías de simulación y control, abordando tanto aspectos de software como de hardware, y permite una evaluación exhaustiva del desempeño de los vehículos en diversas condiciones viales.

La motivación de este estudio radica en la creciente relevancia de los vehículos autónomos en la movilidad urbana y su potencial para mejorar la seguridad vial, reducir la congestión y optimizar el transporte. La implementación de estos sistemas en entornos reales, especialmente en ciudades con infraestructuras viales desafiantes como la Ciudad de Guatemala, presenta retos técnicos y prácticos significativos. Este trabajo responde a la necesidad de desarrollar soluciones tecnológicas adaptadas a las particularidades de estos entornos, contribuyendo a la investigación en vehículos autónomos y su adaptación a infraestructuras locales complejas. Además, aborda desafíos clave como la planificación de rutas, la evasión de obstáculos y la interacción entre múltiples vehículos autónomos.

Para lograr estos objetivos, se diseñaron y planificaron rutas para los vehículos autónomos dentro de un entorno urbano simulado que considera las características viales locales. Se implementaron y evaluaron diferentes sistemas de control, incluyendo controladores Lyapunov y PID con acercamiento exponencial, que gestionan la navegación autónoma en diversos escenarios y condiciones. Asimismo, se extendió el estudio a la evaluación de múltiples vehículos operando simultáneamente, tanto en simulaciones virtuales como en pruebas físicas, con el fin de analizar su interacción y comportamiento en situaciones de tráfico compartido. El método empleado combina la digitalización de carreteras a escala, la adaptación de algoritmos de búsqueda de rutas y la implementación de sistemas de control avanzados para robots diferenciales. Se utilizó el sistema de captura de movimiento OptiTrack, integra-

do en el ecosistema Robotat, para monitorear y validar el comportamiento de los vehículos en simulaciones físicas, permitiendo una evaluación integral de los sistemas de navegación y control en un entorno seguro y controlado.

CAPÍTULO 2

Antecedentes

En la Universidad del Valle de Guatemala, la investigación de vehículos autónomos a escala comenzó en el 2023, siendo un avance significativo en el estudio y desarrollo de sistemas robóticos. Este esfuerzo inicial ha sido fortalecido por las contribuciones de ingenieros mecatrónicos como Gabriel Fong y Carlos Arribas. El Ing. Fong desarrolló la infraestructura para la evaluación de algoritmos de visión computarizada, mientras que el Ing. Arribas se centró en evaluar la efectividad de estos sistemas. Además, el proyecto Robotat, parte del estudio del Ing. Camilo Perafán, amplió el alcance de la investigación integrando sistemas de captura de movimiento y comunicación inalámbrica, lo que permitió un entorno más robusto para el desarrollo y prueba de tecnologías emergentes. Los antecedentes externos también forman una base crucial para este trabajo. Iniciativas como el proyecto Waymo y estudios sobre la gestión de tráfico urbano con vehículos automatizados y conectados, han proporcionado modelos valiosos para la implementación y evaluación en contextos más amplios. Estos proyectos externos han demostrado la capacidad de los vehículos autónomos para operar de manera segura y eficiente en entornos urbanos complejos, y están documentados en publicaciones relevantes que detallan los avances tecnológicos y los desafíos asociados.

Este proyecto se apoya tanto en los fundamentos establecidos internamente en la Universidad del Valle de Guatemala, como en la tendencia internacional por el desarrollo de los vehículos autónomos. El estudio de los vehículos autónomos a escala se enriquece con estos antecedentes, y también apunta hacia la expansión futura de estas tecnologías en aplicaciones prácticas y teóricas.

2.1. *Self-Driving Cars*

Según [1], los vehículos autónomos han demostrado un impacto significativo en los sistemas de transporte, con potenciales beneficios en términos de eficiencia, conveniencia y seguridad vial. Los vehículos autónomos enfrentan una variedad de escenarios complejos y novedosos que pueden llevar a fallos operativos por la cantidad de kilómetros que se manejan

anualmente. Pese a los progresos tecnológicos, existen retos notables en el funcionamiento de estos sistemas, particularmente en la percepción del entorno, el reconocimiento de objetos y la toma de decisiones en tiempo real frente a condiciones cambiantes.

Las investigaciones realizadas en Asia, Europa y Estados Unidos han adoptado un enfoque multidisciplinario que integra ingeniería y tecnologías de la información, logrando avances en la mejora de la percepción mediante sistemas de mapeo digital, computación avanzada, fusión de sensores y aprendizaje automático. Estos avances permiten a los vehículos autónomos realizar tareas de conducción cada vez más complejas con una mayor autonomía y seguridad, especialmente en entornos controlados y, en ocasiones, en contextos urbanos reales.

Sin embargo, la investigación continúa resaltando la necesidad de mejorar en áreas como la toma de decisiones ante situaciones atípicas, la interpretación de señales sociales y la adaptabilidad a condiciones adversas del clima e infraestructuras viales deficientes. Los desafíos técnicos, legales y éticos requieren la definición de estándares claros, la mejora en la precisión de los sensores y el desarrollo de sistemas capaces de entender y responder a la complejidad del comportamiento humano y las interacciones en la carretera. Los costos y la infraestructura necesaria para el soporte de estos vehículos representan barreras importantes. El campo de los vehículos autónomos es prometedor y podría revolucionar la movilidad.

2.2. Robotat: un ecosistema robótico de captura de movimiento y comunicación inalámbrica

El Robotat es un ecosistema desarrollado en la Universidad del Valle de Guatemala y fue parte del trabajo de graduación del Ing. Camilo Perafán [2]. Este ecosistema, mostrado en la Figura 1 integró un sistema de captura de movimiento y una red de comunicación inalámbrica, superando las limitaciones de proyectos anteriores enfocados en áreas específicas de investigación y permitiendo su aplicación en una gama más amplia de actividades robóticas. La infraestructura de Robotat incluyó el empleo de cámaras y sensores OptiTrack que proporcionaron mediciones precisas de la posición y orientación de los objetos dentro del área de experimentación. La red de comunicación basada en el protocolo MQTT (*Message Queuing Telemetry Transport*) facilitó una interacción eficiente y en tiempo real entre los diversos agentes robóticos. Esto contribuyó significativamente a la experimentación y desarrollo tecnológico en el campo de la robótica. Este ecosistema no solo amplió las capacidades de investigación y desarrollo de los estudiantes y facultad de ingeniería Mecatrónica, sino que también estableció un entorno propicio para la innovación y el intercambio académico en tecnologías emergentes. La implementación de Robotat marcó un hito importante en el desarrollo de infraestructuras robóticas multifuncionales, permitiendo a los usuarios explorar nuevas tecnologías y métodos en un ambiente controlado y colaborativo.

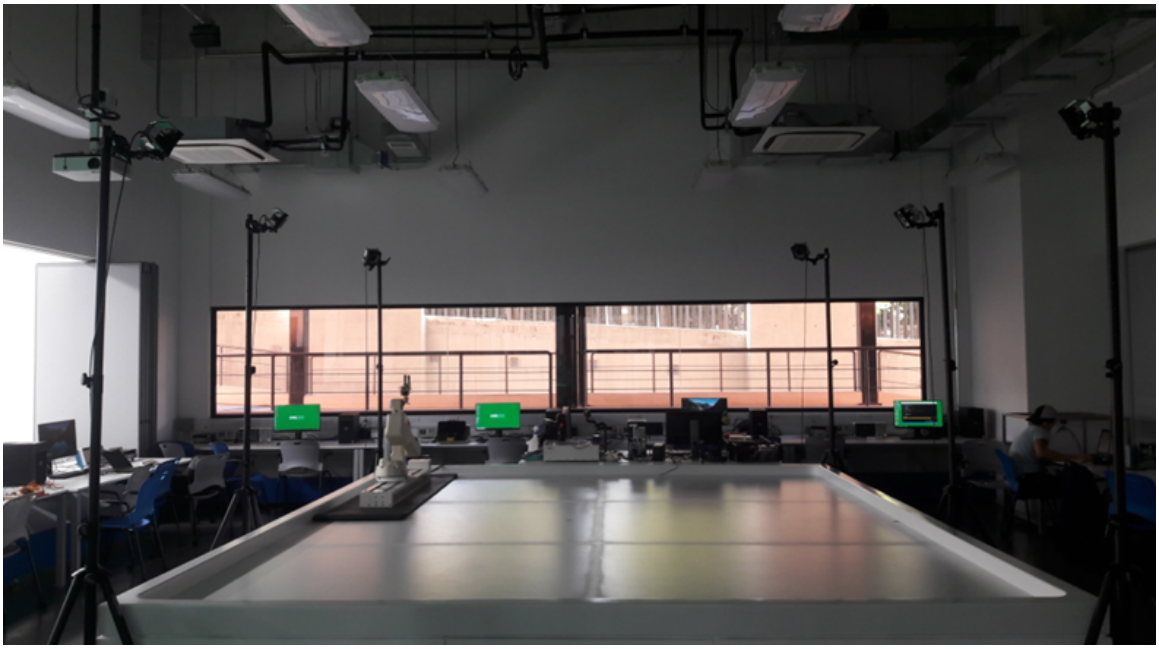


Figura 1: Vista frontal de la colocación de las cámaras en el ecosistema [2].

2.3. Implementación de infraestructura a escala para la evaluación de algoritmos por visión de computador para vehículos autónomos

El objetivo del trabajo de graduación [3] fue el desarrollo e implementación de la infraestructura necesaria para un vehículo autónomo a escala, que pudiera realizar pruebas de visión de computadora e interpretar su entorno de forma eficiente y accesible mediante un algoritmo optimizado. El vehículo consistió en un robot Pololu 3pi+ y la infraestructura vial tiene una escala 1:18.75, hecha mediante el corte láser con MDF e incluye rectas, curvas, señales de tránsito y semáforos. Se integró una cámara OpenMV Cam H7 con la visión de computadora para permitir la implementación del algoritmo de detección de carriles color amarillo, y alineación vertical para que el vehículo se direccione adecuadamente. Se desarrolló un entorno simulado que replica las condiciones reales para evaluar la efectividad de los algoritmos. Con tal de obtener un entrenamiento y optimización del modelo con aprendizaje automático, fue integrada la plataforma Edge Impulse que permitió entrenar el vehículo con un rendimiento de 25 FPS y ocupa 25 % de la memoria de OpenMV Cam H7. El modelo TensorFLow Lite fue capaz de la detección de señal de alto, señal de peatón y colores del semáforo con una validación máxima de 93.6 % y mínima de 89.3 %, por lo que detectó objetos con precisión de forma segura y eficiente para el vehículo; se muestra un ejemplo en la Figura 2. La OpenMV Cam H7 tuvo la limitante del espacio debido a que la memoria RAM es inferior a 400 KB, así que no se logró la creación de más clases o bibliotecas con imágenes adicionales, de forma que se limita la complejidad y escala de la infraestructura.

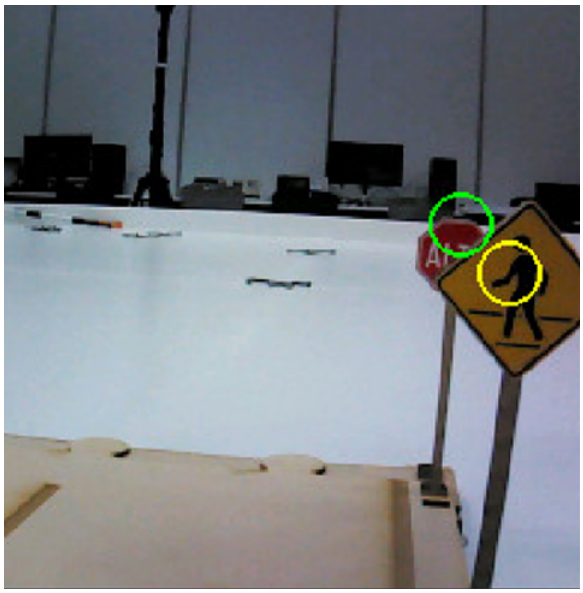


Figura 2: Señal de peatón positiva [3].

2.4. Pruebas a escala de algoritmos básicos de visión de computadora y control para vehículos autónomos a escala

El proyecto [4] consistió en el desarrollo y evaluación de algoritmos que validen la eficiencia y seguridad de la visión por computadora y control de movilidad de un vehículo autónomo a escala con un robot Pololu 3pi+ dentro del entorno del Robotat. Se probaron los algoritmos de control de velocidad longitudinal y dirección lateral *Pure Pursuit* y *Stanley* para desarrollar un entorno de simulación de las condiciones de tráfico, por medio de MATLAB para verificar el funcionamiento. Luego fueron implementados en los Pololu 3pi+ para validar el control por medio de trayectorias definidas. Asimismo se implementaron los algoritmos de control por medio del microcontrolador ESP32 y algoritmos de visión de computadora, mediante OpenMV Cam H7 para la identificación de señales de tránsito y el seguimiento de carriles, como se muestra en la 3. El vehículo fue capaz de seguir las trayectorias adecuadamente desde una perspectiva cualitativa, los algoritmos de detección de señales consiguieron detectarlas satisfactoriamente y el vehículo fue capaz de reaccionar según se esperaba. El algoritmo de control de trayectoria obtuvo mejores resultados con rutas circulares generadas y, de igual forma, el algoritmo de seguimiento de línea mejora su funcionamiento con velocidades mayores a 20 rad/s. Dentro de las limitantes se encuentran que el vehículo autónomo presentó problemas en trayectorias con cambios de dirección cerrada, dado a que no los permiten los algoritmos de control. Además la detección de señales en el Robotat cuando el Pololu 3pi+ se encuentra en movimiento, en ocasiones, presentó puntos ciegos donde las cámaras no son capaces de seguir al robot.

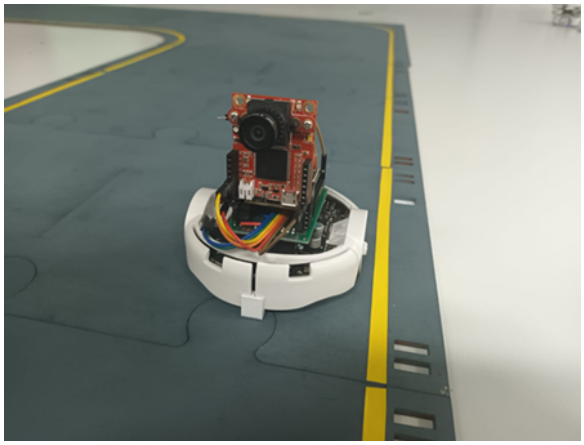


Figura 3: Pololu 3pi+ montado con la OpenMV Cam H7 para la detección de señales [4].

2.5. *Google Self-Driving Car Project*, conocido como Waymo

El proyecto [5] consistió en el desarrollo de tecnología necesaria para permitir a los vehículos navegar y operar de forma segura y autónoma en entornos urbanos y suburbanos complejos, sin intervención humana. El problema por resolver trató de cómo permitir que los vehículos autónomos tomen decisiones de conducción seguras y eficientes al comprender y navegar en entornos dinámicos e, incluso, impredecibles. La implementación de hardware y software permite crear mapas detallados del entorno, detección de señales de tráfico y vehículos y tomar decisiones de conducción. Esto incluyó también sensores *LIDAR*, cámaras y algoritmos de aprendizaje automático. El desarrollo e implementación fue primero en prueba de simulación, pista cerrada con obstáculos y finalmente en entorno reales. Dentro de los resultados importantes se destacaron la demostración exitosa del manejo autónomo durante muchas millas de conducción en distintas condiciones urbanas y suburbanas, tal y como intersecciones complejas, reconocimiento de señales y tráfico seguro de peatones y ciclistas. El alcance incluyó investigación, desarrollo tecnológico y pruebas piloto en varias ciudades. Sin embargo enfrentó limitaciones como la necesidad de una legislación clara para la operación de vehículos autónomos, la variabilidad en las condiciones de conducción que requieren adaptación continua de la tecnología y la aceptación pública de vehículos completamente autónomos.

2.6. *Bridging the Domain Gap between Synthetic and Real-World Data for Autonomous Driving*

El principal objetivo de [6] fue mejorar la eficiencia de los sistemas de conducción autónoma, mediante la adaptación correcta de los datos generados en simulación del entorno real. El problema para resolver trató sobre que los simuladores existentes presentan diferencias significativas en la fidelidad y realismo en comparación con los datos reales, lo cual limita la eficacia de los modelos de aprendizaje automático entrenados exclusivamente en entornos simulados. Por parte de la metodología, se incluyó el desarrollo y evaluación de técnicas de aprendizaje autónomo para que el vehículo se adapte a dominios de simulación y a los reales,

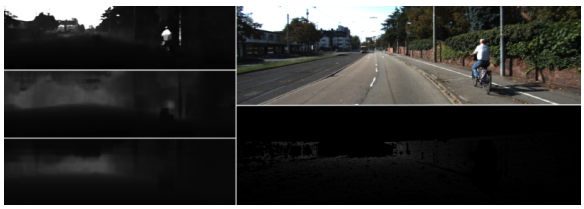


Figura 4: Mapa de profundidad [6].

en los cuales no siempre todas las situaciones las habrá procesado durante el entrenamiento. En la Figura 4 se muestra un ejemplo de mapa de profundidad para la interpretación del entorno. Los resultados mostraron mejoras significativas en la capacidad de los modelos de aprendizaje profundo para interpretar y responder adecuadamente sobre datos del mundo real, después de haber sido entrenados inicialmente con datos simulados. Por tanto, resulta posible reducir la brecha entre simulación y realidad para que así se aumente la utilidad de los simuladores como herramientas de entrenamiento. El estudio permitió probar y mejorar la precisión y eficacia de los sistemas autónomos en entornos reales, mediante la utilización de datos simulados en su entrenamiento. Sin embargo fueron reconocidas limitaciones como la falta de mayor investigación para explorar estrategias, y la validación en una gama más amplia de condiciones de conducción.

2.7. *Urban traffic signal control with connected and automated vehicles: A survey*

El objetivo principal de [7] fue estudiar el estado actual del control de señales de tráfico urbanas con vehículos conectados y automatizados (VCA), destacando las extensiones en los objetivos de control de tráfico, desde la eficiencia hasta la seguridad, economía de energía y reducción de la contaminación. Se estudiaron e integraron el diseño y restricciones para diferentes objetivos, considerando la seguridad, movilidad y objetivos de sostenibilidad para vehículos, ciclistas y peatones. La metodología incluyó una revisión acerca del control de señales de tráfico en el contexto de vehículos conectados y automatizados; examinó enfoques predecibles e impredecibles para el modelado del tráfico y la optimización de señales. Fueron integrados principios de tráfico y enfoques de análisis de datos que mejoraron la precisión y robustez de los métodos de control de tráfico. Los resultados más importantes consistieron en la identificación de tendencias emergentes en el control de tráfico con VCA, cambio de modelos de tráfico continuo a la modelización del movimiento de vehículos individuales, y la optimización de la fase y temporización de las señales para minimizar el tiempo total de viaje. El enfoque propuesto mejoró la eficiencia del tráfico en un 19.7 % y la economía de combustible un 23.7 % mediante la optimización de la trayectoria de cada vehículo. El alcance abarcó teoría y práctica en el control de señales de tráfico con VCA, destacando la necesidad de pruebas a gran escala y la consideración de la protección de la privacidad en la recopilación y uso de datos. Las limitaciones fueron la dependencia de datos de trayectoria para la mayoría de los métodos revisados, y los desafíos asociados con la implementación real debido a errores de observación y variabilidad de las condiciones de tráfico.

2.8. *Autonomous mobile robot development based on duckietown platform for recognizing and following the traffic sign*

El trabajo [6] trató sobre el desarrollo de un robot móvil autónomo capaz de reconocer y responder a señales de tráfico. Su objetivo principal fue la toma de decisiones en tiempo real del vehículo en un entorno controlado, pero significativo, para seguir las indicaciones de tránsito y actuar según las señales de forma adecuada mediante procesamiento de imágenes, navegación y aprendizaje autónomos. La plataforma Duckietown es un estudio sobre la autonomía de vehículos a nivel de investigación y educación, y la metodología implementada involucró el desarrollo de hardware y software. El sistema robótico se compone de una cámara ojo de pez y un controlador Raspberry Pi 3 para el procesamiento de imágenes y el control de movimiento. Los algoritmos de reconocimiento y detección de señales involucran el procesamiento de imágenes para la identificación de colores y formas. La calibración aseguró movimientos precisos y el programa *ROS* (*Robot Operating System*) gestiona el software en el entorno. La combinación de visión por computadora y control robótico permitieron al robot navegar según la interpretación de su entorno, simulando condiciones reales de conducción. Los resultados mostraron un desempeño correcto en la detección y reconocimiento de señales de tráfico, alcanzando hasta 80 % en condiciones ideales. Dichos resultados demostraron la efectividad del vehículo autónomo y presentaron un gran potencial, enfocando los desafíos para mejorar robustez y precisión del sistema. Sin embargo el reconocimiento disminuyó con las interferencias como similitud de colores, distintas intensidades de luz, objetos similares y otros no conocidos por el software, siendo parte de las limitaciones actuales. El alcance del proyecto fue la demostración de la capacidad del robot móvil autónomo para reconocer y seguir señales de tráfico en un entorno controlado, representando un avance significativo hacia la autonomía vehicular.

2.9. *The MiniCity: A 1/10th Scale Evaluation Platform for Testing Autonomous Urban Perception and Planning*

La implementación de vehículos autónomos presenta la necesidad de evaluar y probar el hardware y el software completos en escenarios similares al tráfico real, especialmente para tareas de percepción del entorno como detección de objetos y localización. Por tanto, debe realizarse la evaluación de algoritmos de percepción de forma aislada y en el contexto de cómo afectan a otras tareas críticas como la evasión de obstáculos. MiniCity [8] estudió superar estos desafíos al permitir la evaluación de algoritmos de percepción en un entorno urbano escalado y con múltiples carros, proporcionando un entorno seguro y controlado que simula situaciones de conducción urbanas complejas sin los riesgos asociados con las pruebas en el mundo real. Esta fue una plataforma de evaluación a escala 1/10 que incorpora casas, carreteras, semáforos, infraestructura de tráfico y vehículos autónomos, que permitió la simulación de escenarios de tráfico interactivos y la evaluación de software de percepción y hardware en condiciones que imitan al mundo real. La metodología trató sobre el uso de materiales físicos para crear un entorno realista y el empleo de tecnología de captura de movimiento, para proporcionar la localización y simulación de señales GPS; la Figura

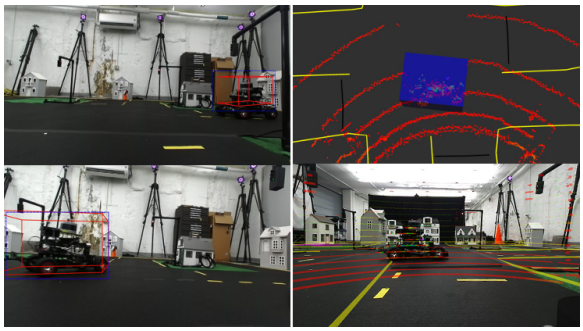


Figura 5: Detección de objetos [8].

5 muestra la detección de objetos en este ecosistema. Asimismo, se facilitaron los mapas detallados de la MiniCity para la navegación autónoma y se describió el uso de semáforos y casas para pruebas de percepción. Los vehículos autónomos, basados en la plataforma RACECAR, están equipados con sensores y controladores para ejecutar algoritmos autónomos. Los resultados destacados de las pruebas fueron la evaluación de algoritmos de detección de objetos mediante el uso de cámaras estéreo y *LIDAR*, demostrando la implementación de algoritmos para evitar colisiones en situaciones similares a la realidad. La plataforma compara el rendimiento de diferentes configuraciones de sensores y su impacto en la evasión de obstáculos y el seguimiento de carriles. Se presentaron métricas específicas como la precisión y el sensibilidad de la detección de objetos, así como la sensibilidad y especificidad de los sistemas de evitación de colisiones. A pesar de su realismo y utilidad, fue una aproximación a escala que puede no capturar completamente todos los aspectos y desafíos en el mundo real, especialmente las condiciones climáticas variables, la interacción con peatones y otros elementos dinámicos. Aún así, MiniCity es un paso significativo para el desarrollo y evaluación de tecnologías de vehículos autónomos, permitiendo un progreso más rápido y seguro de la implementación práctica.

2.10. *Rolling Horizon Approach for Real-time Charging and Routing of Autonomous Electric Vehicles*

El objetivo principal de [9] consistió en la optimización del tiempo real del enrutamiento y la carga de vehículos eléctricos autónomos (VEA), para descarbonizar el sector del transporte mientras se eliminan los errores humanos en los accidentes de tráfico. Con respecto a la metodología, se propone un enfoque que integra decisiones de enrutamiento y programación de carga para los vehículos eléctricos autónomos. Se trabajó para mejorar la eficiencia operativa de los vehículos mientras se asegura que se satisfacen sus necesidades de energía para completar las rutas previstas. Dentro de los resultados más importantes destacaron la eficacia del enfoque propuesto porque gestiona adecuadamente la carga y el enrutamiento de VEA en escenarios dinámicos, mejorando significativamente la operatividad de los vehículos y contribuyendo a la sostenibilidad del sector. El estudio demostró el potencial de los VEA para mejorar la sostenibilidad y seguridad del transporte. Sin embargo, las limitaciones resaltaron la necesidad de infraestructura de tecnología avanzada y la adaptación del modelo a diversas condiciones urbanas y de tráfico.

CAPÍTULO 3

Justificación

La investigación y desarrollo de vehículos autónomos han alcanzado avances significativos globalmente; sin embargo, estos progresos a menudo consideran entornos y condiciones ideales que no reflejan las complejidades de las infraestructuras viales y los patrones de tráfico en regiones como Latinoamérica, específicamente en la Ciudad de Guatemala. La necesidad de adaptar estas tecnologías a las condiciones locales representa un desafío que este proyecto busca abordar. El enfoque se centra en el desarrollo y la validación de tecnologías que pueden funcionar efectivamente bajo la variabilidad y las particularidades de las carreteras y señalizaciones en la Ciudad de Guatemala, un área que hasta ahora ha sido escasamente explorada en la investigación académica.

Este proyecto surge como una extensión y profundización de los trabajos previos realizados en la Universidad del Valle de Guatemala por ingenieros mecatrónicos como Fong y Arribas, quienes iniciaron la validación de algoritmos de control y visión por computadora en escalas reducidas. Al expandir estas investigaciones, se pretende cerrar la brecha entre los modelos teóricos y las aplicaciones prácticas, proporcionando una base sólida para futuras implementaciones en entornos reales.

Dicho enfoque es vital para garantizar la seguridad y eficiencia de los vehículos autónomos en el contexto guatemalteco, donde las condiciones de tráfico y la infraestructura presentan retos únicos que los modelos estándares no logran abordar adecuadamente. La investigación se enfoca en adaptar y probar algoritmos que se ajusten a estas particularidades, sin presuponer la homogeneidad de condiciones encontradas en los entornos más controlados y estandarizados de los estudios previos.

Este proyecto justifica su relevancia al contribuir directamente a la reducción de las barreras técnicas y conceptuales para la implementación de vehículos autónomos en la Ciudad de Guatemala. Al proporcionar datos y modelos validados en entornos locales, se espera establecer un precedente para la tecnología de conducción autónoma, facilitando así su aceptación y adopción en un espectro más amplio de contextos urbanos y rurales.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Desarrollar un sistema de simulación y planificación de rutas para vehículos autónomos a escala que sea capaz de replicar y adaptarse a las condiciones viales específicas de la Ciudad de Guatemala, tanto en aspectos de software como de hardware.

4.2. Objetivos específicos

- Digitalizar una carretera a escala del entorno vial de la Ciudad de Guatemala para su uso en simulaciones de navegación autónoma.
- Adaptar algoritmos de búsqueda de rutas para optimizar la distancia recorrida en un entorno que responda a las señalizaciones viales en el contexto de la Ciudad de Guatemala.
- Implementar y comparar propuestas de sistemas de control para robots diferenciales que respondan a las señales de tránsito e interacción con otros vehículos.
- Establecer un método de validación utilizando el sistema OptiTrack dentro del ecosistema Robotat, para monitorear y evaluar continuamente el comportamiento del vehículo autónomo durante las pruebas.
- Incorporar múltiples vehículos en el entorno simulado para emular condiciones de tráfico realistas y evaluar la interacción entre vehículos autónomos.

CAPÍTULO 5

Alcance

El presente trabajo se centra en el desarrollo de un sistema de simulación y planificación de rutas para vehículos autónomos a escala, adaptado a las condiciones viales de la Ciudad de Guatemala, incluyendo la implementación de algoritmos de control y su evaluación en entornos simulados.

Sin embargo, el estudio no incluye la implementación de vehículos autónomos en tamaño real ni pruebas en entornos reales de tráfico en la Ciudad de Guatemala. Tampoco se abordó el análisis del tráfico complejo con múltiples factores dinámicos, ni la inteligencia autónoma avanzada, ya que los vehículos siguen velocidades predefinidas sin interpretar de forma autónoma el entorno o ejecutar control propio. El control y la toma de decisiones del vehículo dependen de la computadora central, lo que limita la capacidad del robot para procesar su entorno y reducir tiempos de respuesta. No se consideraron mejoras en la velocidad de comunicación del ecosistema Robotat para optimizar el rendimiento en tiempo real. Asimismo, aspectos como la influencia de condiciones climáticas adversas o la implementación de tecnologías avanzadas como *LIDAR* y cámaras estéreo para mejorar la navegación y detección de obstáculos tampoco fueron contemplados. Estos elementos se identifican como áreas a desarrollar en futuras investigaciones.

CAPÍTULO 6

Marco teórico

El marco teórico de este estudio abarca diversas disciplinas y tecnologías relacionadas con la simulación y planificación de rutas para vehículos autónomos. Estos vehículos utilizan algoritmos de control predictivo y tecnologías de visión por computadora, lo que les permite operar sin intervención humana; sin embargo, enfrentan desafíos legales y éticos.

La generación de rutas en aplicaciones como Google Maps implica la recuperación de datos geoespaciales y de tráfico, y la construcción de grafos viales que modelan la red urbana. Algoritmos de búsqueda y optimización de rutas, como Dijkstra y A*, son esenciales para encontrar trayectorias óptimas en tiempo real. La planificación de movimiento, mediante métodos de búsqueda en grafos y de muestreo, asegura la navegación segura y eficiente en entornos complejos.

El modelado cinemático de robots diferenciales proporciona una base matemática para describir la trayectoria del vehículo, facilitando el diseño de controladores que gestionan velocidad y orientación. Controladores basados en el método de Lyapunov y en control PID con enfoque exponencial garantizan la estabilidad y precisión en el seguimiento de trayectorias. La implementación y evaluación de estos controladores se realiza utilizando el robot Pololu 3pi+ 32U4 OLED. El sistema de detección y captura de movimiento con la cámara OptiTrack Primex 41 proporciona datos precisos en tiempo real y optimiza la operación del vehículo. La clasificación de señales de tráfico y sus acciones específicas son esenciales para la interacción segura del vehículo autónomo con su entorno urbano.

Estos temas se interrelacionan para desarrollar un sistema que se adapta a las condiciones viales específicas de la Ciudad de Guatemala, mejorando la eficiencia y seguridad del transporte urbano autónomo.

6.1. Vehículos autónomos

Los vehículos autónomos utilizan algoritmos avanzados de control predictivo basados en modelos para optimizar la planificación y seguimiento de trayectorias, gestionando dinámicamente su posición y navegación. Poseen sistemas de control robustos y adaptativos que manejan incertidumbres y perturbaciones, asegurando un rendimiento óptimo bajo diversas condiciones operativas [10].

En cuanto a la navegación, dependen de tecnologías de visión por computadora, incluyendo geometría multi-vista y estimación de movimiento en tiempo real. Estos sistemas son fundamentales para la localización y mapeo simultáneos, proporcionando la percepción del entorno necesaria para la navegación autónoma [11]. La interacción segura con otros usuarios de la vía se mejora mediante métodos basados en teoría de juegos y procesos de decisión de Markov parcialmente observables, permitiendo una toma de decisiones estratégica y táctica en entornos compartidos con vehículos conducidos por humanos [12].

Según la Sociedad de Ingenieros Automotrices (SAE), existen cinco niveles de conducción autónoma, desde la ausencia de automatización (Nivel 0) hasta la automatización completa (Nivel 5). Los niveles intermedios describen grados crecientes de asistencia y automatización, culminando en vehículos que pueden operar en cualquier entorno sin necesidad de un conductor humano.

Los vehículos autónomos presentan desafíos significativos en los ámbitos legal y ético. Las cuestiones de responsabilidad en accidentes implican complejas consideraciones sobre defectos de producto y la interacción entre normativas legales y avances tecnológicos. Esto ha llevado a propuestas como la responsabilidad empresarial del fabricante, con sistemas de compensación financiados por los productores y administrados a través de un fondo [13]. Además, la integración de estos vehículos en la infraestructura de tráfico actual y su interacción con las leyes de tránsito vigentes son áreas de investigación activa.

Estos vehículos ofrecen potenciales beneficios como mayor eficiencia en el transporte, reducción de accidentes y mejor accesibilidad. Sin embargo, sus limitaciones actuales incluyen la necesidad de mejoras en la detección de obstáculos a larga distancia y alta velocidad, y la adaptación a las dinámicas complejas e impredecibles del tráfico humano [14].

6.2. Generación de rutas en Google Maps

La generación de rutas en aplicaciones de navegación como Google Maps se basa en la construcción de grafos que modelan la red vial utilizando datos geoespaciales y de tráfico [15]. Los nodos representan intersecciones o puntos clave, y las aristas corresponden a los segmentos de carretera que conectan estos nodos. A cada arista se le asigna un peso que refleja métricas como la distancia física, el tiempo de viaje estimado y las condiciones actuales del tráfico, permitiendo así rutas más eficientes y actualizadas.

Para manejar la complejidad y dinámica de la red vial, se emplean algoritmos de búsqueda y optimización de rutas como Dijkstra y A*. El algoritmo de Dijkstra encuentra la ruta más corta en grafos con pesos no negativos, mientras que el algoritmo A* mejora la

eficiencia al utilizar una heurística que estima el costo restante hasta el destino, dirigiendo la búsqueda de manera más efectiva [16]. Mejoras en las heurísticas del A*, como la de *k-step look-ahead*, reducen significativamente el tiempo de ejecución y el número de nodos explorados en redes viales grandes y complejas.

Además, las redes neuronales gráficas (GNN) han emergido como herramientas potentes para predecir tiempos estimados de llegada (ETA) y mejorar la precisión de las rutas en tiempo real [17]. Las GNN capturan eficazmente la topología de la red vial y las condiciones dinámicas del tráfico, ofreciendo predicciones más precisas que los métodos tradicionales y permitiendo una integración más efectiva de datos de tráfico en tiempo real.

6.3. Métodos de planificación de movimiento

La planificación de movimiento es esencial para que los vehículos autónomos naveguen de manera eficiente y segura desde un punto inicial hasta un punto final, considerando obstáculos y restricciones del entorno. Existen diferentes enfoques para lograrlo, entre ellos los métodos de búsqueda en grafos y los métodos de muestreo.

6.3.1. Métodos de búsqueda en grafos

Estos métodos utilizan estructuras matemáticas llamadas grafos, compuestas por nodos (vértices) y aristas (enlaces) que representan intersecciones y caminos. La teoría de grafos estudia estas estructuras y sus aplicaciones en sistemas de navegación y optimización de rutas [18]. El problema de la ruta más corta, que busca el camino más breve entre dos nodos, es fundamental para minimizar costos y tiempos de transporte [19].

Algoritmo de Dijkstra

El algoritmo de Dijkstra es eficiente para resolver el problema de la ruta más corta en grafos con pesos no negativos. Utiliza la técnica de relajación de aristas para determinar iterativamente las distancias más cortas desde un nodo origen a todos los demás nodos [20]. Su procedimiento es el siguiente:

1. Inicializar la distancia del nodo origen a cero y las distancias a los demás nodos a infinito.
2. Crear un conjunto de nodos no visitados.
3. Seleccionar el nodo no visitado con la distancia más corta.
4. Para el nodo seleccionado, actualizar las distancias tentativas de sus vecinos no visitados si se encuentra un camino más corto.
5. Marcar el nodo seleccionado como visitado y removerlo del conjunto de no visitados.

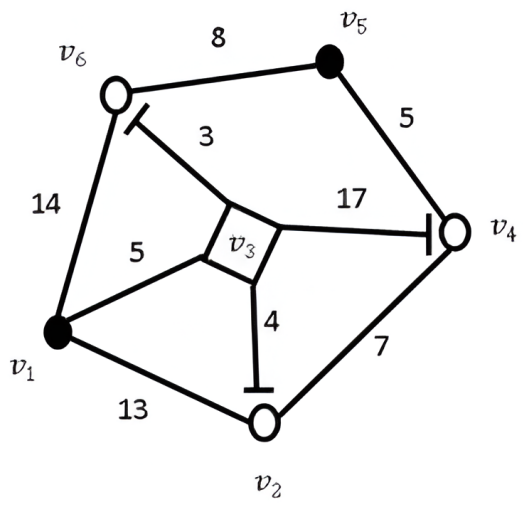


Figura 6: Ejemplo de semigrafo ponderado [19].

6. Repetir hasta que todos los nodos hayan sido visitados o no se puedan actualizar más distancias.

La Figura 6 muestra un ejemplo de semigrafo ponderado.

El algoritmo de Dijkstra es relevante por su capacidad para manejar grafos de gran tamaño con robustez y precisión, proporcionando una base sólida para resolver problemas de optimización de rutas [18], [20].

Variantes de Dijkstra: A* y D*

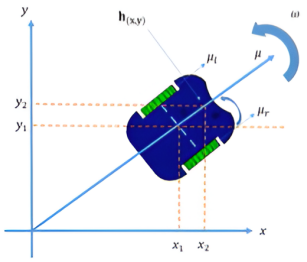
El algoritmo A* (A estrella) extiende el algoritmo de Dijkstra introduciendo una heurística que estima el costo desde un nodo hasta el destino, guiando la búsqueda de manera más eficiente y reduciendo el número de nodos explorados [21]. Es especialmente útil en aplicaciones donde el tiempo de respuesta es crítico, como en la navegación en tiempo real de vehículos autónomos.

El algoritmo D* (Dynamic A*) es una variante que permite la replanificación dinámica, ajustando la ruta en tiempo real ante cambios en el entorno, como la aparición de nuevos obstáculos. Esta capacidad es esencial en entornos dinámicos y no estructurados, donde las condiciones pueden cambiar rápidamente [22].

6.3.2. Métodos de muestreo

Los métodos de muestreo generan y evalúan muestras del espacio de búsqueda sin depender de una estructura de grafo explícita. Son especialmente útiles en entornos de alta dimensión o cuando no se dispone de una descripción completa del entorno.

$$\dot{\mathbf{h}}(t) = \mathbf{J}(\mathbf{q}(t))\dot{\mathbf{q}}(t)$$



$$\dot{h}_x = \mu \cos\varphi - a\omega \sin\varphi$$

$$\dot{h}_y = \mu \sin\varphi + a\omega \cos\varphi$$

$$\underbrace{\begin{bmatrix}\dot{h}_x \\ \dot{h}_y\end{bmatrix}}_{\dot{\mathbf{h}}} = \underbrace{\begin{bmatrix}\cos\varphi & -a\sin\varphi \\ \sin\varphi & a\cos\varphi\end{bmatrix}}_{\mathbf{J}} \underbrace{\begin{bmatrix}\mu \\ \omega\end{bmatrix}}_{\dot{\mathbf{q}}}$$

Figura 7: Cinemática diferencial [25].

***Rapidly-Exploring Random Trees* (RRT)**

El algoritmo RRT explora rápidamente el espacio de configuraciones mediante la generación aleatoria de puntos y la construcción de un árbol que conecta estas configuraciones. Cada nuevo punto se conecta al punto más cercano en el árbol existente, repitiendo el proceso hasta alcanzar el objetivo o cumplir un criterio de terminación. Es efectivo para encontrar caminos en entornos complejos y de alta dimensión debido a su capacidad de explorar grandes áreas eficientemente [23].

***Probabilistic Roadmaps* (PRM)**

El algoritmo PRM consta de dos fases: construcción y consulta. Durante la fase de construcción, se generan aleatoriamente configuraciones libres de colisiones y se conectan para formar un grafo de posibles caminos. En la fase de consulta, se busca la ruta más corta en este grafo desde el inicio hasta el objetivo. La eficacia de PRM radica en su capacidad para preprocesar el espacio de búsqueda y reutilizar el grafo generado para múltiples consultas [24].

6.4. Modelado cinemático de un robot diferencial

La cinemática de vehículos en el contexto de robots diferenciales constituye una parte fundamental en el estudio de sistemas autónomos y robótica. El modelado cinemático permite describir matemáticamente la trayectoria de un robot en un espacio físico a partir de sus velocidades de actuación y considerando sus especificaciones. El detalle del modelo considerado se presenta en las figuras 7 y 8.

La cinemática diferencial basada en un uniciclo implica adaptar la forma en que se considera el movimiento y control del robot. Mientras que la cinemática diferencial describe el comportamiento del robot a partir de la velocidad de cada rueda independiente, como se discute en [26], el modelo del uniciclo simplifica la representación al considerar el robot como si tuviera una sola rueda que controla tanto la velocidad lineal como la angular. Este enfoque

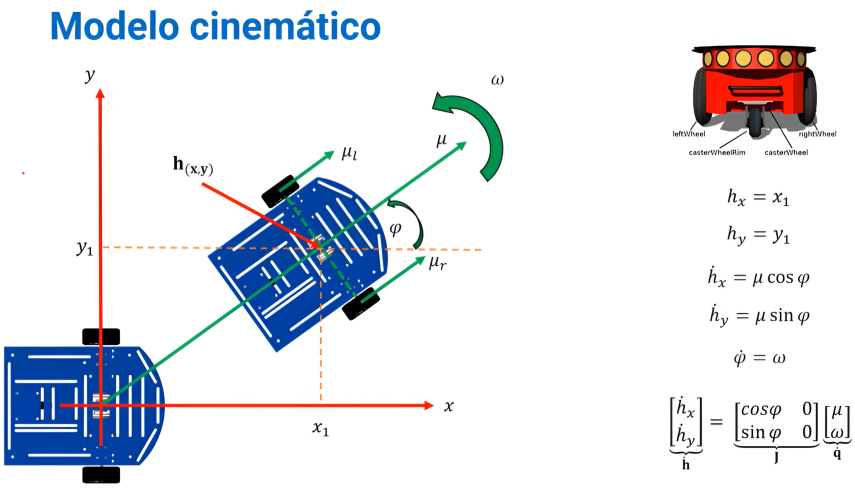


Figura 8: Modelo cinemático [25].

no solo es útil para simplificar el análisis matemático, sino también para la implementación de algoritmos de control más intuitivos.

Por tanto, se permite una integración más directa de la teoría del control y la planificación de trayectorias en robótica, facilitando el diseño de controladores que gestionen directamente la velocidad lineal y angular en lugar de manejar cada rueda de forma independiente. De esta manera, se puede obtener una alta precisión y la simplicidad del control.

6.4.1. Cinemática diferencial del robot

El robot diferencial es un modelo ampliamente utilizado debido a su estructura simple y eficaz control. Este se caracteriza por tener dos ruedas paralelas cada una propulsada de manera independiente y, en algunos casos, una o varias ruedas pivotantes para mantener el equilibrio. Su movimiento se describe a través de coordenadas en el plano (x, y) junto a una orientación angular θ, que se relaciona con el eje perpendicular al plano de movimiento.

Según [27], la relación entre las velocidades de las ruedas y la velocidad del robot se expresa mediante las ecuaciones de la cinemática diferencial:

$$\dot{x} = \frac{r}{2}(\omega_l + \omega_r)\cos(\theta)$$

$$\dot{y} = \frac{r}{2}(\omega_l + \omega_r)\sin(\theta)$$

$$\dot{\theta} = \frac{r}{L}(\omega_r - \omega_l)$$

donde r es el radio de las ruedas, ω_l y ω_r son las velocidades angulares de las ruedas izquierda y derecha, respectivamente, y L es la distancia entre las dos ruedas.

Estas ecuaciones permiten determinar la trayectoria del robot a partir de la velocidad de sus ruedas, que son controlables a través de actuadores.

6.4.2. Transformación de velocidades

La transformación de velocidades en un robot diferencial permite relacionar las velocidades de los actuadores (ruedas) con la velocidad del robot en el espacio de configuración, que incluye su posición en el plano y la orientación angular. Para esto se utiliza la matriz Jacobiana, dado que adapta estas velocidades del sistema de actuadores al sistema de coordenadas global del robot, tomando en cuenta peculiaridades específicas de su diseño como la ubicación del centro de control o de masa respecto a las ruedas motrices.

Considerando el punto de interés alejado del eje central entre las ruedas, estando desplazado a una distancia a, la matriz Jacobiana se modifica para reflejar este desplazamiento. Con dicha adaptación se obtiene una representación precisa del movimiento del robot, ya que el punto de interés puede influir en la dinámica del robot de manera significativa, especialmente en maniobras de giro y navegación en entornos complejos.

La matriz Jacobiana se expresa como:

$$J = \begin{bmatrix} \cos(\theta) & -a\sin(\theta) \\ \sin(\theta) & a\cos(\theta) \end{bmatrix}$$

Donde:

- θ representa la orientación angular del robot en el instante.
- a es la distancia desde el centro geométrico hasta el punto de interés o control sobre el robot.
- $\cos(\theta)$ y $\sin(\theta)$ ajustan las componentes de velocidad lineal a las coordenadas cartesianas.
- Los términos $-a\sin(\theta)$ y $a\cos(\theta)$ ajustan la contribución de la velocidad angular debido al brazo de momento creado por el desplazamiento a del eje de rotación.

El efecto de incluir el término a en la matriz Jacobiana es que cualquier rotación ω (velocidad angular) ahora contribuye no solo al cambio en la orientación del robot, sino también a su posición en el plano (x, y). Dicho enfoque permite modelar con mayor precisión el movimiento y giro del robot, considerando que el centro de control o masa no coincide exactamente con el punto medio entre las ruedas motrices. Esto es particularmente importante en robots con diseños complejos o en aquellos donde el centro de masa afecta significativamente las características de manejo y estabilidad.

6.5. Controladores para robots diferenciales

El diseño de controladores para robots diferenciales es crucial para garantizar su funcionamiento preciso y eficiente en entornos dinámicos y no lineales. Dos métodos destacados son el control basado en Lyapunov y el control PID con enfoque exponencial. El método basado

en Lyapunov asegura la estabilidad asintótica del sistema mediante funciones candidatas que disminuyen a lo largo de las trayectorias del sistema [25]. Por su parte, el control PID con enfoque exponencial mejora la respuesta y estabilidad del robot, ajustando dinámicamente su comportamiento ante errores de posición y orientación. Ambos métodos contribuyen significativamente a la precisión y eficiencia en la operación de robots móviles diferenciales.

6.5.1. Diseño de control Lyapunov

El método de Lyapunov, basado en la teoría de estabilidad propuesta por Aleksandr Lyapunov, permite asegurar la estabilidad de sistemas dinámicos sin necesidad de resolver explícitamente las ecuaciones diferenciales [25].

Fundamentos del método de Lyapunov

Este método se centra en la selección de una función candidata de Lyapunov $V(x)$, que debe ser definida positivamente, decrecer a lo largo de las trayectorias del sistema y converger a cero conforme el estado se aproxima al equilibrio. Es decir, $V(x) > 0$ para todo $x \neq 0$, $V(0) = 0$ y $\dot{V}(x) < 0$ [25]. Conceptualmente, $V(x)$ es comparable a una medida de energía que debe disminuir constantemente en un sistema estable.

Aplicación en control robótico

En robótica, el método de Lyapunov se utiliza para diseñar controladores que garantizan la estabilidad en entornos dinámicos y no lineales [28]. Utilizando la función de Lyapunov, se formulan leyes de control que reducen los errores de estado, como la diferencia entre la posición u orientación deseada y la actual. Por ejemplo, si se define el error $e(t)$ y una función $V(e) = \frac{1}{2}e^T e$, la ley de control se diseña para que:

$$\dot{V}(e) = e^T \dot{e} < 0.$$

Esta condición garantiza que el sistema evoluciona hacia el estado deseado, asegurando la convergencia y estabilidad del robot, permitiendo el posicionamiento preciso y seguimiento de trayectorias complejas.

Control de posición y orientación

El control de posición implica minimizar el error entre la posición actual del robot y un punto de referencia fijo. Con el error de posición $e = x_{\text{deseado}} - x_{\text{actual}}$ y la función de Lyapunov $V(e) = \frac{1}{2}e^T e$, la estabilidad asintótica se satisface si:

$$\dot{V}(e) = e^T \dot{e} < 0.$$

De manera similar, el control de orientación busca minimizar el error de orientación $\theta_e = \theta_{\text{deseado}} - \theta_{\text{actual}}$ utilizando la función $V(\theta_e) = \frac{1}{2}\theta_e^2$ y asegurando que:

$$\dot{V}(\theta_e) = \theta_e \dot{\theta}_e < 0.$$

Estos enfoques garantizan que el error converge a cero, logrando el posicionamiento y orientación precisos del robot.

Control de seguimiento de trayectorias

Para que un robot siga una trayectoria predefinida parametrizada en el tiempo, se define una función de Lyapunov $V(\mathbf{e}) = \frac{1}{2}\mathbf{e}^T\mathbf{e}$, donde $\mathbf{e} = \mathbf{h}_d - \mathbf{h}$ es el vector de error de trayectoria. La estabilidad se garantiza si:

$$\dot{V}(\mathbf{e}) = \mathbf{e}^T\dot{\mathbf{e}} < 0,$$

lo que asegura la convergencia asintótica del error y un seguimiento preciso de la trayectoria [29].

Estabilidad asintótica

La estabilidad asintótica se alcanza cuando el estado del sistema converge al equilibrio a medida que el tiempo tiende al infinito. Esto requiere que la derivada de la función de Lyapunov sea estrictamente negativa para todos los estados excepto en el equilibrio:

$$\dot{V}(x) < 0 \quad \text{para todo } x \neq 0.$$

Esta condición garantiza que el error no solo permanece cerca de cero, sino que converge activamente a cero, asegurando un sistema estable y asintóticamente estable [25].

Seguimiento de caminos

En el seguimiento de caminos, el robot sigue una serie de puntos interpolados en una ruta continua sin una dependencia temporal estricta. Se implementa un controlador que ajusta la trayectoria en tiempo real, basado en el error de posición $h_e = h_d - h$. La ley de control es:

$$\dot{q}_{\text{ref}} = J^{-1}(\dot{p}_d + Kh_e),$$

donde J^{-1} es la matriz jacobiana inversa, $\dot{p}_d$ es la derivada de la posición deseada y K es la matriz de ganancia del controlador. La estabilidad se verifica asegurando que:

$$\dot{V} = h_e^T \dot{h}_e < 0.$$

Esto implica seleccionar K adecuadamente para que el sistema sea estable bajo todas las condiciones operativas previstas [25].

6.5.2. Control PID de posición y orientación con enfoque exponencial

El control PID (Proporcional-Integral-Derivativo) se adapta para mejorar la estabilidad y rapidez de respuesta en sistemas de control automático. En robots móviles diferenciales, se introduce una modificación exponencial al control PID tradicional para mejorar la convergencia y respuesta en el seguimiento de trayectorias.

El controlador PID clásico se describe como:

$$v = k_p e_p + k_i \int e_p \, dt + k_d \frac{de_p}{dt},$$

donde v es la variable de control, e_p es el error de posición y k_p, k_i, k_d son las ganancias proporcional, integral y derivativa.

El controlador PID modificado incluye un término exponencial:

$$v = -k(e_p)e_p = v_0 \left(1 - e^{-\alpha e_p^2}\right) \frac{e_p}{|e_p|}.$$

Este enfoque ofrece una respuesta inicial más agresiva que se suaviza a medida que el error disminuye, permitiendo una aproximación más rápida al objetivo sin sacrificar la estabilidad [30].

El control de orientación se maneja con un controlador PID convencional:

$$\omega = k_{p_o} e_o + k_{i_o} \int e_o \, dt + k_{d_o} \frac{de_o}{dt},$$

donde e_o es el error de orientación.

Las ventajas prácticas del enfoque exponencial incluyen:

- **Respuesta mejorada**: Mayor dinamismo en presencia de grandes errores de posición, vital en navegación autónoma donde el tiempo de respuesta es crítico.
- **Estabilidad aumentada**: Evita oscilaciones excesivas alrededor del punto deseado, manteniendo la estabilidad [31].
- **Convergencia rápida**: Reduce el tiempo para alcanzar la posición deseada sin comprometer la precisión en el seguimiento de la trayectoria.

Este método es especialmente útil en el seguimiento de trayectorias, permitiendo al robot adaptarse a entornos dinámicos y realizar correcciones de curso más efectivas [32].

6.6. Robot Pololu 3pi+ 32U4 OLED

El robot Pololu 3pi+ 32U4 OLED (Figura 9) es un robot móvil educativo diseñado por Pololu. La versión "*Standard Edition*", ensamblada con motores *Micro Metal Gearmotors* 6V de 30:1 MP, ofrece una combinación óptima de velocidad y control, proporcionando una plataforma robusta y versátil para educación y experimentación en robótica.

6.6.1. Especificaciones técnicas

En [33], [34]

- Microcontrolador: Utiliza el ATmega32U4, conocido por su compatibilidad con Arduino[1]. Este controlador facilita la programación del robot a través del entorno Arduino y proporciona soporte nativo para comunicación USB, eliminando la necesidad de un adaptador serial separado.
- Sistema de visualización: Incluye una pantalla OLED que permite mostrar información como el estado de los sensores, valores de depuración o mensajes personalizados.
- Sensores: Está equipado con una variedad de sensores, incluyendo un arreglo de sensores de línea para detección y seguimiento de líneas, esencial para competencias de robótica y actividades educativas relacionadas con automatización y navegación.
- Motores: Los *Micro Metal Gearmotors* 6V de 30:1 MP proporcionan un equilibrio entre velocidad y torque, permitiendo que el robot maneje diferentes tipos de superficies y obstáculos mientras mantiene una respuesta ágil. La velocidad máxima de operación es de 1.5 m/s.
- Conectividad y expansión: Incluye varios puertos y conectores de expansión, permitiendo añadir módulos adicionales como sensores extra, módulos de comunicación inalámbrica o actuadores adicionales, ampliando las posibilidades de proyectos y experimentos.
- Alimentación: Diseñado para funcionar con cuatro baterías AAA, ya sean alcalinas o recargables NiMH. Se recomienda el uso de baterías recargables NiMH, que proporcionan un voltaje nominal de 4.8 V (1.2 V por celda). El sistema de alimentación incluye un regulador que mantiene una salida constante de 8 V para los motores, asegurando que su velocidad no varíe con el nivel de carga de las baterías.

[1] Arduino es una plataforma de hardware y software de código abierto diseñada para crear proyectos de electrónica interactiva. Permite programar microcontroladores de manera accesible para crear prototipos y productos.

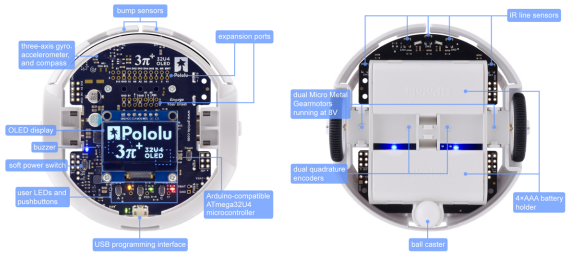


Figura 9: Robot móvil con ruedas Pololu 3pi+ [33].

El Pololu 3pi+ 32U4 OLED es una herramienta para la enseñanza de principios de robótica y programación, y también sirve como plataforma para introducir conceptos avanzados de control automático, sensores y sistemas embebidos. Su diseño modular y expandible lo hace ideal para adaptarse a diversos niveles educativos y necesidades experimentales.

6.7. Sistema de detección y captura de movimiento

Un sistema de detección y captura de movimiento utiliza cámaras y sensores para detectar y registrar los movimientos tridimensionales de objetos o individuos. Esta tecnología se emplea en aplicaciones como animación y efectos visuales en la industria del entretenimiento, y en aplicaciones avanzadas en robótica y medicina. En robótica, los sistemas de captura de movimiento son esenciales para desarrollar modelos precisos de movimiento mecánico y mejorar la interacción entre máquinas y entornos complejos [35].

La integración de un sistema de captura de movimiento en el control de vehículos autónomos permite implementar un control de lazo cerrado más eficiente. El sistema proporciona retroalimentación continua y precisa sobre la posición y orientación del vehículo. Estos datos ajustan dinámicamente las acciones del vehículo en respuesta a condiciones variables del entorno, resultando en un control más preciso y seguro [36].

6.7.1. OptiTrack PrimeX 41

El OptiTrack PrimeX 41, presentado en la Figura 10, es una cámara de captura de movimiento que destaca por su capacidad para ofrecer datos de seguimiento de movimiento en tiempo real con alta precisión. Funciona mediante la emisión de luz infrarroja, que es reflejada por marcadores especiales colocados en el objeto de interés. La cámara captura esta luz reflejada y convierte los datos en coordenadas tridimensionales que describen el movimiento del objeto con gran exactitud [37].

Las características técnicas de la cámara incluyen una resolución de 2048 x 1088 con una tasa de captura de hasta 360 fotogramas por segundo, ideal para aplicaciones que requieren alta granularidad y velocidad de respuesta. Además, cuenta con un amplio rango de visión y es capaz de reducir significativamente la distorsión en los bordes de su campo de visión, mejorando la precisión de la captura en áreas grandes [37].

Figura 10: Cámara OptiTrack PrimeX 41 [37].

Estas capacidades técnicas hacen que el OptiTrack PrimeX 41 sea extremadamente útil en entornos donde se requiere precisión en la captura de movimiento, como el laboratorio de robótica "Robotat" de la Universidad del Valle de Guatemala. La precisión y velocidad de estas cámaras son fundamentales para implementar sistemas de control avanzados en vehículos autónomos a escala, permitiendo ajustes en tiempo real que optimizan la operación del vehículo en su entorno.

6.8. Clasificación de señales de tráfico

Las señales de tráfico en Guatemala [38], mostradas en la Figura 11, se clasifican en varias categorías según su función y el comportamiento que dictan a conductores y peatones:

- Señales de regulación: Incluyen señales de alto, ceda el paso y límites de velocidad. Requieren acciones obligatorias, como respetar límites de velocidad y detenerse, asegurando el derecho de paso a otros usuarios de la vía.
- Señales de prevención: Alertan sobre posibles peligros o cambios en las condiciones del camino, como curvas, intersecciones o zonas escolares. Obligan al conductor a reducir la velocidad y proceder con cautela.
- Señales de información: Proporcionan datos útiles para la navegación, como indicaciones de servicios o direcciones, por ejemplo, la presencia de hospitales, gasolineras o paradas de autobús. Ayudan a planificar acciones futuras, como detenerse o cambiar de ruta.
- Señales temporales: Utilizadas en zonas de construcción o cuando las condiciones de la carretera están alteradas temporalmente. Prevalecen sobre las permanentes y requieren atención y cumplimiento inmediato para garantizar la seguridad, incluyendo desvíos, reducción de carriles o presencia de maquinaria pesada.

Estas categorías se fundamentan en las regulaciones estipuladas en la "Ley y Reglamento de Tránsito" de Guatemala, asegurando que conductores y peatones entiendan sus obligaciones mientras transitan por las vías del país.

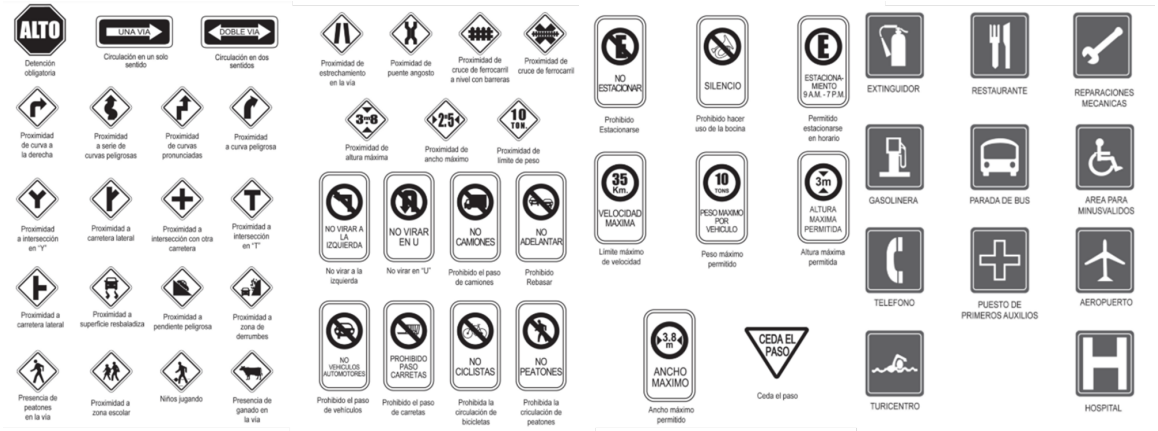


Figura 11: Señales de tránsito Guatemala [38].

6.9. Infraestructura vial de Guatemala: desafíos y comparaciones

La infraestructura vial en Guatemala presenta desafíos que afectan la movilidad y conectividad del país. Según [39], a pesar de ser la mayor economía de Centroamérica, Guatemala ocupa el puesto 134 de 141 países en conectividad vial, y su infraestructura es percibida negativamente, situándose en el puesto 84 de 137 países. Esto se debe en gran parte a la baja densidad de carreteras y al mantenimiento inadecuado, resultando en un sistema vial defectuoso que sufre deslizamientos de tierra y problemas relacionados con condiciones climáticas adversas.

Con solo 0.13 km de carreteras por kilómetro cuadrado, el país enfrenta retos para mejorar su conectividad y movilidad [40]. Esta baja densidad dificulta el transporte de bienes y personas, afectando negativamente la economía y la calidad de vida de los ciudadanos.

En contraste, países desarrollados cuentan con infraestructuras viales extensas, bien mantenidas y equipadas con tecnología avanzada para gestionar el tráfico y mejorar la seguridad. Por ejemplo, Alemania y Japón tienen sistemas de carreteras de alta densidad y calidad, con redes de autopistas bien conectadas y mantenidas regularmente, lo que asegura movilidad eficiente y segura. Además, invierten significativamente en infraestructura vial, reflejándose en mayor durabilidad y eficiencia de sus sistemas de transporte.

La señalización vial en Guatemala también enfrenta problemas [41]. La falta de señales adecuadas y el deterioro de las existentes complican la navegación y aumentan los riesgos de accidentes. En países de alto desarrollo, la señalización es clara, visible y mantenida consistentemente, contribuyendo a una conducción más segura y eficiente. En Guatemala, la presencia de carreteras de terracería y de un solo carril, especialmente en áreas rurales, dificulta el transporte y aumenta el tiempo de viaje. Además, el tráfico en zonas urbanas, particularmente en la Ciudad de Guatemala, es un problema persistente debido a la falta de infraestructura adecuada y a la gestión ineficiente del tráfico.

Es necesario abordar estos desafíos para mejorar la conectividad, movilidad y seguridad vial en el país. La Ciudad de Guatemala, como centro urbano más grande y congestiona-

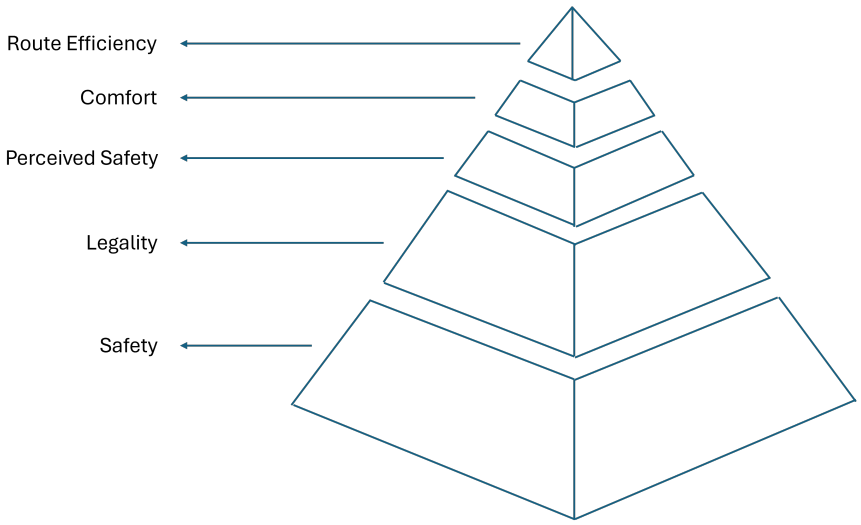


Figura 12: Jerarquía de toma de decisiones.

do, presenta un entorno propicio para investigar y desarrollar soluciones innovadoras en infraestructura vial. La ciudad enfrenta problemas de tráfico significativos que requieren un enfoque integrado de planificación y gestión vial para mejorar la fluidez y reducir los tiempos de viaje.

6.10. Planificación para vehículos autónomos

La planificación de rutas para vehículos autónomos es fundamental para garantizar la seguridad, eficiencia y legalidad de los trayectos. Según Stachniss [42], este proceso se organiza jerárquicamente en diferentes niveles de abstracción y detalle para gestionar eficazmente las múltiples variables involucradas en la conducción autónoma.

6.10.1. Pirámide de jerarquía de objetivos

La planificación se basa en una jerarquía de objetivos representada por una pirámide (Figura 12) [42]. En la base se encuentra la seguridad, el objetivo primordial. La prioridad es evitar colisiones y situaciones peligrosas, asegurando que el vehículo opere de manera segura en todas las condiciones.

El siguiente nivel es la legalidad, esencial para que el vehículo cumpla con las normativas de tráfico, como respetar semáforos, límites de velocidad y señales. El cumplimiento de las leyes garantiza la seguridad y la aceptación social de los vehículos autónomos.

Luego se encuentra la seguridad percibida, relacionada con la sensación de seguridad del pasajero. Un trayecto puede ser técnicamente seguro, pero si el pasajero no se siente cómodo, el viaje no será satisfactorio. Este factor considera la suavidad del trayecto y las maniobras para evitar generar ansiedad.

El confort busca minimizar aceleraciones bruscas y cambios repentinos de dirección para

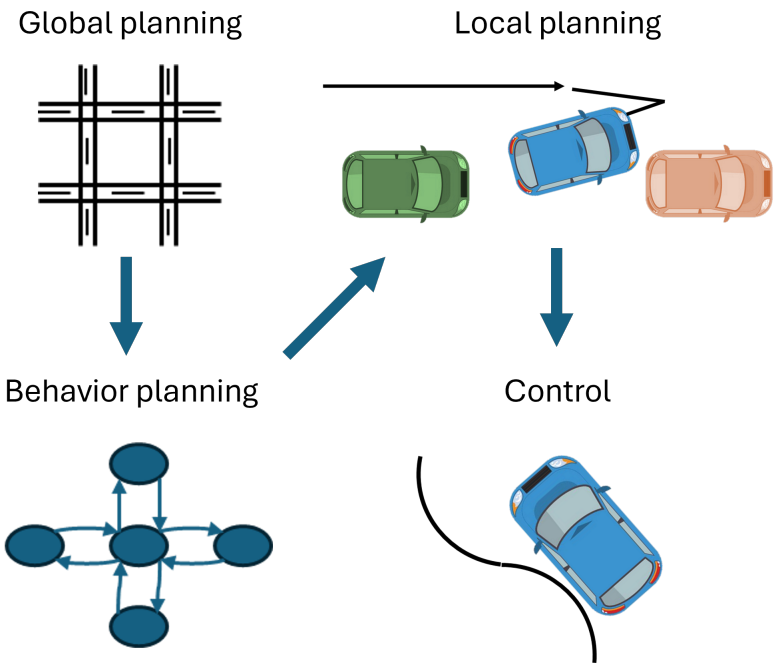


Figura 13: Proceso jerárquico de planificación [42].

que el viaje sea lo más agradable posible. En la cima de la pirámide está la eficiencia de la ruta. Aunque es importante llegar al destino de manera rápida y eficiente, este objetivo se considera después de satisfacer los otros cuatro requisitos. La eficiencia asegura que el vehículo tome rutas óptimas sin comprometer la seguridad, legalidad o confort.

6.10.2. Proceso jerárquico de planificación

El proceso de planificación se estructura en varios niveles jerárquicos, cada uno manejando diferentes aspectos de la ruta. La Figura 13 ilustra este enfoque, que gestiona la complejidad de la conducción autónoma y considera múltiples factores que pueden cambiar en tiempo real [42].

1. Planificación global: Determina la ruta general que el vehículo debe seguir desde el inicio hasta el destino final. Funciona de manera similar a un sistema de navegación, definiendo puntos de referencia que el vehículo debe alcanzar. Opera con una alta abstracción y no requiere actualizaciones frecuentes, ya que proporciona una visión general del trayecto.

2. Planificación del comportamiento: Gestiona las interacciones del vehículo con su entorno inmediato, como otros vehículos, peatones y señales de tráfico. Adapta la ruta general a las circunstancias inmediatas, determinando maniobras específicas como detenerse en un semáforo, ceder el paso o cambiar de carril. Opera con una frecuencia media, respondiendo a cambios en el entorno.

3. Planificación local: Calcula las trayectorias precisas que el vehículo seguirá a corto plazo. Genera la trayectoria detallada para evitar obstáculos y cumplir con los objetivos

de seguridad, legalidad y confort. Se actualiza con alta frecuencia debido a la naturaleza dinámica del entorno de conducción.

4. Control: Ejecuta la trayectoria definida por la planificación local, traduciendo las instrucciones en comandos específicos que guían el movimiento del vehículo, como aceleración, frenado y dirección. Es el nivel más bajo de la jerarquía y se actualiza continuamente para mantener al vehículo en la trayectoria planificada.

Este proceso jerárquico garantiza que cada aspecto de la conducción autónoma se gestione adecuadamente, permitiendo que los vehículos naveguen de manera segura, eficiente y cómoda, adaptándose a las condiciones cambiantes del entorno.

CAPÍTULO 7

Diseño, planificación y evaluación de rutas para vehículos autónomos en entornos urbanos a escala

Este capítulo describe la creación de un entorno simulado inspirado en la Ciudad de Guatemala, donde se desarrollaron y validaron algoritmos de planificación de rutas para vehículos autónomos a escala. La metodología incluye el diseño de carreteras, la digitalización de recorridos, la construcción de grafos direccionales y la implementación de métodos de búsqueda en grafos, asegurando eficiencia y precisión en la navegación autónoma dentro de un entorno urbano complejo y realista.

7.1. Metodología

Se desarrolló un proceso metodológico integral para el diseño, digitalización y evaluación de rutas en un entorno simulado basado en las condiciones viales de la Ciudad de Guatemala. Este proceso se dividió en las etapas clave detalladas a continuación:

1. Diseño de la carretera: Inspirado en elementos viales específicos de la Ciudad de Guatemala (como rotondas, cruces, carriles auxiliares y curvas pronunciadas), se creó un entorno de simulación realista y desafiante. Estos elementos permiten evaluar el desempeño de los vehículos autónomos en diversos escenarios urbanos.

2. Digitalización del recorrido: Utilizando herramientas CAD y de diseño 3D, se digitalizó con precisión el entorno de simulación. Posteriormente, el diseño se exportó en formato DXF y se realizaron ajustes para eliminar elementos no esenciales, asegurando que solo se conservasen las líneas necesarias para la simulación en MATLAB[1].

[1]MATLAB es un entorno de programación y cálculo numérico desarrollado por MathWorks, ampliamente utilizado en ingeniería y ciencias para análisis de datos, desarrollo de algoritmos y creación de modelos.

3. Creación del grafo direccional: Empleando la herramienta DXFtool[2], se importaron y procesaron los datos del diseño en MATLAB. Se limpiaron las líneas, corrigieron sus direcciones y ajustaron sus longitudes, creando así un grafo direccional coherente, elemento fundamental para la planificación de rutas.

4. Implementación de métodos de búsqueda en grafos: Se aplicó el algoritmo de Dijkstra para calcular la ruta más corta entre dos puntos en el grafo direccional. Esto permitió evaluar la eficiencia y precisión en el cálculo de rutas óptimas para la navegación autónoma del vehículo en el entorno simulado.

Este enfoque metodológico, que integra el diseño urbano detallado con herramientas CAD, la digitalización precisa del entorno, la construcción de grafos direccionales y la aplicación de algoritmos de búsqueda eficientes, proporciona una base robusta para la simulación y evaluación de vehículos autónomos a escala en un entorno urbano realista inspirado en la Ciudad de Guatemala.

7.2. Diseño de carretera

Se diseñó una carretera para la simulación de vehículos autónomos en entornos urbanos a escala, inspirada en características viales específicas de la Ciudad de Guatemala, con el objetivo de recrear condiciones de tráfico reales y desafiantes (ver Figura 14). El diseño incluye:

- Rotonda central: Emula la rotonda de la "Calle Mariscal", conocida por sus múltiples entradas y salidas que desafían la gestión del tráfico y la toma de decisiones en tiempo real de los vehículos autónomos.
- "Zona 1": Ubicada en la parte inferior del diseño, reproduce la estructura rectangular y cuadrada característica de esta zona, con cruces de 3 y 4 vías, permitiendo evaluar la capacidad de los vehículos para navegar en entornos urbanos densos y con patrones de tráfico complejos.
- Carril auxiliar: En el lado derecho, se incorpora un carril auxiliar similar al de la "Avenida La Reforma", para probar la habilidad de los vehículos en el manejo de cambios de carril y la interacción con el tráfico.
- "Carretera Muxbal": La sección superior izquierda se asemeja a esta carretera, conocida por sus curvas pronunciadas, ofreciendo un escenario para evaluar el desempeño de los vehículos en condiciones de conducción exigentes y sinuosas.

Estos elementos combinados conforman un entorno de simulación robusto y realista, fundamentado en las especificidades viales de la Ciudad de Guatemala.

[2]DXFtool es una herramienta para MATLAB desarrollada por Mikkel Pedersen [43], que permite leer y graficar archivos DXF en 2D. Soporta entidades como líneas, puntos, arcos, círculos, elipses y polilíneas, respetando el orden y los colores de los objetos, y admite diferentes características de polilíneas y estilos de líneas. Está optimizada para generar gráficos a partir de archivos DXF e importar y graficar modelos CAD. Disponibile en MathWorks File Exchange.

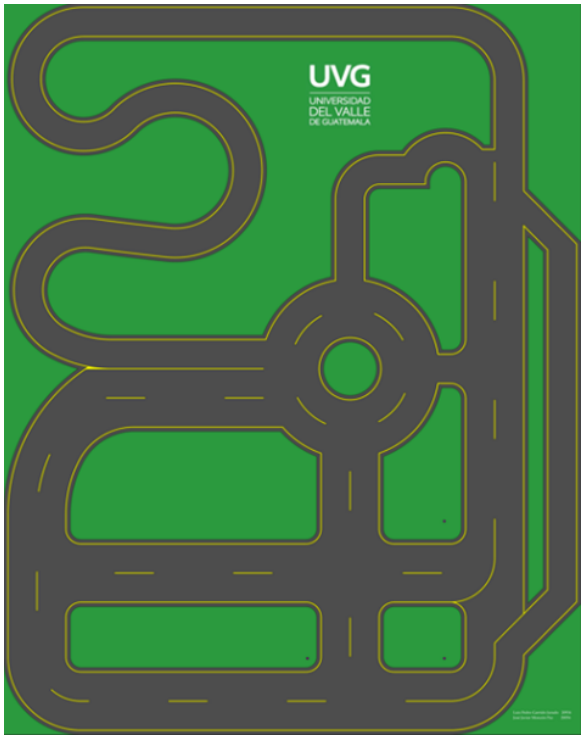


Figura 14: Diseño de la carretera a escala.

7.3. Creación del bosquejo del recorrido del vehículo

Se utilizó Autodesk Inventor Professional 2024[3] para diseñar la carretera en la simulación de vehículos autónomos a escala. La carretera se diseñó asegurando que los extremos de los carriles fueran paralelos, diferenciándose solo en sus dimensiones para facilitar la navegación autónoma. Se generó un bosquejo del recorrido central del carril, correspondiente al camino ideal del vehículo. Siguiendo [3], se mantuvo la escala 1:18.75, con carriles de 183 mm de ancho, situando el recorrido a 91.5 mm de cada extremo para garantizar una trayectoria central y equidistante.

Se incluyeron los posibles cruces y giros del recorrido, emulando un entorno urbano realista y diseñando transiciones suaves para facilitar el control y movimiento fluido del vehículo. Las curvas y giros se configuraron con restricciones de tangencia, asegurando que cada curva fuera tangente a las líneas o curvas conectadas, manteniendo una conducción estable y predecible.

Como se muestra en la Figura 15, el bosquejo inicial incluye elementos adicionales innecesarios para la simulación. Se procedió a limpiar el bosquejo exportándolo en formato DXF[4]. Este archivo se importó en Inkscape[5] para eliminar las líneas no deseadas. Tras la

[3]Autodesk Inventor Professional 2024 es una herramienta de diseño CAD 3D utilizada para crear modelos digitales precisos mediante bocetos, ensamblajes y simulación de movimientos y propiedades físicas.

[4]Un archivo DXF (Drawing Exchange Format) es un formato de intercambio de dibujos desarrollado por Autodesk, ampliamente utilizado para compartir datos de diseño entre diferentes aplicaciones de CAD. La exportación en formato DXF permite una edición posterior detallada y precisa.

[5]Inkscape es un editor de gráficos vectoriales de código abierto que permite manipular y limpiar elementos

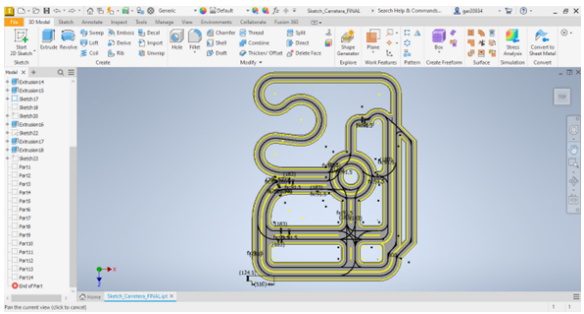


Figura 15: Bosquejo del recorrido del vehículo autónomo.

limpieza, se guardó un archivo DXF que contiene únicamente las líneas necesarias para la simulación.

Este proceso garantizó que el diseño del camino fuera preciso y libre de elementos no requeridos, facilitando una simulación realista y eficiente. Adicionalmente, se exportó una silueta completa de la carretera en formato DXF para ser limpiada, con el objetivo de obtener una representación clara del mapa en el programa de simulación y visualizar detalladamente el entorno vial y el recorrido del vehículo a escala.

7.4. Herramienta DXFtool

La DXFtool es una herramienta de MATLAB diseñada para leer y graficar archivos DXF en 2D. Al utilizar el comando `dxf = DXFtool('nombreArchivo.dxf')`, se obtiene un objeto `dxf` de tipo `1x1 DXFtool` con propiedades como `filename`, `entities`, `ne` y `divisions`. La propiedad `entities` es una estructura que incluye `name`, `layer`, `linetype`, `color`, `line` y `handle`. Esta herramienta permitió importar el diseño realizado en Inventor a MATLAB, facilitando la obtención y manipulación de sus valores para la simulación y análisis detallado del entorno vial diseñado. La Figura 16 muestra el gráfico generado a partir de DXFtool con el archivo que detalla las posibles rutas del vehículo.

7.5. Transformación de trayectorias DXF en conjuntos lineales direccionales

Para crear un grafo para la búsqueda de rutas en la simulación de vehículos autónomos, se realizó el siguiente procedimiento para manipular las líneas obtenidas del archivo DXF:

1. Conversión del DXF con DXFTool: Se importó el diseño realizado en Inventor a MATLAB utilizando DXFTool, obteniendo un formato manipulable.

2. Extracción y limpieza de líneas: A partir de la variable `dxf`, se extrajeron todas las líneas creadas en el archivo DXF. Se eliminaron las líneas repetidas y aquellas cuyo

no deseados del diseño, enfocándose en las líneas que representan el camino ideal del vehículo.

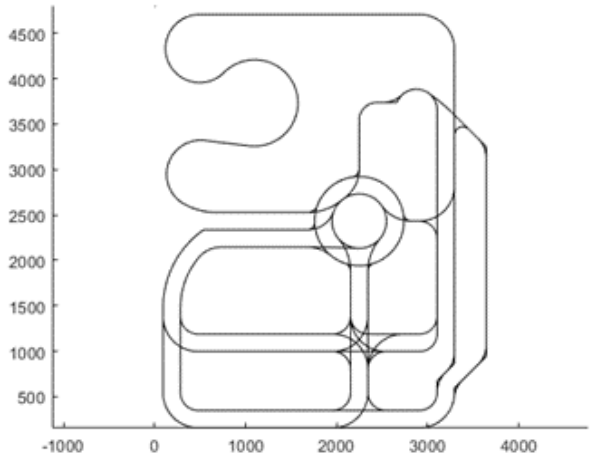


Figura 16: Gráfico resultante de la DXFtool.

punto inicial y final son iguales, ya que no aportan al diseño. Se identificaron grupos de líneas consecutivas, donde el punto final de una línea coincide con el punto inicial de la siguiente y comparten el mismo signo de pendiente global, manteniendo la coherencia en el recorrido.

3. Corrección de la dirección de las líneas: Se ajustó la dirección de cada grupo mediante una función interactiva que muestra el recorrido completo y resalta las líneas del grupo, indicando su dirección actual (Figura 17). El usuario decide si mantener o invertir la dirección de las líneas del grupo. Al invertir, se intercambian los puntos inicial y final de cada línea en el grupo, asegurando que el arreglo tenga las direcciones correctas y evitando que el vehículo transite por vías incorrectas.

4. Ajuste personalizado de direcciones: En grupos que requerían ajustes específicos, se utilizó una función interactiva que permite visualizar rangos de líneas y decidir individualmente si cambiar o mantener la dirección, garantizando que el diseño final refleje con precisión el entorno real.

5. Alineación y conversión de unidades: Se desplazaron todas las líneas para alinearlas con el marco de referencia global, donde el origen (0,0) está en la esquina inferior izquierda del mapa, con el eje `X` hacia la derecha y el eje `Y` hacia arriba, ubicando la esquina opuesta en (3.8, 4.8). Los valores, originalmente en milímetros, se convirtieron a metros para mantener la consistencia en las unidades.

6. Ajuste de longitud de líneas: Para que ninguna línea sea más larga que la más corta original, se dividieron las líneas más largas en segmentos más cortos, considerando la mínima distancia entre las líneas originales. Esto asegura longitudes similares, mejorando el control de seguimiento de trayectorias y la precisión del vehículo. Los valores se redondearon a cuatro decimales, logrando una precisión de ± 0.0001 metros (± 0.1 mm), evitando diferencias numéricas que podrían causar problemas en la simulación.

7. Limpieza final y corrección de conexiones: Se eliminaron líneas repetidas o con puntos inicial y final idénticos. Se ajustaron las líneas para garantizar conexiones adecuadas,

Figura 17: Función de cambio de dirección.

asignando puntos iniciales y finales a sus puntos más cercanos y verificando que no quedaran puntos separados.

Este procedimiento aseguró que el diseño del recorrido fuera preciso y funcional, proporcionando una base sólida para la creación del grafo y la aplicación de algoritmos de búsqueda de trayectorias en la simulación de vehículos autónomos en un entorno urbano a escala.

Para la silueta de la carretera, se obtuvo la variable `dxf`, y se procesaron sus líneas desplazándolas al origen, convirtiéndolas a metros y limpiándolas para conservar solo la información relevante.

7.6. Creación de grafo direccional

Para planificar rutas en la simulación de vehículos autónomos, se creó un grafo direccional a partir de las líneas digitales obtenidas, siguiendo un enfoque similar al utilizado en Google Maps. Este método modela con precisión la red vial, asegurando que todas las partes del grafo estén conectadas y listas para la planificación de rutas óptimas en tiempo real.

El procedimiento en MATLAB para generar el grafo direccional fue el siguiente:

1. Extracción de puntos únicos: Se obtuvieron los puntos únicos a partir de las coordenadas iniciales y finales de las líneas, eliminando duplicados y almacenando solo los distintos.
2. Asignación de índices de nodo: Se creó un mapa que asocia cada punto único con un índice de nodo, facilitando la referencia y mapeo de coordenadas a índices en el grafo. Cada coordenada se vinculó con su índice numérico correspondiente para su uso en la visualización.
3. Preparación de vectores para el grafo: Se prepararon tres vectores:
   - `s`: Contiene los nodos iniciales de cada línea.
   - `t`: Contiene los nodos finales de cada línea.

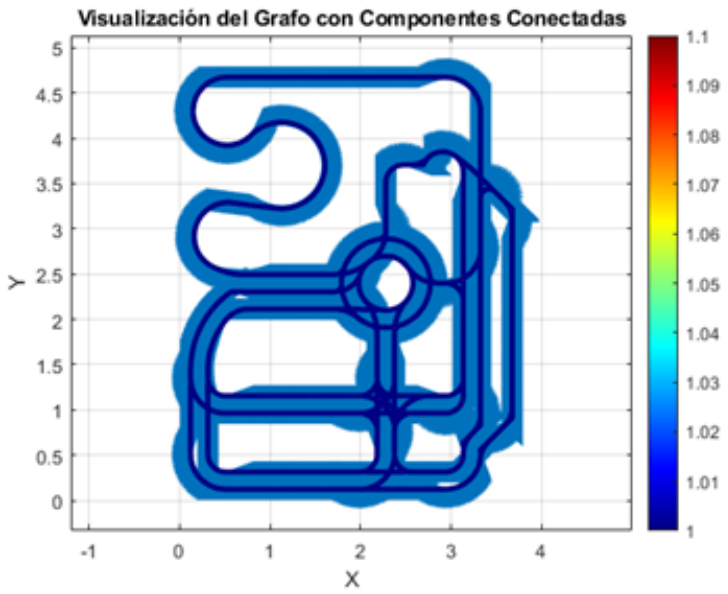


Figura 18: Grafo direccional y visualización de su conectividad.

- `weights`: Contiene los pesos de las aristas, calculados como la distancia euclidiana entre los puntos inicial y final de cada línea.

4. Construcción del grafo dirigido: Utilizando los vectores `s`, `t` y `weights`, se creó el grafo direccional que representa la red vial.

5. Análisis de conectividad: Se identificaron las componentes conectadas del grafo, obteniendo un vector que indica a qué componente pertenece cada nodo.

6. Visualización del grafo: Se visualizó el grafo utilizando un mapa de colores para diferenciar las componentes conectadas, ajustando la barra de colores según el número de componentes. Gracias al correcto procesamiento previo de las líneas, el grafo resultante estaba completamente conectado, lo que se refleja en un color uniforme en toda su extensión (ver Figura 18). Esto indica que no hay secciones aisladas y todas las partes del grafo están interconectadas, facilitando la navegación del vehículo autónomo.

La creación de este grafo direccional permitió una planificación de rutas eficiente y precisa en la simulación de vehículos autónomos en un entorno urbano a escala.

7.7. Cuadrícula de ocupación para la silueta del mapa

Se generó una cuadrícula de ocupación a partir de las líneas de la silueta del mapa con una resolución de 0.01 m (10 mm). Primero, se calcularon los valores máximos y mínimos de las coordenadas de los puntos extremos de las líneas para establecer los límites y determinar el tamaño de la cuadrícula en términos de filas y columnas, redondeando hacia arriba para incluir todas las líneas.

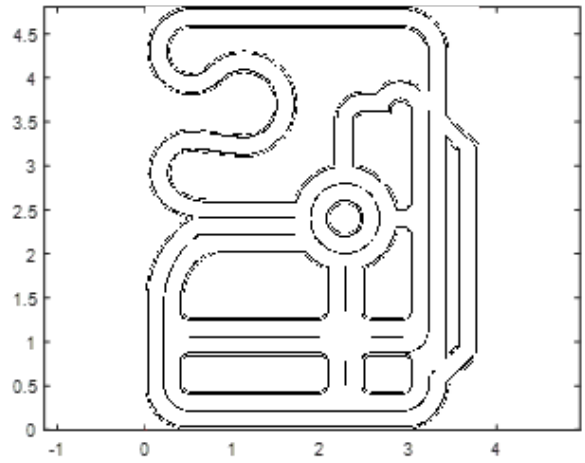


Figura 19: Cuadrícula de ocupación para silueta del mapa.

Luego, se inicializó una matriz de ceros que representa la cuadrícula, donde cada celda puede estar ocupada (valor 1) o libre (valor 0). Por cada línea, se extrajeron sus puntos inicial y final, convirtiendo estas coordenadas a índices de la cuadrícula según la resolución. Se utilizó el algoritmo de Bresenham para trazar las líneas en la cuadrícula, asegurando su representación en un espacio discreto.

Las celdas por donde pasa cada línea se marcaron como ocupadas, siempre que los índices calculados estuvieran dentro de los límites de la cuadrícula, evitando accesos fuera de rango. La cuadrícula de ocupación se visualizó utilizando `imagesc`, mostrando las celdas ocupadas en negro sobre fondo blanco, lo que permite una visualización clara de las áreas ocupadas por las líneas dentro del área especificada. Esto permite graficar la silueta del mapa de manera eficiente sin consumir recursos excesivos que afecten la ejecución del programa. En la Figura 19 se muestra el resultado final.

7.8. Método de planificación de movimiento

Se implementó un método de planificación de movimiento en MATLAB para calcular rutas óptimas dentro del grafo direccional creado, similar al algoritmo de Dijkstra utilizado en Google Maps. El objetivo fue encontrar la ruta más corta entre dos puntos en un grafo ponderado utilizando la función `shortestpath`.

La función recibe como parámetros el grafo direccional, las coordenadas de los nodos y las coordenadas de los puntos inicial y final. Primero, determina los nodos más cercanos a los puntos dados mediante una función auxiliar que traduce las coordenadas en nodos del grafo, permitiendo que la búsqueda de la ruta se realice dentro del grafo.

Una vez identificados los nodos más cercanos, la función calcula la ruta más corta entre ellos utilizando `shortestpath`, que emplea el algoritmo de Dijkstra debido a que todas las aristas del grafo tienen pesos no negativos. Este algoritmo es adecuado por su rapidez y

eficiencia en grafos donde cada nodo tiene pocas aristas asociadas.

El resultado es una lista de índices de nodos que conforman la ruta más corta y la distancia total de dicha ruta. Si no se encuentra una ruta entre los nodos especificados, la función devuelve un mensaje indicando que no se encontró ruta y retorna un valor vacío. Si se encuentra una ruta, se extraen las coordenadas de los nodos correspondientes, creando una matriz de coordenadas que representa la ruta en términos espaciales. Esta matriz se asigna a `ruta` y la distancia se devuelve en `d`.

Este enfoque traduce la ruta óptima directamente en coordenadas espaciales que pueden ser utilizadas por los vehículos autónomos para navegar de manera eficiente en términos de distancia. Al determinar caminos óptimos en un entorno urbano complejo representado por un grafo de caminos y conexiones, se optimiza la navegación y el análisis de trayectorias, contribuyendo a la implementación de sistemas autónomos en entornos simulados. De esta manera, los vehículos pueden navegar de manera efectiva y segura, maximizando la eficiencia operativa en el entorno simulado.

7.9. Resultados de la función calcular ruta

7.9.1. Análisis estadístico

Se realizaron 100 iteraciones de la función `calcular ruta`, almacenando en cada una datos como coordenadas iniciales y finales, distancia euclidiana, distancia de la ruta, tamaño de la ruta y tiempo de ejecución. Los resultados demuestran alta eficiencia y precisión en los escenarios simulados.

La distancia euclidiana promedio es de 2.1824 metros, mientras que la distancia de la ruta promedia 6.9499 metros. La diferencia se debe a las restricciones del grafo que obligan a seguir caminos específicos en lugar de una línea recta. La correlación entre ambas distancias es moderada (0.3831), indicando que aunque las rutas más largas tienden a tener una mayor distancia euclidiana, esta relación no es estrictamente lineal. Esto refleja las complejidades del entorno vial simulado, donde la ruta óptima puede diferir significativamente de la distancia directa entre dos puntos.

El tiempo de ejecución promedio es de 1.8 milisegundos con una desviación estándar de 0.6 milisegundos. El histograma de los tiempos de ejecución (Figura 20) muestra que la mayoría de las iteraciones se completaron rápidamente, confirmando la consistencia y rapidez de la función. Existe una correlación positiva (0.5506) entre la distancia de la ruta y el tiempo de ejecución, sugiriendo que rutas más largas requieren más tiempo de cálculo, aunque este incremento es mínimo y no afecta significativamente el rendimiento.

El tamaño de la ruta, definido como el número de nodos en la ruta, tiene una fuerte correlación (0.9999) con la distancia de la ruta, lo cual es esperado, ya que rutas más largas generalmente implican más nodos. También muestra una correlación positiva con el tiempo de ejecución (0.5520), reforzando la relación entre la complejidad de la ruta y el tiempo necesario para calcularla.

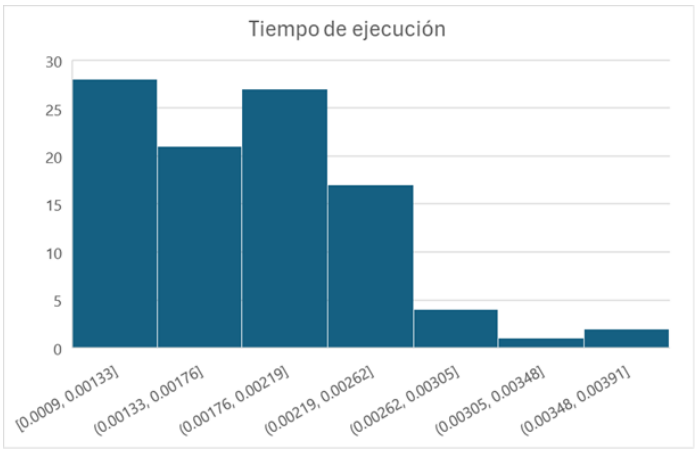


Figura 20: Histograma de tiempos de ejecución de `calcular ruta`.

La función `calcular ruta` demostró un desempeño excelente. Los tiempos de ejecución fueron consistentemente bajos, incluso para rutas largas y complejas, lo cual es crucial para aplicaciones en tiempo real en vehículos autónomos. La variabilidad mínima en los tiempos de ejecución indica que la función es óptima y predecible en su rendimiento, siendo altamente valiosa para la planificación de movimientos en entornos urbanos simulados.

Media	0.001802
Error típico	5.7524E-05
Mediana	0.0018
Moda	0.0013
Desviación estándar	0.00057524
Varianza de la muestra	3.3091E-07
Curtosis	0.36177677
Coeficiente de asimetría	0.74645618
Rango	0.0027
Mínimo	0.0009
Máximo	0.0036
Suma	0.1802
Cuenta	100

Cuadro 1: Estadística descriptiva del tiempo de ejecución (s) de la función `calcular ruta`.

7.9.2. Análisis descriptivo

Se evaluó el comportamiento de los vehículos autónomos en un entorno vial basado en la Ciudad de Guatemala mediante cinco pruebas de simulación con diferentes escenarios de inicio y destino. Estas pruebas analizaron cómo los vehículos respetan las vías establecidas y responden a las restricciones del entorno vial.

En la prueba 1 (Figura 21), los puntos de inicio y final están cercanos, pero debido a la configuración de la rotonda, el vehículo debe dar la vuelta completa para llegar a su destino. Esto evaluó la capacidad del vehículo para manejar intersecciones circulares y cumplir la

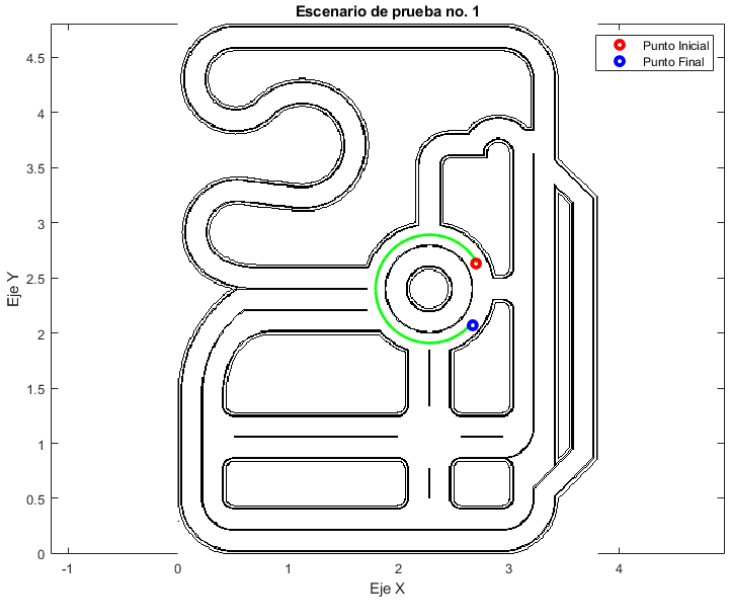


Figura 21: Ruta encontrada - Caso 01.

señalización de entrada y salida de la rotonda, reflejando un comportamiento que respeta las reglas de tránsito locales.

La prueba 2 (Figura 22) situó al vehículo en una carretera de una sola vía, obligándolo a seguir la dirección única antes de determinar el camino óptimo al destino. Esta prueba validó que el vehículo respeta las direcciones de tráfico y busca rutas alternativas dentro de las restricciones viales, demostrando eficiencia al calcular la ruta tras salir de la vía unidireccional.

En la prueba 3 (Figura 23), el escenario incluyó carriles auxiliares, una rotonda y curvas pronunciadas. El vehículo tomó una ruta más larga para llegar al destino, indicando que respeta las vías establecidas y maneja múltiples tipos de intersecciones y cambios de dirección en un entorno complejo, manteniéndose en las vías correctas y evitando rutas prohibidas.

La prueba 4 (Figura 24) obligó al vehículo a pasar por una intersección de cuatro vías, evaluando su capacidad para manejar intersecciones complejas y seleccionar la vía más adecuada. La simulación mostró que el vehículo maneja la intersección eficientemente, eligiendo el camino que minimiza la distancia y respeta las reglas de tránsito, confirmando su capacidad para gestionar entornos urbanos densos con múltiples opciones de ruta.

En la prueba 5 (Figura 25), los puntos de inicio y final están en extremos opuestos del mapa, requiriendo un recorrido largo. Este escenario evaluó la eficiencia del sistema en la planificación de rutas a larga distancia y su capacidad para evitar atajos que violen las restricciones viales. La ruta calculada siguió todas las vías permitidas, llegando al destino de manera eficiente y destacando la robustez del sistema en entornos urbanos extensos.

En general, el análisis de las rutas obtenidas demostró que el sistema de planificación de trayectorias responde adecuadamente a las restricciones viales. Los vehículos respetaron

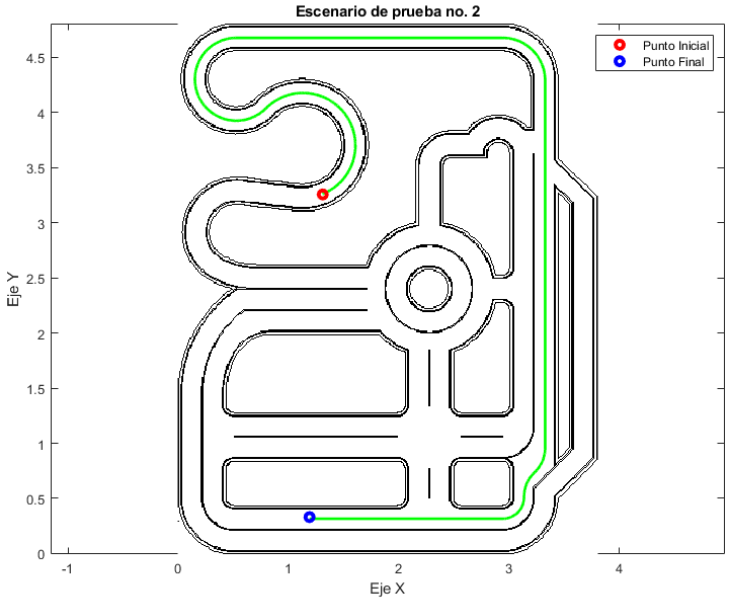


Figura 22: Ruta encontrada - Caso 02.

direcciones únicas, manejaron intersecciones complejas y siguieron rutas óptimas dentro de las restricciones establecidas. Esto sugiere que el sistema de detección de rutas es robusto y adecuado para simular condiciones de tráfico realistas, proporcionando una base sólida para el desarrollo de la navegación autónoma. Las pruebas confirmaron que los vehículos pueden adaptarse a diversas configuraciones viales y planificar rutas efectivamente, esencial para su implementación en entornos urbanos reales.

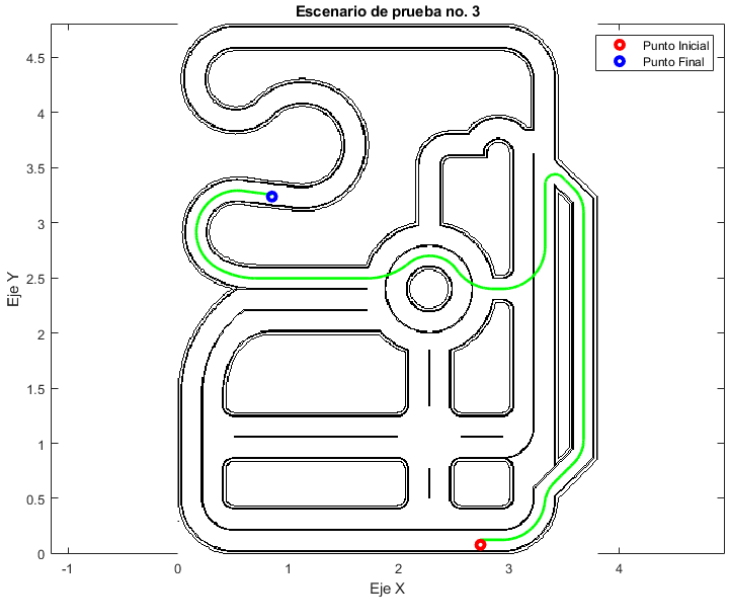


Figura 23: Ruta encontrada - Caso 03.

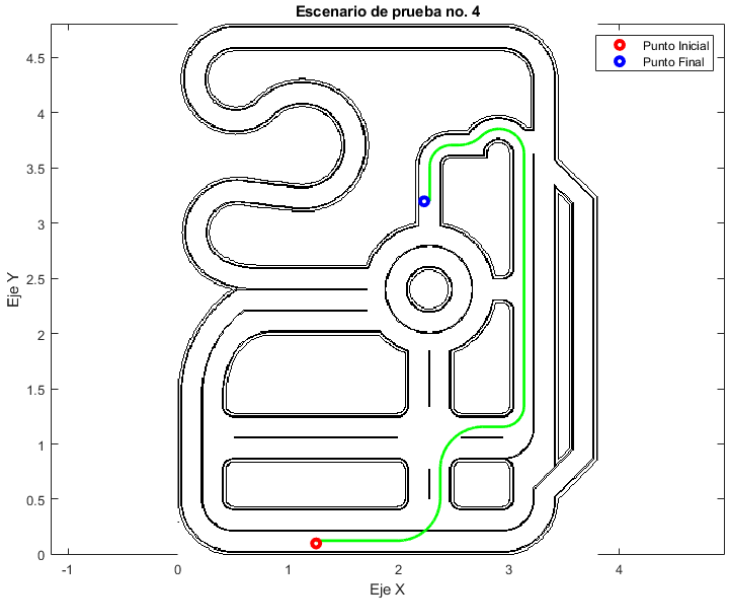


Figura 24: Ruta encontrada - Caso 04.

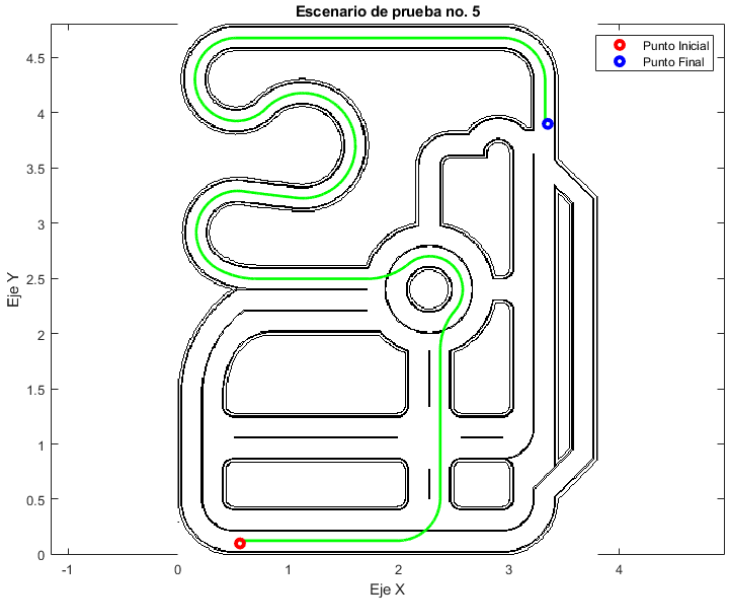


Figura 25: Ruta encontrada - Caso 05.

CAPÍTULO 8

Implementación y validación de sistemas de control y señalización en vehículos autónomos a escala

Este capítulo aborda la implementación y validación de sistemas de control y señalización en vehículos autónomos a escala. Se exploran los métodos empleados para gestionar señales de tránsito, la traducción de coordenadas entre el sistema Robotat y el mapa de simulación, así como la aplicación de dos controladores para asegurar una navegación autónoma eficiente. Además, se analizan las limitaciones encontradas al utilizar MATLAB para la simulación, lo que motivó la migración a Python, mejorando significativamente el rendimiento y la flexibilidad del sistema. Finalmente, se examinan los resultados obtenidos tanto en simulaciones virtuales como físicas, comparando el desempeño de distintos controladores bajo diversas condiciones.

8.1. Metodología

Para implementar y validar sistemas de control y señalización en vehículos autónomos a escala, se desarrolló un entorno de simulación en MATLAB que incorpora señales de tránsito como alto, reducción de velocidad y semáforos, evaluando la respuesta de los vehículos a estas señales. Se tradujeron las coordenadas del sistema de captura de movimiento Robotat al mapa de simulación, asegurando una correcta interpretación de las posiciones y orientaciones en el entorno simulado. Se implementaron dos controladores para el movimiento de los vehículos: un controlador PID con acercamiento exponencial y un controlador basado en Lyapunov, analizando sus características y desempeño.

Debido a limitaciones de rendimiento y gestión de tiempo en MATLAB, se migró el código a Python, lo que resultó en mayor eficiencia. En este nuevo entorno, se diseñó una clase para el carro autónomo que incluye funcionalidades como evitar colisiones y realizar

parqueo automático, garantizando un entorno de prueba seguro y realista.

Esta metodología permitió implementar y evaluar sistemas de control y señalización en vehículos autónomos en simulaciones virtuales y físicas con el robot Pololu 3pi+ 32U4 OLED. Se destacaron las mejoras en precisión y estabilidad logradas con la migración a Python.

8.2. Señales de tránsito

Se implementaron señales de tránsito específicas para simular un entorno urbano realista y evaluar la interacción del vehículo autónomo en diferentes escenarios. Las señales consideradas fueron: alto, reducción de velocidad y semáforos en sus estados verde, amarillo y rojo. A continuación, se describe su implementación en MATLAB y las acciones que el vehículo autónomo realiza en respuesta a cada señal.

8.2.1. Estructura de las señales en MATLAB

Las señales de tránsito se gestionaron en MATLAB como estructuras (`structs`), permitiendo una organización clara y flexible de los datos asociados a cada señal y facilitando la gestión de múltiples señales simultáneamente. Cada señal posee los siguientes campos:

1. `id`: Identificador único de la señal.
2. `activo`: Indicador booleano de si la señal está activa.
3. `tipo`: Tipo de señal (*alto*, *bajar velocidad*, *semáforo*).
4. `posicion`: Ubicación de la señal en el mapa. Para las señales de alto y semáforo es un punto, mientras que para la señal de bajar velocidad se define mediante dos puntos que delimitan un rectángulo de aplicación.
5. `valor`: Parámetro asociado a la señal según su tipo. Para la señal de alto, es el tiempo de detención en segundos; para bajar velocidad, es un factor (entre 0.0 y 1.0) que reduce la velocidad; para el semáforo, indica su color (*verde*, *amarillo* o *rojo*).

8.2.2. Señal de alto

La señal de alto detiene el vehículo por un tiempo determinado antes de continuar su trayectoria. Su implementación es la siguiente:

1. Detección de la señal: Se verifica si la posición del vehículo está dentro de un umbral de 0.06 m respecto a la señal de alto. Si es así, se almacena el `id` de la señal, el tiempo de detención, y se activa una bandera indicando que el vehículo debe detenerse.
2. Ejecución del alto: Durante cada iteración del código:

a) Si no hay un alto cercano, se llama a la función de detección. Si se encuentra un alto, se almacena el tiempo inicial y se desactiva la señal.

b) Si hay un alto cercano, se evalúa si el tiempo transcurrido desde el tiempo inicial es mayor o igual al tiempo de detención. Si se cumple, se desactiva la bandera de alto cercano y se almacena el tiempo actual.

c) Después de cumplir con el alto, se espera 6 segundos antes de reactivar la señal, evitando altos consecutivos.

3. Implementación del freno: Si la bandera de alto cercano está activada, se establecen la velocidad lineal y angular del vehículo a cero, deteniéndolo. Esta detección se realiza en cada iteración del código, asegurando una respuesta inmediata.

8.2.3. Señal de bajar velocidad

La señal de bajar velocidad requiere que el vehículo reduzca su velocidad en un factor dado mientras se encuentra dentro del área afectada. El procedimiento es:

1. Detección de la señal: Para cada señal activa de bajar velocidad, se evalúa si la posición del vehículo está dentro del rango definido. Si es así, se almacena el factor de velocidad.

2. Aplicación del factor de velocidad: El factor almacenado se multiplica por la velocidad máxima del vehículo, ajustando la velocidad efectiva en los cálculos del controlador. Si el vehículo está en el área de influencia de múltiples señales, se utiliza el factor más restrictivo (mínimo). Este ajuste dinámico asegura que la reducción de velocidad se cumpla con precisión.

8.2.4. Señal de semáforo

El semáforo regula la detención y avance del vehículo según su estado (*rojo*, *amarillo*, *verde*). La implementación es:

1. Detección de la señal: Se evalúa si el vehículo está dentro de un umbral de 0.06 m del semáforo. Si es así, se almacena el `id` y el color del semáforo, y se activa una bandera de semáforo cercano.

2. Ejecución según el color:

   a) Rojo o amarillo: Si la bandera está activada y el color es rojo o amarillo, se establecen las velocidades lineal y angular a cero, deteniendo el vehículo.

   b) Verde: Si el color es verde, el vehículo continúa su movimiento normal.

3. Actualización dinámica: Se implementó una función que permite actualizar el estado del semáforo mediante un archivo `.csv`, realizando cambios en el color de manera dinámica sin detener la ejecución del código.

La implementación detallada de cada tipo de señal asegura que el vehículo responda de manera precisa y adecuada a diversas situaciones de tráfico. La señal de alto simula intersecciones donde los vehículos deben detenerse y ceder el paso; la señal de bajar velocidad ajusta dinámicamente la velocidad del vehículo, replicando zonas con restricciones; el semáforo regula el flujo en intersecciones, garantizando que los vehículos se detengan o avancen según corresponda. Esta gestión eficiente y flexible de las señales, permitiendo múltiples señales simultáneamente y modificaciones dinámicas, es fundamental para evaluar la capacidad del vehículo de adaptarse a cambios inesperados y condiciones de tráfico realistas.

8.3. Traducción del lenguaje de Robotat a coordenadas del mapa

Para garantizar que las posiciones y orientaciones detectadas por el sistema OptiTrack se interpreten correctamente en el mapa de simulación, se realizó una traducción de las coordenadas de Robotat al sistema del mapa. Esto es esencial para que el vehículo autónomo se comporte con precisión en el entorno simulado, lo cual es crucial para el desarrollo y evaluación de los sistemas de control.

El sistema Robotat utiliza seis cámaras OptiTrack para detectar marcadores retroreflectivos en los objetos, calculando su posición en el espacio tridimensional. El sistema de coordenadas de Robotat tiene su origen en el centro de la plataforma, con el eje X positivo hacia la izquierda y el eje Y positivo hacia la derecha cuando se observa desde la puerta de entrada. En contraste, el mapa de simulación tiene su origen en la esquina inferior izquierda, con el eje X positivo hacia la derecha y el eje Y positivo hacia arriba. Debido a estas diferencias, fue necesario ajustar las posiciones y orientaciones de los datos de Robotat al sistema de coordenadas del mapa.

El proceso de traducción, ilustrado en la Figura 26, se basa en transformaciones matemáticas que alinean y convierten las coordenadas entre ambos sistemas. Se consideran los sistemas de Robotat (O), el marcador (M) y el mapa (V). Primero, se genera una matriz de transformación VT_M que alinea los ejes del sistema del marcador al del mapa. Luego, se obtiene la pose del marcador en el mapa y se traduce al origen de Robotat mediante la matriz OT_M. Al multiplicar OT_M con MT_V, se obtiene OT_V, permitiendo transformar los datos del mapa al sistema de Robotat. La matriz inversa VT_O realiza la conversión inversa, adaptando las coordenadas de Robotat al sistema del mapa. Esta matriz se aplica a la pose del vehículo para ajustar su posición en el mapa, y se corrige la orientación mediante un delta experimental, asegurando una alineación precisa en la simulación.

Esta traducción es fundamental para interpretar y utilizar correctamente los datos de posicionamiento y orientación en la simulación. Al adaptar las posiciones y orientaciones detectadas por Robotat al sistema de coordenadas del mapa, se asegura que el vehículo autónomo se comporte de manera precisa en el entorno simulado, esencial para la interacción entre múltiples vehículos y la evaluación de los sistemas de control en condiciones de tráfico realistas.

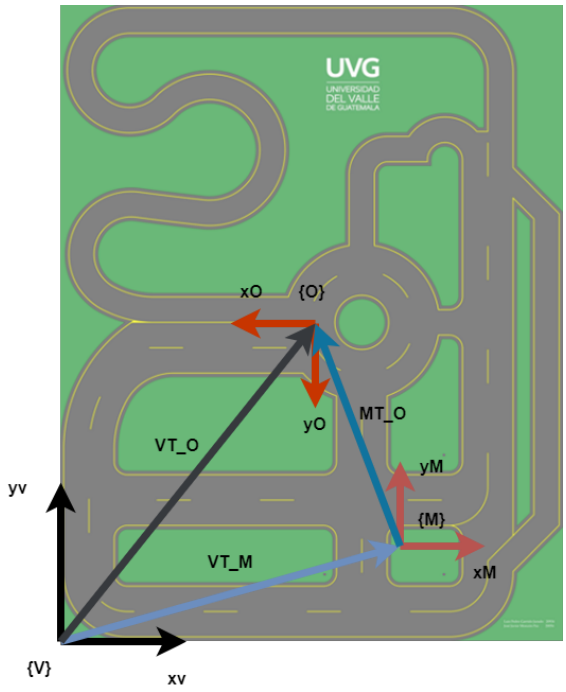


Figura 26: Traducción de sistemas.

8.4. Sistemas de control

El control preciso del movimiento es esencial para el seguimiento de las rutas planificadas y la respuesta adecuada a las señales de tránsito. Se implementaron dos controladores en MATLAB: un controlador PID con acercamiento exponencial para mejorar la precisión en el seguimiento de trayectorias, y un controlador basado en Lyapunov para asegurar la estabilidad del sistema.

8.4.1. Controlador PID con acercamiento exponencial

La implementación del controlador PID con acercamiento exponencial se centró en aplicar las fórmulas del marco teórico para gestionar la posición y orientación del vehículo. En cada iteración del lazo de control, se calcularon los errores de posición y orientación. El término exponencial se aplicó al error de posición para ajustar la velocidad lineal, suavizando el acercamiento del robot a la trayectoria objetivo.

La velocidad angular se controló mediante las componentes proporcional, integral y derivativa del error de orientación, manejando cuidadosamente la integral del error para evitar acumulaciones que afecten la estabilidad. Se optimizó el código para calcular eficientemente las velocidades de control, minimizando el tiempo de procesamiento y permitiendo que el vehículo responda rápidamente a variaciones en la trayectoria.

La combinación de las velocidades lineal y angular definió un vector de control utilizado para ajustar dinámicamente la trayectoria del robot, asegurando un recorrido suave y estable hacia el objetivo. La implementación resultó en un seguimiento de trayectorias, alineado con

las expectativas teóricas.

8.4.2. Controlador basado en Lyapunov

Para el controlador basado en Lyapunov, se aplicaron las fórmulas del marco teórico para garantizar la estabilidad durante el seguimiento de la trayectoria, adaptándolas a un entorno de simulación en tiempo real. Se implementó el cálculo del diferencial de la trayectoria para que el vehículo cumpliera con la "velocidad máxima modificada", adaptándose a la velocidad media en cada iteración y ajustando dinámicamente la posición objetivo a lo largo de la trayectoria.

La integración de la velocidad esperada $\dot{p}_d$ incorporó un componente predictivo al modelo, mejorando la precisión del seguimiento y aportando robustez al controlador al permitir que el robot responda eficazmente a perturbaciones. Se calcularon la matriz Jacobiana y su inversa en cada paso para transformar las velocidades de referencia entre los marcos del vehículo y del entorno, aplicando correcciones de trayectoria con precisión según la orientación y posición del vehículo.

La implementación final permitió que el vehículo siguiera la trayectoria deseada de manera estable y precisa, ajustando las velocidades lineal y angular en tiempo real. Este enfoque logró un control robusto, capaz de adaptarse a variaciones en la trayectoria y garantizar la estabilidad del sistema bajo diferentes condiciones operativas.

8.5. Limitaciones de MATLAB y migración a Python

El uso de MATLAB para la simulación y control de vehículos autónomos a escala presentó limitaciones significativas que afectaron la eficiencia y precisión en tiempo real. Uno de los principales problemas fue el rendimiento y tiempo de ejecución. Se observó que MATLAB podía tardar hasta 300 ms o más en completar una iteración del código, especialmente al incluir tareas adicionales como la representación en tiempo real de la trayectoria. Este tiempo elevado afecta negativamente la simulación, ya que incrementos en el período de iteración pueden invalidar cálculos críticos que dependen de una sincronización precisa.

El Cuadro **??** muestra los resultados de cinco iteraciones en la simulación virtual, evaluando el tiempo promedio que MATLAB tarda en calcular una iteración mientras grafica la trayectoria del vehículo. Los tiempos de ejecución oscilan entre 0.0181 y 0.2187 segundos, con una media de 0.04498 segundos por iteración. Aunque estos tiempos son aceptables en el entorno simulado, cualquier incremento adicional en tareas, como la comunicación en tiempo real con Robotat, aumenta considerablemente este período, haciendo inviable su uso para aplicaciones en tiempo real.

Además, el ecosistema Robotat requiere comunicación inalámbrica por WiFi para proporcionar la retroalimentación necesaria en los cálculos del controlador. Aunque MATLAB puede manejar esta comunicación, el rendimiento es significativamente más lento al combinarla con otras tareas. La dificultad para implementar multithreading en MATLAB limita la capacidad de ejecutar múltiples tareas simultáneamente, como la representación en tiem-

Iteración	No. 1	No. 2	No. 3	No. 4	No. 5	General
Media (s)	0.0335	0.0336	0.0477	0.0477	0.0624	0.04498
Máximo (s)	0.0626	0.0784	0.0892	0.1108	0.2187	0.2187
Mínimo (s)	0.0197	0.0181	0.0304	0.0281	0.0252	0.0181

Cuadro 2: Resultados de las iteraciones

po real, la comunicación con Robotat y la gestión de múltiples vehículos. Intentar realizar estas tareas simultáneamente aumenta drásticamente el tiempo de ejecución, impidiendo que el sistema opere en tiempo real con la precisión necesaria. Estos factores subrayaron la necesidad de considerar alternativas más eficientes.

Se decidió migrar de MATLAB a Python para mejorar el rendimiento y la capacidad de respuesta del sistema de control. Python ofreció varias ventajas:

1. Mejor tiempo de ejecución: Python redujo el tiempo de ejecución, mejorando la respuesta de los controladores, el procesamiento y la visualización.
2. Mayor capacidad de procesamiento: Con bibliotecas optimizadas como NumPy, Python permitió un procesamiento más eficiente, preparando el entorno para la simulación y control de una mayor cantidad de vehículos, tanto en simulación como en aplicaciones físicas.
3. Visualización en tiempo real: Bibliotecas como Matplotlib facilitaron una visualización en tiempo real más eficiente y dinámica, permitiendo el monitoreo continuo de las simulaciones virtuales y físicas.
4. Capacidad de multithreading: Python ofrece capacidades avanzadas de multithreading y multiprocessing, permitiendo ejecutar partes del código simultáneamente y mejorar la eficiencia general del sistema.

En resumen, aunque MATLAB es una herramienta poderosa para el desarrollo inicial y la simulación, sus limitaciones en rendimiento y comunicación hicieron necesaria la migración a Python. Esta transición mejoró la eficiencia y capacidad de respuesta del sistema de control de vehículos autónomos en el entorno de Robotat, permitiendo una mejor simulación y control en tiempo real, y preparando el sistema para escenarios más complejos con un mayor número de vehículos.

8.6. Clase para el carro autónomo

En el desarrollo de simulaciones para vehículos autónomos a escala, se creó una clase específica `CarroAutonomo` en Python que encapsula todos los aspectos esenciales del comportamiento y las características del vehículo. Esta clase permite gestionar eficientemente múltiples vehículos en la misma simulación, lo cual es crucial para estudiar la interacción entre ellos y evaluar su comportamiento en condiciones de tráfico realistas.

En el método constructor `__init__`, se inicializan atributos como el número de identificación, velocidad máxima, ruta planificada, señales de tránsito, estado del vehículo, constantes del controlador (PID con acercamiento exponencial o basado en Lyapunov), y diversas banderas y parámetros necesarios para su funcionamiento. Estos atributos incluyen parámetros básicos del vehículo y específicos para el control, como condiciones iniciales y gestión de señales de tránsito.

La clase incluye métodos que gestionan el comportamiento del vehículo en la simulación:

- Método de control: Calcula las velocidades de las ruedas en función del controlador seleccionado, ajustando las velocidades lineal y angular según los errores de posición y orientación en cada iteración. Considera factores externos como señales de tránsito y evita colisiones con otros vehículos.
- Detección y respuesta a señales de tránsito: Detecta señales como alto, reducción de velocidad y semáforos. Si el vehículo se aproxima a una señal, ajusta su comportamiento en consecuencia, por ejemplo, deteniéndose ante una señal de alto o reduciendo la velocidad en zonas designadas.
- Evitación de colisiones: Ajusta la velocidad del vehículo según la proximidad de otros vehículos, asegurando un comportamiento seguro y realista en el entorno simulado.

La implementación de la clase `CarroAutonomo` en Python mejora significativamente el rendimiento y la flexibilidad de la simulación, permitiendo una gestión más eficiente y realista de los vehículos en el entorno simulado. Al encapsular las propiedades y comportamientos en una clase, se facilita la integración y control de múltiples vehículos, lo que es fundamental para el desarrollo y validación de sistemas de control en la navegación autónoma.

8.7. Método para evitar colisiones entre vehículos

Para asegurar un entorno de prueba seguro y realista en la simulación de vehículos autónomos a escala, se desarrolló el método `evitar_colision`, diseñado para detectar y prevenir colisiones entre vehículos ajustando dinámicamente su velocidad en función de la proximidad a otros vehículos. Esto se logra mediante un factor de velocidad que reduce la velocidad cuando es necesario, manteniendo una distancia segura y evitando colisiones.

Inicialmente, se asume que no hay proximidad a otros vehículos y el factor de velocidad se establece en 1.0, indicando que no es necesaria una reducción de velocidad. Se calcula una distancia de seguridad basada en la velocidad máxima del vehículo y el tiempo de simulación, ajustada con un margen adicional.

La función revisa la lista de otros vehículos en el entorno. Si la lista está vacía, no hay riesgo de colisión. De lo contrario, se itera sobre cada vehículo para evaluar su proximidad en relación con el vehículo actual, asegurándose de no compararlo consigo mismo. Se calcula la posición relativa y se rota el sistema de coordenadas del vehículo actual para alinear la dirección de movimiento, utilizando transformaciones trigonométricas que ajustan las coordenadas relativas según el ángulo de orientación del vehículo. Esto permite evaluar

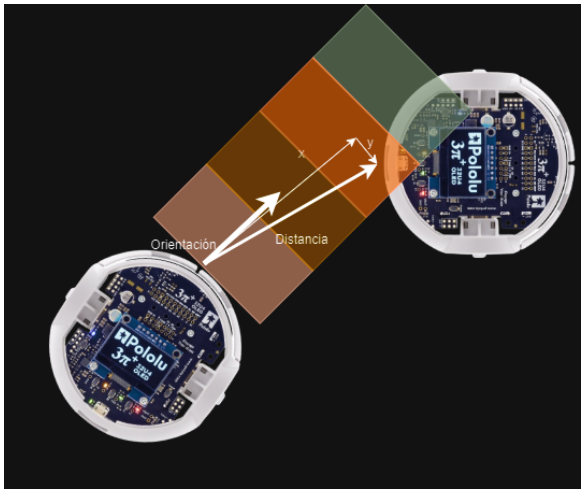


Figura 27: Lógica para evitar colisión entre vehículos.

correctamente la proximidad de otros vehículos en la dirección de movimiento (ver Figura 27).

Solo se consideran los vehículos que están delante y dentro del ancho de carril definido, evitando ajustes innecesarios de velocidad para vehículos en carriles adyacentes. Si un vehículo se encuentra dentro de un rango de proximidad crítico, se ajusta gradualmente el factor de velocidad para reducir la velocidad del vehículo actual, asegurando que mantenga una distancia segura. La función evalúa múltiples distancias y factores de velocidad para determinar el ajuste adecuado, proporcionando una respuesta gradual y precisa y evitando reducciones bruscas que podrían afectar la estabilidad del vehículo.

La implementación de este método permite que múltiples vehículos interactúen de manera segura en el entorno simulado, ajustando dinámicamente sus velocidades para mantener distancias seguras y evitar accidentes. Esto es fundamental para estudiar y mejorar los sistemas de control de los vehículos, contribuyendo al desarrollo de la navegación autónoma más segura y eficiente en condiciones de tráfico vehicular.

8.8. Funcionalidad de parqueo automático

Se implementó una funcionalidad de parqueo automático en el vehículo autónomo a escala para permitir un estacionamiento preciso y eficiente al finalizar su recorrido. El proceso se desarrolló de la siguiente manera:

Una vez completada la ruta establecida, el vehículo inició un temporizador de 10 segundos antes de comenzar el parqueo, asegurando una transición suave y permitiendo visualizar el cumplimiento de la ruta. Tras el tiempo de espera, calculó la ruta desde su posición actual hasta el punto de parqueo asignado, utilizando las coordenadas del grafo direccional previamente creado para determinar el camino óptimo.

El vehículo siguió la ruta calculada hacia el punto de parqueo. Al finalizar el recorrido, si el controlador no estaba configurado en PID con acercamiento exponencial, se realizó

el cambio para aprovechar su componente angular, desactivando las constantes del cálculo de velocidad lineal. Antes de entrar al espacio de parqueo, ajustó su orientación utilizando el controlador PID para reducir el error a 10 grados o menos, asegurando una alineación correcta.

Con la orientación adecuada, procedió a moverse hacia el punto de parqueo final de manera controlada, utilizando el controlador original de la trayectoria para ajustar las velocidades lineal y angular, garantizando una entrada suave y precisa en el espacio de parqueo.

Dentro de la clase del vehículo autónomo, las funcionalidades de parqueo se integraron mediante banderas y condiciones que aseguraban la correcta ejecución del proceso, indicando si el vehículo había calculado y estaba siguiendo la ruta hacia el punto de parqueo, si estaba realizando los ajustes finales, si había ajustado su orientación y si finalizó el proceso de parqueo.

Esta implementación permitió una transición fluida desde la finalización de la ruta principal hasta el estacionamiento en el punto designado, combinando controladores para ajustar la orientación y asegurar un estacionamiento eficiente y seguro en el entorno simulado. Este proceso fue crucial para validar sistemas de navegación autónoma en condiciones de tráfico realistas.

8.9. Análisis del comportamiento del vehículo autónomo en simulación virtual

8.9.1. Análisis estadístico

Se realizó un análisis comparativo del comportamiento de un vehículo autónomo a escala, probado en ocho ocasiones utilizando dos controladores: PID con acercamiento exponencial y Lyapunov. Cada controlador se evaluó a diferentes velocidades máximas, analizando el error cuadrático medio (ECM), la semejanza en posiciones `X` y `Y`, estadísticas descriptivas de las velocidades lineal y angular, y los tiempos de simulación.

El ECM en posiciones `X` y `Y` (Cuadro 3) indica la precisión al seguir la trayectoria deseada. El controlador Lyapunov presentó menores valores de ECM, demostrando mayor precisión en la trayectoria respecto al PID.

Controlador	Velocidad Máxima (m/s)	ECM en X	ECM en Y
Lyapunov	0.07	8.65e-05	0.001383
Lyapunov	0.1	9.34e-05	0.001777
PID	0.1	0.065513	0.032592
Lyapunov	0.13	0.000101	0.002028
PID	0.13	0.062683	0.030498
Lyapunov	0.2	0.000566	0.003254
PID	0.2	0.060527	0.029383
PID	0.25	0.057305	0.030719

Cuadro 3: Error cuadrático medio en `X` y `Y` - Simulación virtual.

La semejanza en posiciones (Cuadro 4) mide la similitud entre la trayectoria seguida y la deseada. El controlador Lyapunov mostró mayor semejanza en `X` y `Y`, indicando que sigue la trayectoria con mayor fidelidad.

Controlador	Velocidad Máxima (m/s)	Semejanza en X (%)	Semejanza en Y (%)	Semejanza total (%)
Lyapunov	0.07	99.69	98.19	98.94
Lyapunov	0.1	99.68	97.94	98.81
PID	0.1	91.54	91.24	91.39
Lyapunov	0.13	99.67	97.80	98.73
PID	0.13	91.74	91.54	91.64
Lyapunov	0.2	99.21	97.22	98.21
PID	0.2	91.90	91.71	91.81
PID	0.25	92.13	91.53	91.83

Cuadro 4: Semejanza de la carretera - Simulación virtual.

Las estadísticas descriptivas de las velocidades lineal y angular se presentan en los cuadros 5 y 6. El controlador Lyapunov mostró mayor variabilidad en las velocidades, reflejada en coeficientes de variación y curtosis más altos. El controlador PID presentó valores de curtosis elevados en velocidades angulares, indicando presencia de valores atípicos significativos.

Controlador	Velocidad Máxima (m/s)	Media	Desviación Estándar	Coeficiente de Variación	Curtosis	Asimetría
Lyapunov	0.07	0.0502	0.0274	0.55	-0.66	-0.36
Lyapunov	0.1	0.0647	0.0392	0.61	-1.19	-0.46
PID	0.1	0.0558	0.0365	0.65	-1.40	-0.08
Lyapunov	0.13	0.0743	0.0535	0.72	-1.44	-0.01
PID	0.13	0.0779	0.0568	0.73	-1.46	0.03
Lyapunov	0.2	0.1107	0.0870	0.79	-1.56	-0.05
PID	0.2	0.0979	0.0772	0.79	-1.48	0.12
PID	0.25	0.1115	0.0981	0.88	-1.45	0.27

Cuadro 5: Estadística descriptiva velocidad lineal - Simulación virtual.

En cuanto a los rendimientos de tiempo y velocidad (Cuadro 7), el controlador PID se mantuvo consistentemente dentro del límite de velocidad, sugiriendo un control más estable y seguro. El controlador Lyapunov presentó más tiempos con velocidades atípicas y menos tiempo dentro del límite, lo que podría comprometer la estabilidad.

En resumen, el controlador Lyapunov es superior en precisión de trayectoria, mientras que el PID ofrece mayor estabilidad y seguridad al mantenerse dentro del límite de velocidad. Si se prioriza la precisión en la trayectoria, el Lyapunov a 0.07 m/s es recomendable. Si la estabilidad y seguridad son la prioridad, el PID a cualquier velocidad máxima dentro del rango evaluado es más adecuado.

Controlador	Velocidad Máxima (m/s)	Media	Desviación Estándar	Coeficiente de Variación	Curtosis	Asimetría
Lyapunov	0.07	0.0280	0.1500	5.36	20.95	-1.66
Lyapunov	0.1	0.0358	0.1896	5.29	11.88	-0.65
PID	0.1	0.0322	0.1295	4.02	32.73	-3.37
Lyapunov	0.13	0.0411	0.2148	5.23	8.99	-0.08
PID	0.13	0.0427	0.1807	4.24	18.35	-2.35
Lyapunov	0.2	0.0611	0.3077	5.03	4.63	0.65
PID	0.2	0.0507	0.2375	4.69	11.47	-1.84
PID	0.25	0.0652	0.2720	4.17	9.24	-1.39

Cuadro 6: Estadística descriptiva velocidad angular - Simulación virtual.

Controlador	Velocidad Máxima (m/s)	Tiempo Total (s)	Tiempo con Velocidad Lineal Atípica (s)	Tiempo con Velocidad Angular Atípica (s)	Tiempo Detenido (Vel. Lineal 0) (s)	Tiempo Detenido (Vel. Angular 0) (s)	% Dentro del Límite de Velocidad
Lyapunov	0.07	170.6	0.3	2.6	21.6	50.0	71.10
Lyapunov	0.1	132.6	0.0	3.0	21.5	42.9	69.53
PID	0.1	141.9	0.0	2.0	19.2	41.9	100.00
Lyapunov	0.13	115.6	0.0	2.8	21.5	38.5	74.74
PID	0.13	103.2	0.0	1.5	19.3	34.6	100.00
Lyapunov	0.2	77.8	0.0	2.1	21.7	30.1	70.05
PID	0.2	83.4	0.0	1.3	19.3	30.8	100.00
PID	0.25	73.7	0.0	0.9	21.8	28.6	100.00

Cuadro 7: Análisis de tiempos - Simulación virtual.

8.9.2. Análisis descriptivo

El análisis de los errores en las trayectorias de la simulación virtual revela diferencias notables entre los controladores Lyapunov y PID. En la Figura 28 se comparan las posiciones reales y deseadas del vehículo autónomo a diversas velocidades máximas.

Las trayectorias bajo control Lyapunov siguen más de cerca la ruta deseada, especialmente a velocidades bajas como 0.07 m/s y 0.1 m/s, manteniendo el error consistentemente bajo con desviaciones mínimas. Esto se refleja en los menores valores de Error Cuadrático Medio (ECM) en `X` y `Y` presentados en el Cuadro 3, así como en la mayor semejanza total indicada en el Cuadro 4.

En contraste, las trayectorias controladas por PID muestran mayores desviaciones, especialmente a velocidades superiores a 0.13 m/s. Estas desviaciones se evidencian en mayores valores de ECM y menor semejanza total en comparación con Lyapunov. La mayor variabilidad en la adherencia a la ruta deseada sugiere limitaciones del controlador PID a velocidades más altas en este entorno de simulación.

Este análisis destaca la efectividad del controlador Lyapunov en mantener la precisión en el seguimiento de trayectorias, incluso a velocidades moderadas, mientras que el controlador PID es menos preciso a velocidades mayores, lo que podría afectar su desempeño en aplicaciones que requieren alta exactitud en la trayectoria.

El análisis de las velocidades en las trayectorias simuladas, presentado en la Figura 29, muestra diferencias significativas entre los controladores Lyapunov y PID. Los gráficos de posiciones reales versus deseadas, coloreados según la velocidad, indican que el controlador Lyapunov experimenta variaciones notables de velocidad en las curvas, con incrementos de velocidad en estos tramos y discontinuidades en las trayectorias. En contraste, el controlador PID mantiene velocidades más consistentes a lo largo de todo el recorrido.

Los datos del Cuadro 5 respaldan estas observaciones. El controlador Lyapunov presenta una mayor desviación estándar y coeficiente de variación en la velocidad lineal, indicando mayor variabilidad. La asimetría negativa más pronunciada y una curtosis más baja sugieren una distribución de velocidad más dispersa y con tendencia a velocidades más bajas. Por otro lado, el PID muestra una distribución más concentrada y simétrica, reflejando mayor consistencia en la velocidad.

El análisis de tiempos en el Cuadro 7 destaca que el controlador Lyapunov registra tiempos con velocidades lineales y angulares atípicas, y un menor porcentaje de tiempo dentro del límite de velocidad, especialmente a velocidades máximas más altas. En contraste, el PID mantiene un 100 % del tiempo dentro del límite de velocidad, evidenciando su capacidad para seguir las trayectorias con mayor estabilidad y sin presentar velocidades extremas.

Estos resultados indican que, aunque el controlador Lyapunov puede adaptarse a cambios de velocidad necesarios para navegar eficientemente por las curvas, el controlador PID ofrece un desempeño más uniforme y predecible. Esto es ventajoso en aplicaciones que requieren estabilidad y precisión en el control de la velocidad, ya que minimiza las variaciones abruptas y mantiene al vehículo dentro de los límites establecidos.

8.10. Análisis del comportamiento del vehículo autónomo en simulación física

Se empleó el robot móvil diferencial Pololu 3pi+ 32U4 OLED para todas las simulaciones físicas en el ecosistema Robotat, utilizando la plataforma OptiTrack dentro del ecosistema Robotat de la Universidad Del Valle de Guatemala.

8.10.1. Análisis estadístico

Se realizó un análisis comparativo del comportamiento de un vehículo autónomo a escala, probado en ocho ocasiones utilizando los dos tipos de controladores. Cada uno fue evaluado a diferentes velocidades máximas establecidas, y los resultados se analizaron en términos de error cuadrático medio, semejanza en posiciones `X` y `Y`, estadísticas descriptivas de las velocidades lineal y angular, y los tiempos de simulación.

El error cuadrático medio (ECM) para las posiciones en `X` y `Y` se presenta en el Cuadro 8. El controlador Lyapunov mostró menores valores de ECM en comparación con el PID en todas las velocidades, indicando una mayor precisión en el seguimiento de la trayectoria deseada. Esto se refuerza con los resultados de semejanza en posiciones, resumidos en el Cuadro 9, donde el Lyapunov supera al PID en términos de semejanza en `X` y `Y`, evidenciando una mayor fidelidad en la trayectoria.

Controlador	Velocidad Máxima (m/s)	ECM en X	ECM en Y
Lyapunov	0.07	0.000209	0.001289
Lyapunov	0.1	0.000276	0.001762
PID	0.1	0.064632	0.032432
Lyapunov	0.13	0.000314	0.00204
PID	0.13	0.064723	0.029924
Lyapunov	0.2	0.000626	0.004826
PID	0.2	0.06298	0.028521
PID	0.25	0.053383	0.028026

Cuadro 8: Error cuadrático medio en `X` y `Y` - Simulación física.

Controlador	Velocidad Máxima (m/s)	Semejanza en X (%)	Semejanza en Y (%)	Semejanza total (%)
Lyapunov	0.07	99.52	98.25	98.88
Lyapunov	0.1	99.45	97.95	98.7
PID	0.1	91.56	91.21	91.38
Lyapunov	0.13	99.41	97.8	98.6
PID	0.13	91.55	91.56	91.55
Lyapunov	0.2	99.17	96.61	97.89
PID	0.2	91.67	91.75	91.71
PID	0.25	92.33	91.82	92.07

Cuadro 9: Semejanza de la carretera - Simulación física.

Las estadísticas descriptivas para las velocidades lineal y angular se muestran en los Cuadros 10 y 11, respectivamente. El controlador Lyapunov presenta mayores coeficientes de variación y curtosis en varias velocidades, sugiriendo mayor variabilidad en las velocidades. Por su parte, el PID exhibe valores de curtosis extremadamente altos en velocidades angulares, indicando la presencia de valores atípicos significativos. La asimetría negativa en la mayoría de los casos sugiere una tendencia de los valores hacia la izquierda de la media.

En el análisis de rendimientos de tiempo y velocidad (Cuadro 12), el PID se mantiene consistentemente dentro del límite de velocidad, lo que sugiere un control más estable y seguro. El Lyapunov muestra más tiempos con velocidades atípicas y un menor porcentaje de tiempo dentro del límite de velocidad, lo que puede comprometer la estabilidad del vehículo. El tiempo detenido se mantuvo consistentemente similar en todos los casos.

En conclusión, el controlador Lyapunov es preferible cuando la precisión de la trayectoria es crítica, a pesar de su mayor variabilidad en velocidades. El PID es más adecuado en situaciones donde la estabilidad es prioritaria, debido a su capacidad para mantenerse dentro del límite de velocidad. El Lyapunov a una velocidad máxima de 0.07 m/s es el más indicado

Controlador	Velocidad Máxima (m/s)	Media	Desviación Estándar	Coeficiente de Variación	Curtosis	Asimetría
Lyapunov	0.07	0.047	0.0357	0.75	7.84	-0.09
Lyapunov	0.1	0.061	0.0467	0.77	1.99	0.01
PID	0.1	0.056	0.0377	0.67	-1.49	-0.11
Lyapunov	0.13	0.067	0.0553	0.82	0.22	0.34
PID	0.13	0.078	0.0578	0.74	-1.51	0
Lyapunov	0.2	0.107	0.0916	0.85	-1.24	0.16
PID	0.2	0.102	0.0793	0.77	-1.54	0.01
PID	0.25	0.127	0.0997	0.79	-1.56	0.02

Cuadro 10: Estadística descriptiva velocidad lineal - Simulación física.

Controlador	Velocidad Máxima (m/s)	Media	Desviación Estándar	Coeficiente de Variación	Curtosis	Asimetría
Lyapunov	0.07	0.026	0.2706	10.32	10.39	-0.28
Lyapunov	0.1	0.033	0.3274	9.67	5.99	-0.42
PID	0.1	0.033	0.2152	6.4	44.33	-5.35
Lyapunov	0.13	0.038	0.3092	8.18	8.49	-0.59
PID	0.13	0.046	0.2204	4.77	30.21	-3.7
Lyapunov	0.2	0.058	0.607	10.48	3.18	0.84
PID	0.2	0.061	0.2684	4.42	13.36	-1.47
PID	0.25	0.09	0.3748	4.16	5.19	-0.23

Cuadro 11: Estadística descriptiva velocidad angular - Simulación física.

Controlador	Velocidad Máxima (m/s)	Tiempo Total (s)	Tiempo con Velocidad Lineal Atípica (s)	Tiempo con Velocidad Angular Atípica (s)	Tiempo Detenido (Vel. Lineal 0) (s)	Tiempo Detenido (Vel. Angular 0) (s)	% Dentro del Límite de Velocidad
Lyapunov	0.07	177.3	2.7	3.4	21.5	19.4	77.16
Lyapunov	0.1	137.5	1.7	2.6	21.5	19.5	82.55
PID	0.1	133.2	0	1.6	19.3	19.5	100
Lyapunov	0.13	119.2	0.8	2.5	21.7	19.4	84.4
PID	0.13	99	0	1.2	19.3	19.5	100
Lyapunov	0.2	78.2	0	1.6	21.6	19.4	80.69
PID	0.2	79.4	0	1.1	19.3	19.5	100
PID	0.25	62.7	0	1.3	15.7	13.6	100

Cuadro 12: Análisis de tiempos - Simulación física.

si la semejanza en la trayectoria es la prioridad, mientras que el PID es recomendable para mantener estabilidad a cualquier velocidad dentro del rango evaluado. El mejor caso del controlador PID con acercamiento exponencial en términos de precisión y velocidad fue a

una velocidad máxima de 0.13 m/s.

8.10.2. Análisis descriptivo

El análisis de las trayectorias en la simulación física, validado mediante el sistema OptiTrack, reveló diferencias significativas respecto a las simulaciones virtuales, principalmente debido a irregularidades en la superficie de la lona utilizada, como dobleces que afectan el movimiento fluido del vehículo autónomo.

En las pruebas con el controlador Lyapunov, el error se mantuvo bajo a velocidades menores, incrementándose a velocidades más altas (Figura 30). El Cuadro 8 muestra que el error cuadrático medio (ECM) en `X` y `Y` es significativamente menor para Lyapunov en comparación con PID, indicando una mayor precisión en el seguimiento de la trayectoria deseada. La semejanza de la trayectoria con la ruta deseada es mayor con Lyapunov, alcanzando una semejanza total del 98.7 % a 0.1 m/s, frente al 91.38 % del PID (Cuadro 9).

El controlador PID presentó discontinuidades y desviaciones mayores en las curvas, atribuibles a las limitaciones físicas del entorno y a su menor capacidad para manejar variaciones bruscas en la trayectoria. Estas irregularidades se reflejaron en una disminución de la semejanza total del PID a mayores velocidades, evidenciando las limitaciones del controlador en condiciones físicas reales.

En cuanto a las velocidades, el controlador Lyapunov mostró variaciones significativas, especialmente en las curvas y zonas con dobleces en la lona, reflejándose en mayores desviaciones estándar y coeficientes de variación en los datos descriptivos de velocidad lineal (Cuadro 10). Las discontinuidades en la superficie obligaron al controlador a ajustar la velocidad para compensar, resultando en mayor variabilidad.

Por el contrario, el controlador PID mantuvo una velocidad más constante, incluso en las curvas, lo cual se refleja en menores valores de curtosis y asimetría. Su capacidad para mantener velocidades constantes se evidencia en el análisis de tiempos, donde el tiempo total y el porcentaje dentro del límite de velocidad son más altos y consistentes en comparación con Lyapunov (Cuadro 12).

En resumen, el controlador Lyapunov demostró mejor precisión y semejanza a la ruta deseada, siendo más adecuado en entornos con irregularidades y cuando la precisión es prioritaria. Sin embargo, el controlador PID ofreció un rendimiento más estable y controlado en términos de velocidad, minimizando las variaciones y asegurando una adherencia más estricta a los límites establecidos. La implementación física resalta la importancia de elegir el controlador adecuado según las condiciones del entorno y los objetivos del vehículo autónomo.

8.11. Análisis comparativo entre resultados de simulación virtual y física

El análisis comparativo de los resultados en las simulaciones virtual y física reveló diferencias significativas en el desempeño de los controladores PID y Lyapunov para vehículos autónomos a escala. En la simulación virtual, sin irregularidades físicas, el controlador Lyapunov mostró mayor precisión en el seguimiento de la trayectoria deseada, con menores valores de error cuadrático medio (ECM) y mayor semejanza en las posiciones `X` y `Y`. Este controlador mantuvo su superioridad en precisión a distintas velocidades, superando consistentemente al PID.

En cuanto a las velocidades, el controlador Lyapunov presentó mayor variabilidad en la simulación virtual, especialmente en las curvas, reflejándose en mayores desviaciones estándar y coeficientes de variación en las velocidades lineales. El PID mantuvo velocidades más constantes y una adherencia más estricta a los límites establecidos, indicando un control más estable.

En la simulación física, las condiciones del entorno, como irregularidades y dobleces en la superficie, afectaron el comportamiento de los vehículos. A pesar de ello, el controlador Lyapunov continuó mostrando mayor precisión en la trayectoria, aunque con incrementos en la variabilidad de las velocidades debido a las condiciones físicas. Esto se evidencia en los mayores valores de desviación estándar y coeficiente de variación en las velocidades lineales y angulares.

El controlador PID demostró una mejor capacidad para mantener velocidades constantes en presencia de irregularidades físicas, pero su precisión en la trayectoria se vio comprometida. Las discontinuidades y errores en las curvas afectaron su desempeño, aumentando el ECM y reduciendo la semejanza total en las posiciones.

El análisis de tiempos reforzó estas observaciones: en la simulación física, el PID mantuvo un mayor porcentaje del tiempo dentro del límite de velocidad, reflejando un control más estable. El Lyapunov experimentó más tiempos con velocidades atípicas y menos tiempo dentro del límite, lo que puede afectar la estabilidad en condiciones reales.

En resumen, el controlador Lyapunov ofreció mayor precisión en el seguimiento de trayectorias en ambos entornos, aunque con mayor variabilidad en velocidades en la simulación física. El PID, si bien menos preciso en la trayectoria, proporciona un control de velocidad más estable, especialmente en condiciones físicas reales. Este análisis destaca la importancia de equilibrar precisión en trayectoria y estabilidad en velocidad al seleccionar controladores para vehículos autónomos a escala en diferentes condiciones de simulación.

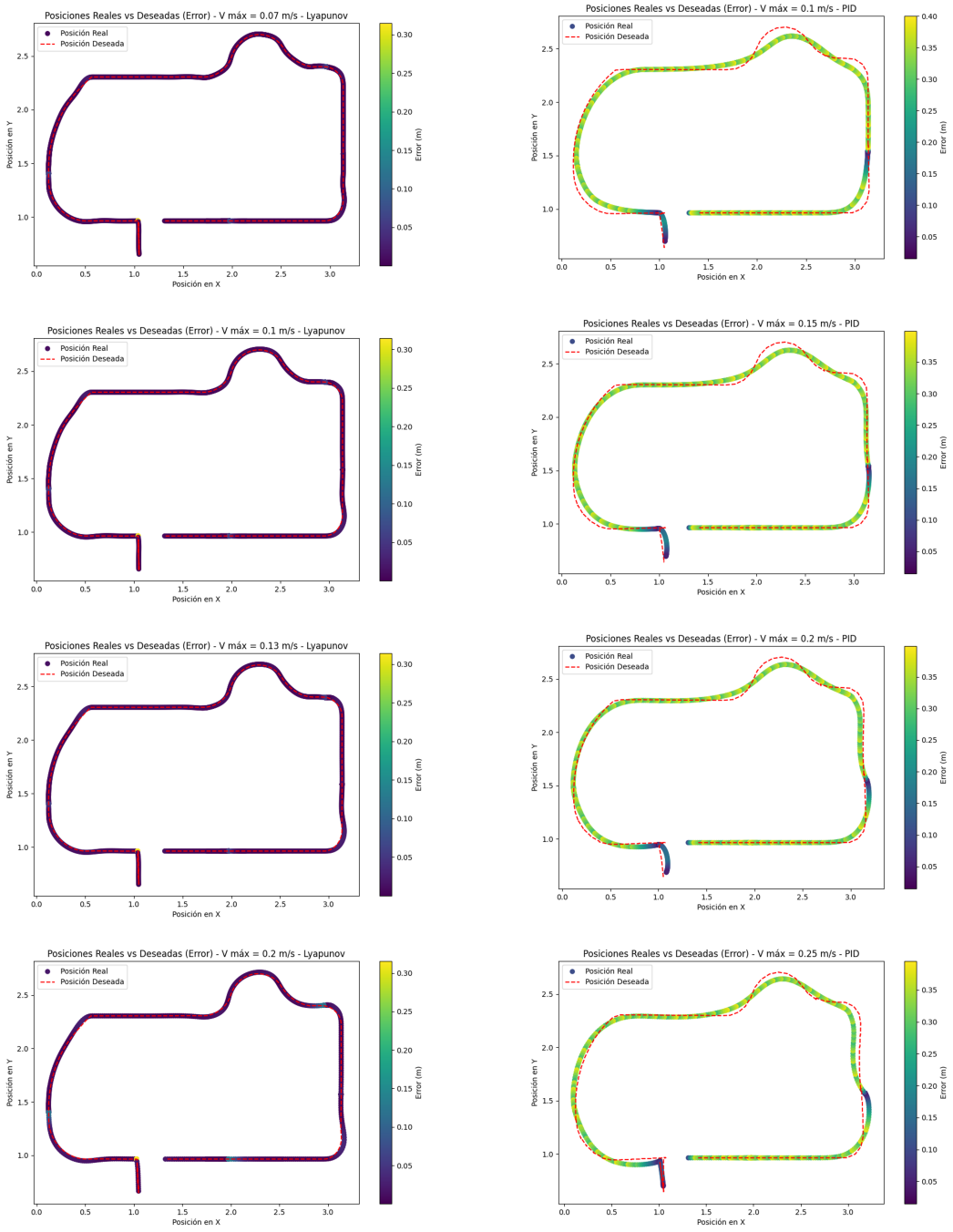


Figura 28: Gráfico de errores en trayectorias - Simulación virtual.

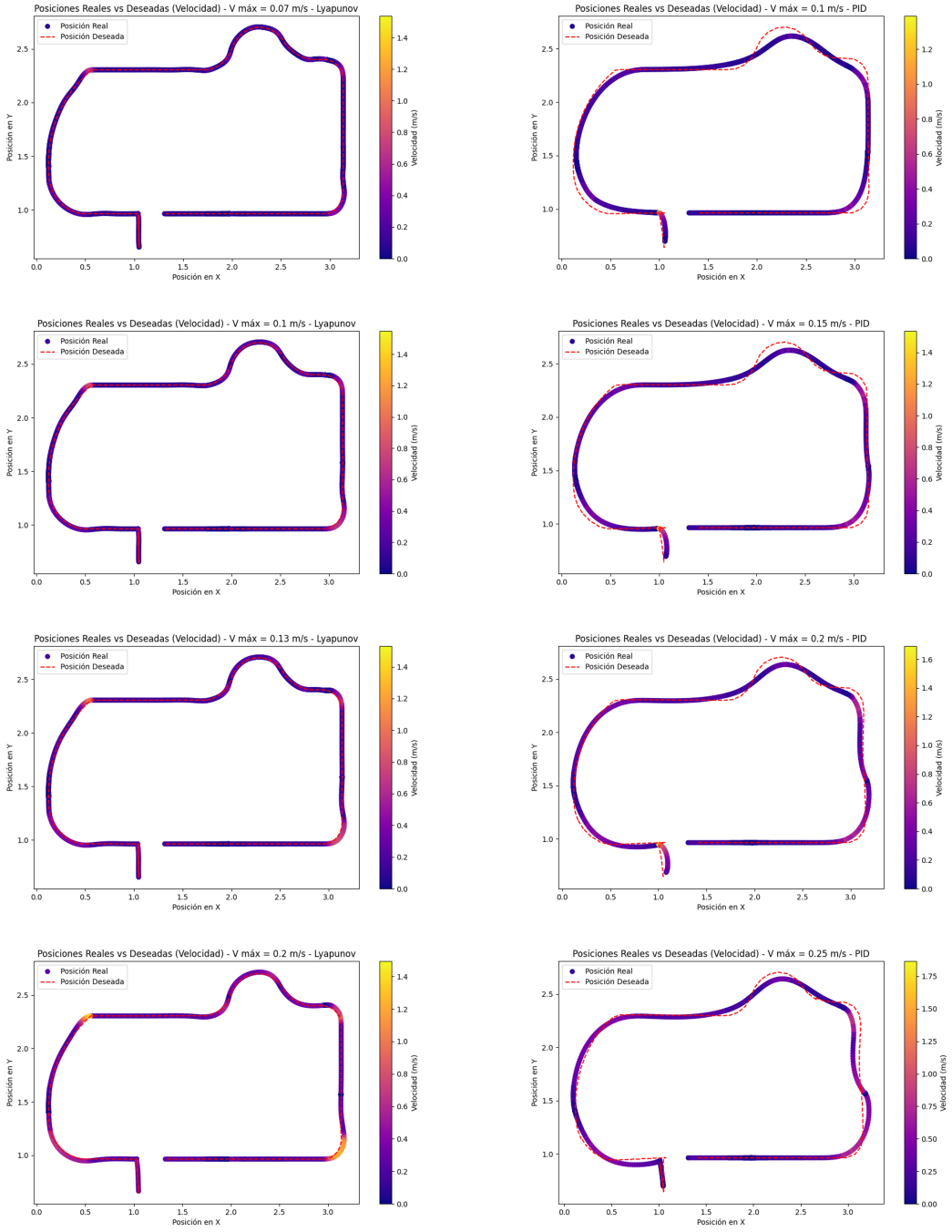


Figura 29: Gráfico de velocidades en trayectorias - Simulación virtual.

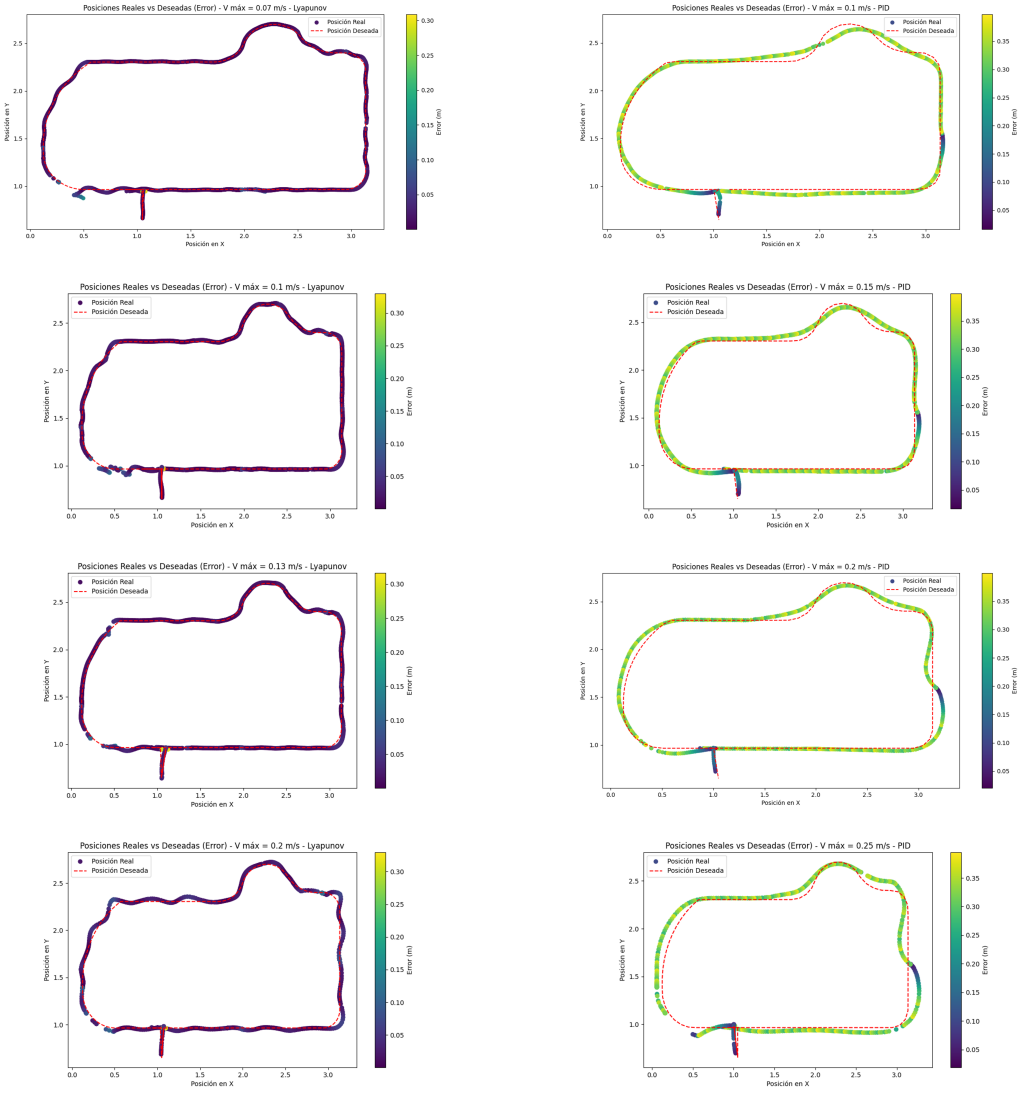


Figura 30: Gráfico de errores en trayectorias - Simulación física.

Figura 31: Gráfico de velocidades en trayectorias - Simulación física.

CAPÍTULO 9

Implementación y evaluación de sistemas de control para múltiples vehículos autónomos en simulación virtual y física

Este capítulo presenta la implementación de un sistema de evasión de obstáculos y el desarrollo de una interfaz gráfica de usuario (GUI) integrada con la clase `CarroAutonomo` para la simulación de vehículos autónomos. Se describe la jerarquía de objetivos y planificación utilizada para gestionar el comportamiento autónomo de los vehículos. Además, se realiza un análisis estadístico y descriptivo del comportamiento de múltiples vehículos, utilizando controladores Lyapunov y PID con acercamiento exponencial, en entornos de simulación virtual y física, destacando las limitaciones encontradas en la implementación física, principalmente los altos tiempos de ejecución.

9.1. Metodología

Se implementó una lógica de evasión de obstáculos para permitir que los vehículos autónomos detecten y eviten colisiones con objetos en su entorno, integrándose en una simulación que evalúa su comportamiento en condiciones controladas. Paralelamente, se desarrolló una interfaz gráfica de usuario (GUI) que facilita la interacción con la simulación, permitiendo configurar y monitorear el comportamiento de los vehículos. Esta interfaz se diseñó para integrar la clase `CarroAutonomo`, responsable de funcionalidades avanzadas como la planificación de rutas y el manejo autónomo.

La jerarquía de objetivos y planificación implementada prioriza criterios como seguridad, confort y eficiencia durante la navegación. Esta jerarquía se refleja en los diferentes niveles de planificación y control del sistema, asegurando que los vehículos actúen de acuerdo con las reglas del entorno simulado y respondan adecuadamente a situaciones como la presencia de obstáculos o cambios en la ruta.

Se realizó un análisis estadístico y descriptivo del comportamiento de múltiples vehículos, tanto en simulaciones virtuales como en pruebas físicas que utilizaron el robot Pololu 3pi+ 32U4 OLED como vehículo dentro de la plataforma OptiTrack en el ecosistema Robotat. Este análisis incluyó evaluaciones de controladores como Lyapunov y PID con acercamiento exponencial, enfocándose en métricas como error cuadrático medio, velocidades y precisión en el seguimiento de trayectorias. Los resultados mostraron diferencias en el desempeño de los controladores y evidenciaron limitaciones en la implementación física, principalmente relacionadas con altos tiempos de ejecución y desafíos en la reproducción fiel de las condiciones simuladas.

9.2. Método de evasión de obstáculos

La funcionalidad de evasión de obstáculos es esencial en la navegación autónoma, permitiendo al vehículo ajustar su trayectoria para evitar colisiones con objetos imprevistos. El sistema detecta obstáculos en la ruta prevista y modifica el camino del vehículo para mantener una ruta segura y eficiente.

El proceso inicia con la detección de proximidad a obstáculos: el vehículo evalúa los puntos futuros de su trayectoria para determinar si alguno está dentro de un área definida como obstáculo y si se encuentra en un espacio con doble carril que permita una evasión segura. Si el vehículo se acerca a un obstáculo, reduce su velocidad proporcionalmente a la cercanía del mismo, permitiendo una mayor capacidad de reacción. Si el obstáculo está muy próximo, el sistema busca el siguiente punto de la ruta fuera del área del obstáculo para evitar la colisión.

Una vez detectado el obstáculo, se genera una ruta alternativa que esquiva el objeto, calculando puntos que forman una trayectoria suave y segura alrededor del mismo. Este cálculo considera distancias y ángulos entre los nuevos puntos para asegurar una maniobra de evasión fluida, sin afectar significativamente el rendimiento del sistema ni interrumpir el desplazamiento del vehículo.

Este método funciona de manera óptima en tramos rectos; en trayectorias curvas, aunque operativo, puede requerir ajustes adicionales debido a la complejidad de la maniobra. Además, la trayectoria generada puede, en ocasiones, pasar ligeramente dentro del área definida como obstáculo, por lo que es crucial que esta área sea suficientemente amplia para evitar riesgos de colisión.

La Figura 32 muestra cómo el sistema genera rutas alternativas para esquivar un obstáculo en diferentes escenarios, ajustando la trayectoria para mantenerla lo más cercana posible al camino original y demostrando la capacidad del sistema para adaptarse y garantizar la seguridad durante el movimiento.

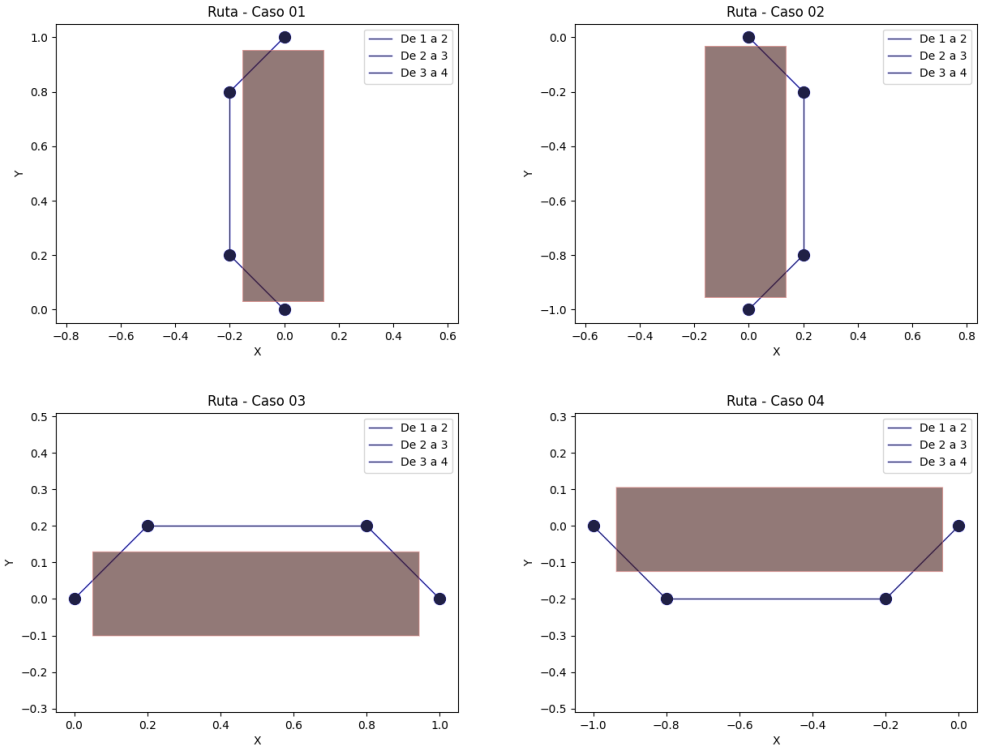


Figura 32: Rutas para esquivar obstáculos.

9.3. Implementación extendida de la clase `CarroAutonomo`

Se ampliaron las funcionalidades de la clase `CarroAutonomo` para lograr una simulación más detallada y realista de vehículos autónomos a escala. La Figura 33 muestra el diagrama de la clase, detallando los métodos y atributos principales. Las mejoras incluyen la capacidad de evitar colisiones entre vehículos, esquivar obstáculos, manejo avanzado de señales de tránsito y un sistema de parqueo automático.

Se incorporaron métodos para detectar la proximidad de otros vehículos y obstáculos, permitiendo la recalculación dinámica de rutas para evitar colisiones mediante el ajuste de velocidad y modificación de la trayectoria. Este control modular permite al vehículo reaccionar de forma autónoma ante diversas situaciones de tráfico.

La funcionalidad de parqueo automático permite al vehículo identificar y maniobrar hacia un espacio de estacionamiento predefinido, realizando una alineación precisa en el espacio designado y seguimiento a dicho punto. La gestión de obstáculos se mejoró con métodos que detectan y reaccionan ante obstáculos en el camino, ajustando la ruta en tiempo real y creando rutas alternativas para garantizar un trayecto seguro.

Además, se implementaron nuevas señales y banderas que proporcionan mayor control sobre el estado del vehículo y sus reacciones al entorno, gestionando situaciones como detenerse en una intersección o reducir la velocidad en zonas de peligro.

Estas mejoras ofrecen una simulación más robusta y flexible, adaptada a condiciones

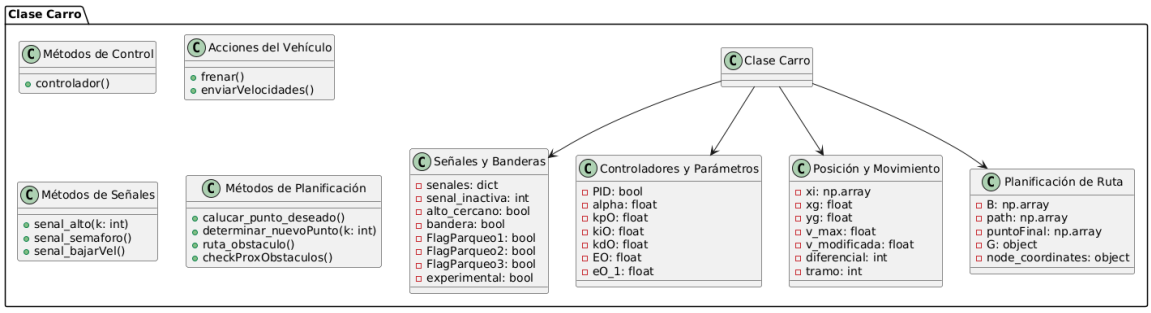


Figura 33: Diagrama de la clase `CarroAutonomo`.

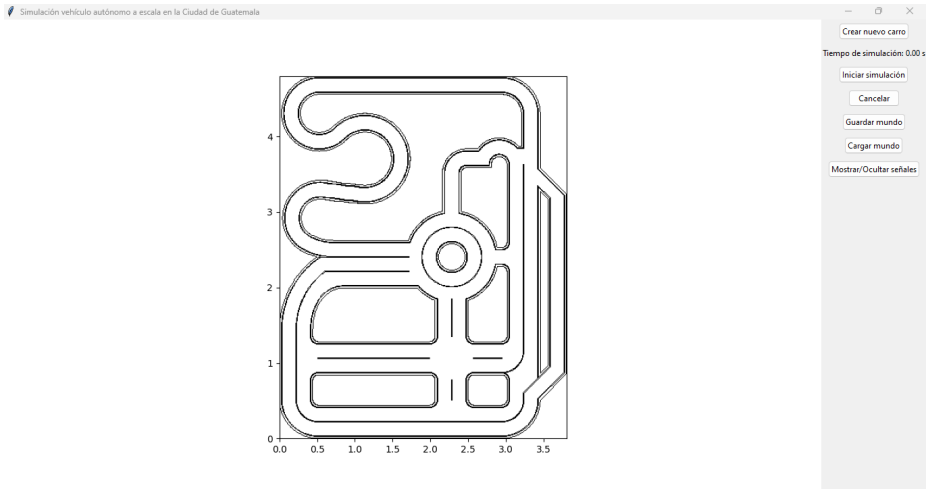


Figura 34: Interfaz gráfica GUI.

dinámicas del entorno y que sirve como base sólida para pruebas avanzadas en entornos simulados.

9.4. Interfaz gráfica de usuario

La interfaz gráfica de usuario (GUI), mostrada en la Figura 34, permite la representación e interacción con la simulación de vehículos autónomos en entornos urbanos a escala. Desarrollada en Python utilizando Tkinter y Matplotlib, esta herramienta proporciona una plataforma intuitiva que facilita la interacción con la simulación, permitiendo crear, configurar y manejar vehículos autónomos en diversos escenarios. La interfaz es un componente clave para la simulación y análisis de sistemas de conducción autónoma por su flexibilidad y accesibilidad.

La GUI integra la capacidad de cargar y gestionar señales de tránsito desde un archivo JSON, lo que proporciona flexibilidad para modificar y actualizar el entorno de manera sencilla. El archivo incluye información detallada de cada señal, como su tipo, ubicación y estado actual. Este enfoque permite una configuración modular de las señales, facilitando la personalización del entorno de simulación.

La funcionalidad de "Crear nuevo carro" permite al usuario introducir nuevos vehículos en la simulación con facilidad. Al activarla, se despliega una ventana de diálogo que solicita parámetros esenciales como el número de identificación del robot, la velocidad máxima, el tipo de simulación (física o virtual), el controlador a utilizar (PID con acercamiento exponencial o Lyapunov), y los puntos inicial y final en el mapa que definen el recorrido del vehículo. Tras configurar estos parámetros, el vehículo se crea mediante la clase `CarroAutonomo` y se almacena en un arreglo global que consolida todos los vehículos presentes en la simulación, facilitando su gestión y monitoreo. Esta estructura de almacenamiento permite la creación dinámica de múltiples vehículos sin necesidad de modificar el código base.

El botón "Iniciar simulación" pone en marcha la simulación una vez que todos los elementos del entorno han sido configurados. Al presionarlo, la simulación se inicia utilizando multi-threading, permitiendo manejar múltiples vehículos simultáneamente y optimizando el rendimiento del sistema. Durante la simulación, un cronómetro en la interfaz muestra el tiempo transcurrido, y se ejecutan los métodos encargados de las acciones de cada vehículo, asegurando que sigan sus trayectorias y respondan adecuadamente a las condiciones del entorno.

El botón "Cancelar" permite detener la simulación en cualquier momento, ofreciendo control preciso sobre el proceso. Al activarlo, el cronómetro y los vehículos se detienen de inmediato, permitiendo analizar los resultados obtenidos hasta ese punto o realizar ajustes en el entorno. Esta capacidad contribuye a una mayor flexibilidad y control en el desarrollo y pruebas de los escenarios simulados.

La interfaz incluye funciones para gestionar el entorno de simulación: "Guardar mundo" y "Cargar mundo". La opción de "Guardar mundo" almacena todas las variables críticas de la simulación en un archivo JSON, conservando configuraciones complejas para ser replicadas en futuras simulaciones. La función de "Cargar mundo" permite recuperar estas configuraciones, restaurando automáticamente todos los vehículos y sus parámetros en el entorno simulado. Este sistema proporciona flexibilidad significativa al gestionar múltiples escenarios y ajustarlos según las necesidades del usuario.

El botón "Mostrar/Ocultar señales" permite controlar la visibilidad de las señales de tránsito en la simulación. Esta funcionalidad es útil para analizar distintos aspectos del entorno sin distracciones visuales, permitiendo al usuario enfocar la simulación según sus necesidades.

9.5. Aplicación de la jerarquía de objetivos y planificación en el vehículo autónomo a escala

9.5.1. Jerarquía de objetivos

En la implementación del vehículo autónomo a escala, la jerarquía de objetivos se integra directamente en la lógica programada en la clase `CarroAutonomo`, asegurando que las decisiones y acciones del vehículo se alineen con las prioridades establecidas: seguridad, legalidad, seguridad percibida, confort y eficiencia.

1. Seguridad: Es el pilar fundamental en la operación del vehículo. Se maneja a través de métodos de evasión de obstáculos y colisiones con otros vehículos mediante detección de proximidad y disminución de velocidad. La lógica de frenar cuando se detecta un peligro inmediato refuerza esta prioridad. Los resultados muestran que el controlador basado en Lyapunov cumple mejor este objetivo por su mayor precisión en el seguimiento de la ruta.

2. Legalidad: Se logra mediante el cumplimiento de señales de tráfico y reglas de la vía, implementado en los métodos de señal de alto, semáforo y señal de bajar velocidad. Estos métodos monitorean el entorno para asegurar que el vehículo respete dichas señales, ajustando su comportamiento en consecuencia.

3. Seguridad percibida: Implícita en la suavidad de las maniobras y la evitación de situaciones peligrosas que podrían causar ansiedad en los ocupantes. La planificación del comportamiento y el control de la velocidad para evitar frenadas bruscas contribuyen a este objetivo. El control PID con acercamiento exponencial se consideró óptimo aquí debido a las rutas más suavizadas y cumplimiento de la velocidad establecida.

4. Confort: Se maneja asegurando que el vehículo no realice cambios bruscos en velocidad o dirección, gestionado a través de la planificación local que calcula trayectorias suaves y ajusta la velocidad para una conducción más agradable.

5. Eficiencia: Considerado al final de la jerarquía, asegurando que, una vez cumplidos los objetivos anteriores, el vehículo tome la ruta más eficiente hacia su destino. La planificación global y el ajuste dinámico de la velocidad y trayectoria garantizan una operación eficiente sin comprometer seguridad o confort.

9.5.2. Jerarquía de planificación

La jerarquía de planificación del vehículo autónomo está estructurada en varios niveles, cada uno con un rol específico en la navegación y control del vehículo, como se muestra en la Figura 35.

1. Planificación global: Implementada en la inicialización de la ruta ideal en la clase `CarroAutonomo`. Determina la ruta general desde la posición inicial hasta el destino final, calculando puntos clave que el vehículo debe alcanzar. Esta ruta proporciona una guía general y no se actualiza constantemente.

2. Planificación del comportamiento: Manejada en el método principal, donde el vehículo adapta la ruta global a las circunstancias inmediatas. Decide maniobras necesarias para evitar colisiones, ajustar la velocidad según señales de tráfico y seguir la trayectoria establecida, tomando decisiones en tiempo real.

3. Planificación local: Encargada de calcular la trayectoria precisa en el corto plazo. Implementada en los métodos de parqueo automático y evitar obstáculos, ajusta la trayectoria para evitar colisiones y cumplir con las condiciones de seguridad y legalidad. Es altamente dinámica y se actualiza con cada iteración de la simulación.

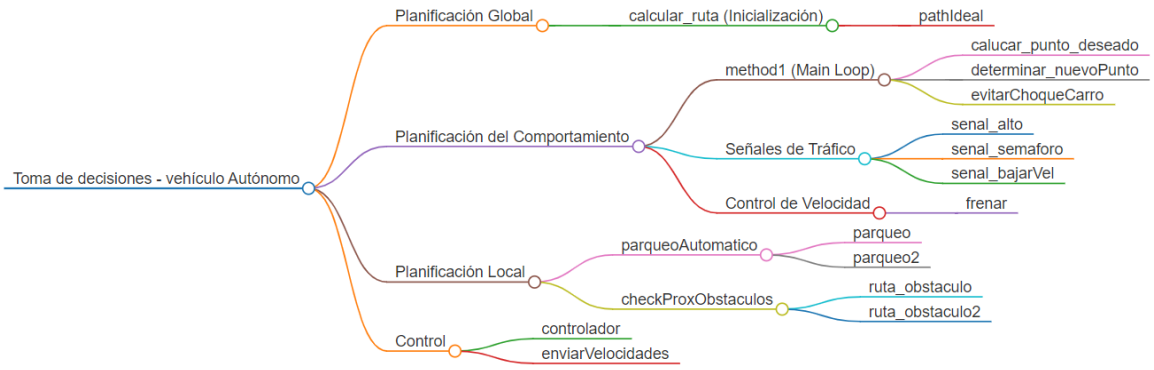


Figura 35: Árbol de jerarquía de planificación.

4. Control: Responsable de ejecutar la trayectoria definida por la planificación local. Implementado en el método del controlador, traduce la trayectoria en comandos específicos para guiar el movimiento del vehículo, convirtiendo la velocidad angular y lineal general en velocidades angulares de cada rueda. Se actualiza continuamente para asegurar un seguimiento preciso y seguro de la trayectoria.

Esta estructura jerárquica asegura que el vehículo autónomo navegue de manera eficiente y segura, adaptándose a las condiciones cambiantes del entorno y garantizando una conducción cómoda y legal en todo momento.

9.6. Análisis del comportamiento de múltiples vehículos autónomos con controlador Lyapunov en simulación virtual

9.6.1. Análisis estadístico

Se analizó el comportamiento de cuatro vehículos autónomos en una simulación virtual con el controlador de Lyapunov a una velocidad de 0.07 m/s. Se consideraron parámetros como el error cuadrático medio (ECM) en posiciones `X` y `Y`, la semejanza de las posiciones, estadísticas descriptivas de velocidad lineal y angular, y los tiempos de simulación. A continuación, se presentan los resultados obtenidos.

El Cuadro 13 muestra que el ECM en `X` y `Y` es bajo para todos los vehículos, indicando alta precisión en las trayectorias seguidas. La semejanza en posiciones, detallada en el Cuadro 14, supera el 99 % en la mayoría de los casos, lo que sugiere un excelente seguimiento de la trayectoria deseada.

Las estadísticas descriptivas de las velocidades lineales (Cuadro 15) muestran coeficientes de variación bajos, indicando consistencia en las velocidades durante la simulación. Las distribuciones presentan asimetría negativa, lo que sugiere una concentración de valores por debajo de la media.

En contraste, las velocidades angular presentan mayor variabilidad (Cuadro 16), con co-

Vehículo	ECM en X	ECM en Y	ECM General
carro1	8.30e-05	0.000614	0.000349
carro2	5.00e-05	0.000100	0.000075
carro3	4.90e-05	0.000363	0.000206
carro4	6.20e-05	0.000070	0.000065
General	6.10e-05	0.000287	0.000174

Cuadro 13: Error cuadrático medio en `X` y `Y` - Simulación múltiple de vehículos.

Vehículo	Semejanza en X (%)	Semejanza en Y (%)	Semejanza total (%)
carro1	99.696681	98.791259	99.243970
carro2	99.765440	99.732039	99.748740
carro3	99.781399	99.562737	99.672068
carro4	99.754364	99.791141	99.772752
General	99.749471	99.469294	99.609383

Cuadro 14: Semejanza de posiciones `X`, `Y` y total - Simulación múltiple de vehículos.

Vehículo	Media	Desviación	Mínimo	Máximo	CV	Asimetría
carro1	0.0499	0.0259	0.0000	0.1457	0.519	-0.424
carro2	0.0547	0.0284	0.0000	0.1455	0.519	-0.506
carro3	0.0526	0.0173	0.0000	0.1459	0.329	-1.566
carro4	0.0526	0.0197	-0.0603	0.1457	0.375	-0.946
General	0.0526	0.0221	-0.0603	0.1459	0.420	-0.763

Cuadro 15: Estadísticas descriptivas de la velocidad lineal - Simulación múltiple de vehículos.

eficientes de variación elevados y asimetrías negativas significativas, especialmente en `carro1` y `carro3`. Esto indica fluctuaciones más amplias en la velocidad angular, posiblemente debido a maniobras de giro o ajustes en la trayectoria.

Vehículo	Media	Desviación	Mínimo	Máximo	CV	Asimetría
carro1	0.0269	0.1757	-3.0498	0.6859	6.537	-6.513
carro2	0.0217	0.2159	-1.5632	0.9221	9.962	-1.203
carro3	-0.0044	0.1505	-3.0119	0.6534	-34.288	-5.222
carro4	-0.0036	0.1668	-1.7865	0.6534	-46.364	-3.221
General	0.0057	0.1746	-3.0498	0.9221	30.659	-3.401

Cuadro 16: Estadísticas descriptivas de la velocidad angular - Simulación múltiple de vehículos.

El Cuadro 17 resume los tiempos de simulación y porcentajes relevantes. Se observa que `carro3` y `carro4` pasaron más tiempo en movimiento y alcanzaron mayores porcentajes de tiempo a velocidad máxima, en comparación con los otros vehículos.

Los vehículos autónomos mostraron trayectorias precisas y consistentes. La mayor variabilidad en la velocidad angular sugiere ajustes dinámicos durante la navegación. Los vehículos con menos tiempo en reposo alcanzaron mayores velocidades máximas, reflejando un comportamiento eficiente dentro de la simulación. Las diferencias entre vehículos fueron bajas, evidenciando un desempeño uniforme del controlador Lyapunov en entornos virtuales.

Vehículo	Tiempo Total (s)	% Tiempo Quieto Lineal	% Tiempo Quieto Angular	% Tiempo Quieto Promedio	% Tiempo Vel Máx
carro1	176.600	11.38	13.65	10.82	76.84
carro2	224.100	10.08	10.08	10.08	56.98
carro3	355.800	5.65	12.68	5.37	98.57
carro4	401.400	5.51	6.93	5.51	90.43
General	289.475	8.16	10.83	7.94	80.71

Cuadro 17: Tiempos y porcentajes en simulación múltiple de vehículos.

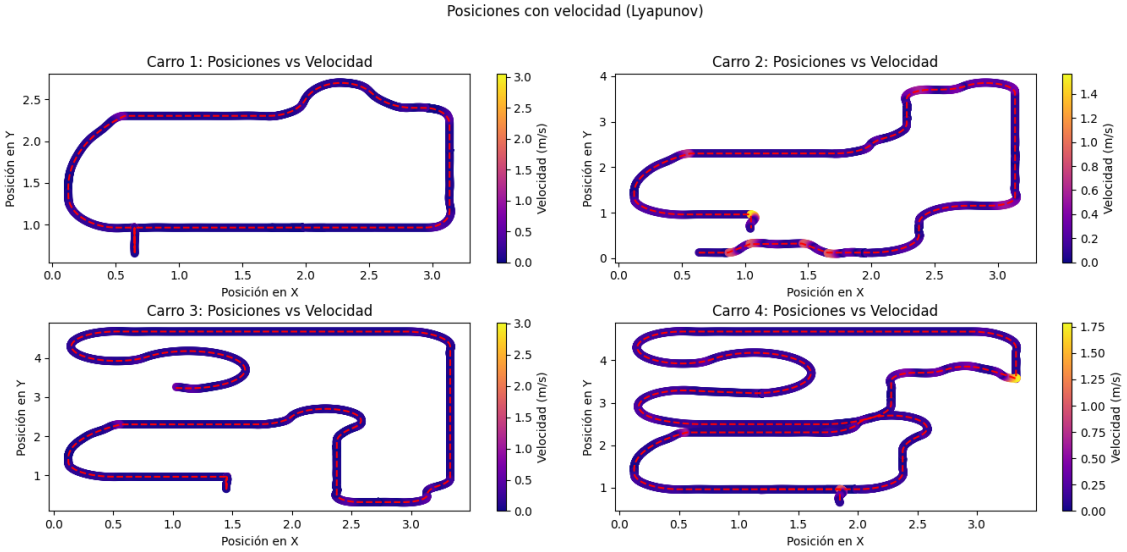


Figura 36: Posiciones de los vehículos con relación a la velocidad (m/s) - Controlador Lyapunov.

9.6.2. Análisis descriptivo

Se analizaron las trayectorias y el desempeño de cuatro vehículos autónomos en simulación virtual utilizando un controlador Lyapunov con una velocidad máxima de 0.07 m/s. La Figura 36 muestra la trayectoria ideal (línea punteada roja) y las posiciones de cada vehículo, con una escala de color que representa la velocidad en cada punto. Los resultados indican una alta precisión en el seguimiento de la trayectoria ideal, con variaciones mínimas de velocidad a lo largo del recorrido, lo que demuestra consistencia en el control de velocidad. Asimismo, no se presentaron colisiones entre vehículos y el `carro2` fue capaz de evitar un obstáculo adecuadamente.

En la Figura 37, la escala de color ilustra el error de seguimiento en cada posición. Los vehículos presentan errores bajos en la mayoría de los puntos, confirmando la capacidad del controlador para mantener la trayectoria deseada con precisión. Las diferencias de error entre los vehículos son mínimas, lo cual sugiere un desempeño homogéneo en el control de posición y velocidad, evidenciando la efectividad del controlador Lyapunov en minimizar desviaciones.

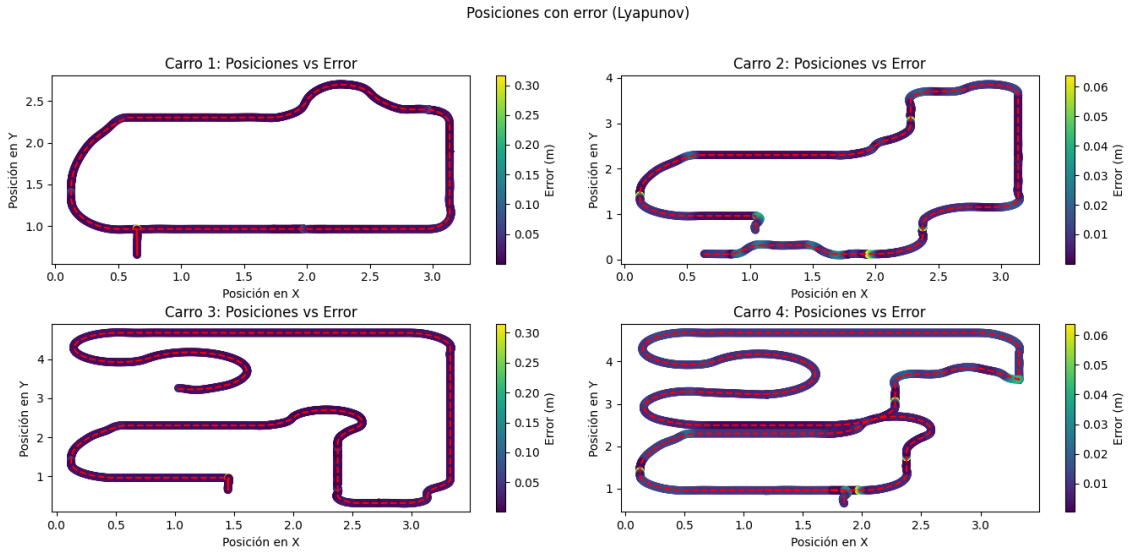


Figura 37: Posiciones de los vehículos con relación al error (m) - Controlador Lyapunov.

9.7. Análisis del comportamiento de múltiples vehículos autónomos con controlador PID con acercamiento exponencial en simulación virtual

9.7.1. Análisis estadístico

Se analizó estadísticamente el comportamiento de cuatro vehículos autónomos en una simulación virtual utilizando el controlador PID con acercamiento exponencial a una velocidad de 0.13 m/s. Se evaluaron métricas como el error cuadrático medio (ECM) en posiciones `X` y `Y`, la semejanza en las posiciones, estadísticas descriptivas de las velocidades lineal y angular, y los tiempos de simulación. Los valores promedios se incluyen como referencia general.

El Cuadro 18 muestra que los valores de ECM son más elevados en comparación con los obtenidos bajo el controlador Lyapunov, indicando una menor precisión en las trayectorias seguidas por los vehículos con PID.

Vehículo	ECM en X	ECM en Y	ECM General
carro1	0.063748	0.027465	0.045607
carro2	0.032449	0.051754	0.042102
carro3	0.038795	0.058205	0.048500
carro4	0.055762	0.042480	0.049121
General	0.047689	0.044976	0.046333

Cuadro 18: Error cuadrático medio en `X` y `Y` - Simulación múltiple de vehículos con PID.

La semejanza en posiciones `X` y `Y`, presentada en el Cuadro 19, es generalmente menor que con el controlador Lyapunov, aunque aún supera el 91 %, lo que indica un seguimiento aceptable de la trayectoria.

Vehículo	Semejanza en X (%)	Semejanza en Y (%)	Semejanza total (%)
carro1	91.614649	91.907482	91.761065
carro2	94.017358	93.896709	93.957033
carro3	93.848718	94.466588	94.157653
carro4	92.625219	94.879956	93.752588
General	93.026486	93.787684	93.407085

Cuadro 19: Semejanza de posiciones `X`, `Y` y total - Simulación múltiple de vehículos con PID.

Las estadísticas descriptivas de las velocidades lineales, en el Cuadro 20, muestran un coeficiente de variación (CV) moderado, indicando cierta variabilidad en las velocidades en comparación con el controlador Lyapunov. Las distribuciones presentan asimetría negativa, sugiriendo una concentración de valores menores que la media.

Vehículo	Media	Desviación	Mínimo	Máximo	CV	Asimetría
carro1	0.068531	0.048237	0.0000	0.1299	0.704	0.035
carro2	0.069858	0.046890	0.0000	0.1299	0.671	-0.059
carro3	0.088912	0.046532	0.0000	0.1299	0.523	-0.716
carro4	0.083546	0.046456	0.0000	0.1299	0.556	-0.510
General	0.079335	0.047631	0.0000	0.1299	0.600	-0.356

Cuadro 20: Estadísticas descriptivas de la velocidad lineal - Simulación múltiple de vehículos con PID.

En cuanto a las velocidades angulares, el Cuadro 21 muestra una mayor variabilidad, con coeficientes de variación elevados y asimetrías negativas significativas, especialmente en `carro1` y `carro3`, lo que indica la presencia de valores atípicos y fluctuaciones en la velocidad angular.

Vehículo	Media	Desviación	Mínimo	Máximo	CV	Asimetría
carro1	0.040630	0.158153	-1.9395	0.7156	3.893	-2.992
carro2	0.030287	0.205833	-2.5266	1.4972	6.796	-1.628
carro3	-0.007635	0.200660	-2.6777	0.6291	-26.282	-3.120
carro4	-0.006401	0.219044	-2.6905	0.6916	-34.222	-3.794
General	0.009674	0.202635	-2.6905	1.4972	20.946	-3.062

Cuadro 21: Estadísticas descriptivas de la velocidad angular - Simulación múltiple de vehículos con PID.

El Cuadro 22 resume los tiempos de simulación y porcentajes relevantes. Se observa que los vehículos `carro1` y `carro2` pasaron más tiempo en reposo, y que todos los vehículos estuvieron el 100 % del tiempo dentro del límite de velocidad, lo que sugiere un control estable aunque menos preciso.

En resumen, el controlador PID mostró una mayor variabilidad y menor precisión en el seguimiento de trayectorias en comparación con el controlador Lyapunov. Aunque los vehículos mantuvieron un comportamiento estable y seguro, reflejado en el 100 % de tiempo dentro del límite de velocidad, la precisión fue menor, como lo indican los mayores valores de ECM y menor semejanza en las trayectorias.

Vehículo	Tiempo Total (s)	% Tiempo Quieto Lineal	% Tiempo Quieto Angular	% Tiempo Quieto Promedio	% Tiempo Vel Máx
carro1	118.500	13.67	19.24	13.67	100.0
carro2	159.200	13.88	13.88	13.88	100.0
carro3	192.000	9.95	9.95	9.95	100.0
carro4	225.600	9.80	9.93	9.80	100.0
General	173.825	11.82	13.25	11.82	100.0

Cuadro 22: Tiempos y porcentajes en simulación múltiple de vehículos con PID.

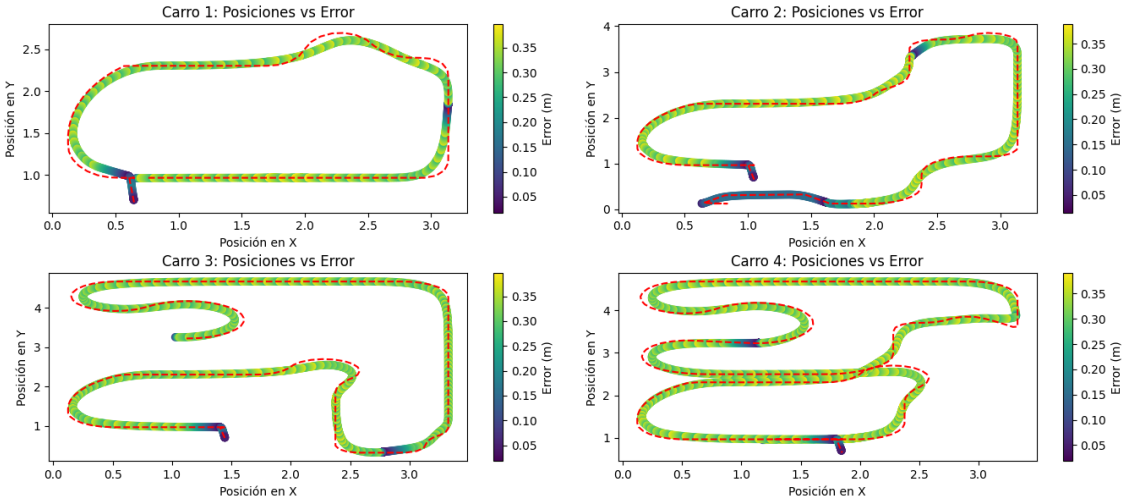


Figura 38: Posiciones de los vehículos con relación al error (m) - Controlador PID.

9.7.2. Análisis descriptivo

La Figura 38 muestra el seguimiento de los vehículos autónomos a la trayectoria ideal, indicada por la línea punteada roja. Aunque el controlador PID no logra mantener completamente la precisión de la trayectoria, los vehículos siguen la ruta deseada de manera relativamente consistente, con errores controlados y estables. La homogeneidad en el seguimiento entre los distintos vehículos refleja un desempeño uniforme en términos de precisión relativa, aunque con ligeras desviaciones en comparación con la trayectoria ideal. De igual forma, no se registraron colisiones entre vehículos, y el `carro2` logró esquivar un obstáculo de manera efectiva.

En la Figura 39, se observa que el controlador PID mantiene una velocidad prácticamente constante en todo el recorrido. Las variaciones en velocidad son mínimas y ocurren principalmente en tramos con discontinuidades en la trayectoria, donde se observan aumentos insignificantes. Este comportamiento indica que, aunque el PID no mantiene la misma precisión en posición que el controlador Lyapunov, sí logra una gestión estable y constante de la velocidad, contribuyendo a un movimiento fluido y controlado a lo largo del recorrido.

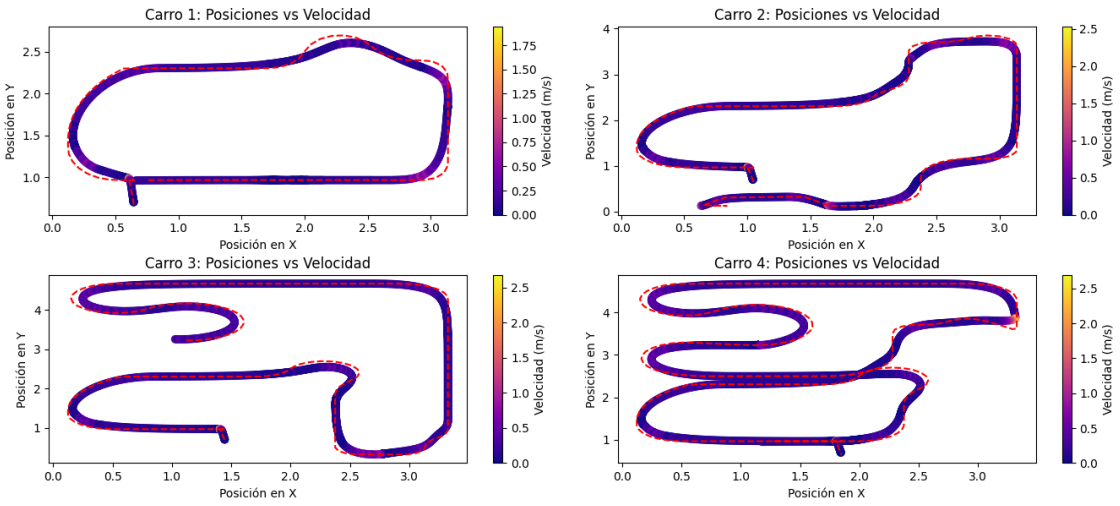


Figura 39: Posiciones de los vehículos con relación a la velocidad (m/s) - Controlador PID.

9.8. Análisis del comportamiento de vehículos autónomos con controlador Lyapunov en simulación física

9.8.1. Análisis estadístico

Se analizó el comportamiento de dos vehículos autónomos en una simulación física utilizando el controlador Lyapunov a una velocidad de 0.07 m/s. Se evaluaron el error cuadrático medio (ECM) en las posiciones `X` y `Y`, la semejanza de las posiciones, estadísticas descriptivas de las velocidades lineal y angular, y los tiempos de simulación. Los valores generales representan el promedio de los vehículos analizados.

El Cuadro 23 muestra que el ECM en `X` y `Y` es bajo para ambos vehículos, indicando alta precisión en las trayectorias seguidas.

Vehículo	ECM en X	ECM en Y	ECM General
carro1	0.000130	0.000643	0.000387
carro2	0.000095	0.000604	0.000350
General	0.000112	0.000624	0.000369

Cuadro 23: Error cuadrático medio en `X` y `Y` - Simulación física de vehículos.

La semejanza en posiciones `X` y `Y`, presentada en el Cuadro 24, es superior al 99 % para ambos vehículos, lo que indica un excelente seguimiento de la trayectoria.

Vehículo	Semejanza en X (%)	Semejanza en Y (%)	Semejanza total (%)
carro1	99.621228	98.763012	99.192120
carro2	99.676890	98.800478	99.238684
General	99.649059	98.781745	99.215402

Cuadro 24: Semejanza de posiciones `X`, `Y` y total - Simulación física de vehículos.

Las estadísticas descriptivas de las velocidades lineales, en el Cuadro 25, muestran un coeficiente de variación (CV) bajo, indicando consistencia en las velocidades. La asimetría y curtosis sugieren distribuciones generalmente simétricas.

Vehículo	Media	Desviación	Mínimo	Máximo	CV	Asimetría
carro1	0.0466	0.0292	-0.0293	0.1468	0.625	0.099
carro2	0.0419	0.0228	-0.0270	0.1424	0.544	-0.161
General	0.0440	0.0259	-0.0293	0.1468	0.588	0.065

Cuadro 25: Estadísticas descriptivas de la velocidad lineal - Simulación física de vehículos.

El Cuadro 26 presenta las estadísticas de la velocidad angular. Los CV más altos indican mayor dispersión en las velocidades angulares. Los valores elevados de curtosis y la asimetría negativa sugieren la presencia de valores atípicos y una concentración hacia la izquierda de la media.

Vehículo	Media	Desviación	Mínimo	Máximo	CV	Asimetría
carro1	0.0282	0.2878	-3.4363	0.9248	10.205	-3.719
carro2	0.0196	0.2508	-3.3061	0.8668	12.815	-4.681
General	0.0233	0.2675	-3.4363	0.9248	11.470	-4.185

Cuadro 26: Estadísticas descriptivas de la velocidad angular - Simulación física de vehículos.

El Cuadro 27 resume los tiempos de simulación y porcentajes relevantes. El `carro2` pasó más tiempo en movimiento y alcanzó un mayor porcentaje de tiempo a la velocidad máxima en comparación con el `carro1`.

Vehículo	Tiempo Total (s)	% Tiempo Quieto Lineal	% Tiempo Quieto Angular	% Tiempo Quieto Promedio	% Tiempo Vel Máx
carro1	163.90	12.20	11.65	11.65	76.45
carro2	213.20	9.71	9.24	9.24	95.73
General	188.55	10.96	10.45	10.45	86.09

Cuadro 27: Tiempos y porcentajes en simulación física de vehículos.

En resumen, ambos vehículos lograron trayectorias precisas con baja variabilidad en las posiciones `X` y `Y`, evidenciada por el bajo ECM. Las velocidades lineales fueron consistentes, mientras que las velocidades angulares mostraron mayor variabilidad. El `carro2` tuvo un mejor rendimiento en términos de tiempo a velocidad máxima. Estos resultados reflejan un comportamiento eficiente de los vehículos autónomos en la simulación física utilizando el controlador Lyapunov.

9.8.2. Análisis descriptivo

La Figura 41 muestra la precisión en el seguimiento de la trayectoria ideal por parte de los vehículos en la simulación física con el controlador Lyapunov. Ambos vehículos mantienen un bajo nivel de error en la mayoría de la trayectoria, demostrando una alta precisión en

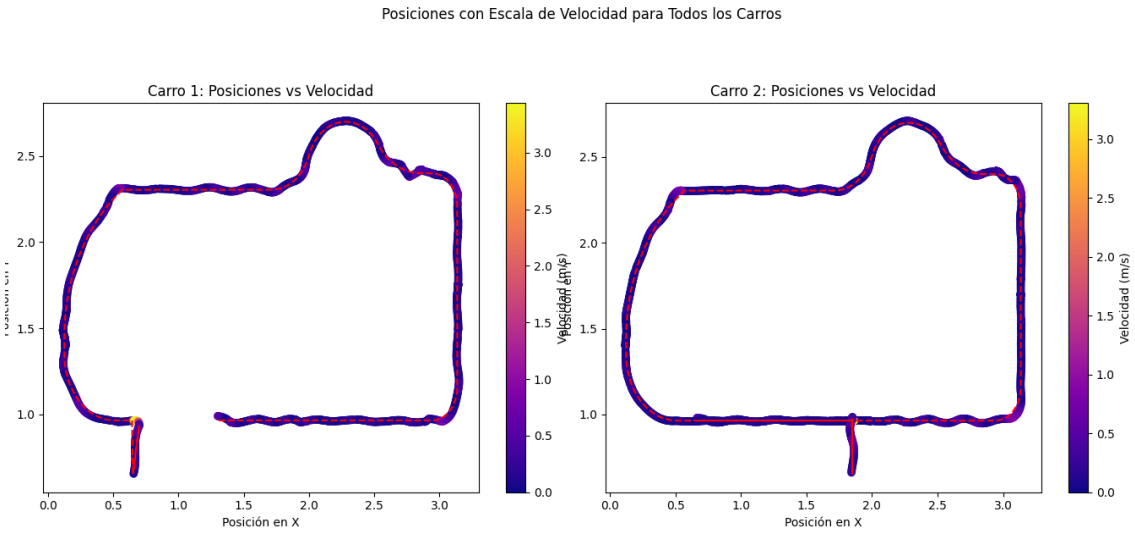


Figura 40: Posiciones de los vehículos con relación a la velocidad (m/s) - Controlador Lyapunov en simulación física.

el cumplimiento de la ruta deseada. La consistencia en el error a lo largo de la trayectoria indica que el controlador Lyapunov logra mantener la ruta con desviaciones mínimas entre los distintos vehículos, evidenciando un control homogéneo, incluso en simulación física. Además, no se produjeron colisiones entre vehículos, lo que evidencia que, a pesar de las irregularidades en la superficie causadas por la lona, el método para evitar colisiones sigue siendo efectivo.

En la Figura 40, se observa el comportamiento de la velocidad a lo largo del recorrido. A diferencia del controlador PID, el controlador Lyapunov presenta mayores variaciones en las velocidades, especialmente en tramos curvos o con discontinuidades en la trayectoria. Sin embargo, estas variaciones en la velocidad permiten al controlador ajustar dinámicamente la posición del vehículo para mantener la precisión en la trayectoria, aunque a costa de una menor estabilidad en la velocidad.

9.9. Análisis del comportamiento de múltiples vehículos autónomos con controlador PID con acercamiento exponencial en simulación física

9.9.1. Análisis estadístico

Se analizó el comportamiento de varios vehículos autónomos en una simulación física utilizando el controlador PID con acercamiento exponencial a una velocidad máxima de 0.13 m/s. Se evaluaron el error cuadrático medio (ECM), la semejanza de las posiciones en `X` y `Y`, estadísticas descriptivas de las velocidades lineal y angular, y los tiempos de simulación. Los valores generales son promedios de los datos de todos los vehículos.

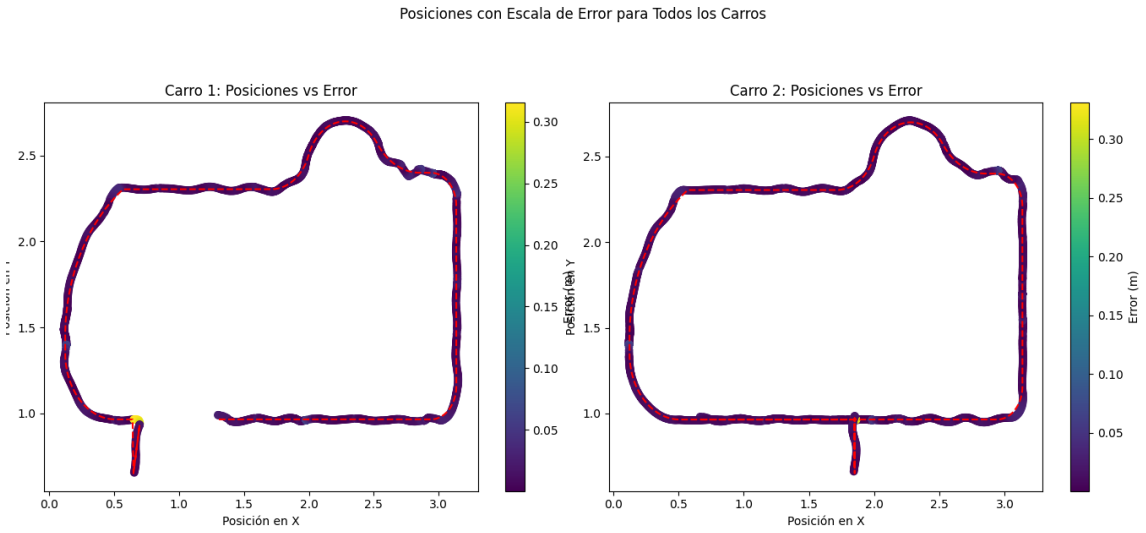


Figura 41: Posiciones de los vehículos con relación al error (m) - Controlador Lyapunov en simulación física.

El Cuadro 28 muestra que el ECM es más alto en comparación con otros controladores, indicando menor precisión en el seguimiento de las trayectorias. La semejanza en posiciones `X` y `Y`, presentada en el Cuadro 29, es alta pero inferior al 93 %, reflejando menor precisión en comparación con otros controladores.

Las estadísticas descriptivas de las velocidades lineales, en el Cuadro 30, muestran una variabilidad moderada, con coeficientes de variación (CV) más elevados. Las distribuciones son ligeramente asimétricas hacia la derecha, indicando la presencia de algunos valores más altos que la media. En cuanto a las velocidades angulares, el Cuadro 31 indica alta variabilidad, con CV extremadamente altos, especialmente para el `carro1`. Los valores significativos de curtosis y asimetría sugieren la presencia de valores atípicos y una distribución altamente asimétrica.

El Cuadro 32 resume los tiempos de simulación y porcentajes relevantes. El `carro2` y `carro3` pasaron más tiempo en reposo en comparación con el `carro1`, sugiriendo un comportamiento menos eficiente. Todos los vehículos se mantuvieron en la velocidad máxima establecida durante el 100 % del tiempo.

En resumen, el controlador PID con acercamiento exponencial muestra un comportamiento estable en la simulación física, pero con menor precisión en las trayectorias en comparación con otros controladores como el Lyapunov. Aunque los vehículos operaron a velocidad máxima, la alta variabilidad en las velocidades angulares indica anomalías que pueden afectar la eficiencia general del control.

9.9.2. Análisis descriptivo

La Figura 42 muestra el seguimiento de los vehículos respecto a la trayectoria ideal en la simulación física con el controlador PID. Los vehículos presentan mayores errores en compa-

Vehículo	ECM en X	ECM en Y	ECM General
carro1	0.052518	0.032012	0.042265
carro2	0.054613	0.032718	0.043666
carro3	0.059214	0.028305	0.043759
General	0.055448	0.031012	0.043230

Cuadro 28: Error cuadrático medio en `X` y `Y` - Simulación física con PID.

Vehículo	Semejanza en X (%)	Semejanza en Y (%)	Semejanza total (%)
carro1	92.388941	91.266215	91.827578
carro2	92.238675	91.174341	91.706508
carro3	91.918328	91.789863	91.854095
General	92.181981	91.410140	91.796060

Cuadro 29: Semejanza en posiciones `X`, `Y` y total - Simulación física con PID.

Vehículo	Media	Desviación	Mínimo	Máximo	CV	Asimetría
carro1	0.065844	0.050885	0.0000	0.1299	0.773	-0.009
carro2	0.056652	0.050664	0.0000	0.1299	0.894	0.282
carro3	0.047516	0.050102	0.0000	0.1299	1.054	0.617
General	0.055001	0.050993	0.0000	0.1299	0.927	0.345

Cuadro 30: Estadísticas descriptivas de la velocidad lineal - Simulación física con PID.

Vehículo	Media	Desviación	Mínimo	Máximo	CV	Asimetría
carro1	-0.002896	0.634118	-5.1261	3.8531	-218.98	-2.916
carro2	0.059399	0.509132	-3.4775	4.2125	8.571	1.107
carro3	0.025005	0.279323	-3.4736	1.3539	11.171	-6.956
General	0.029099	0.465853	-5.1261	4.2125	16.009	-2.032

Cuadro 31: Estadísticas descriptivas de la velocidad angular - Simulación física con PID.

Vehículo	Tiempo Total (s)	% Tiempo Quieto Lineal	% Tiempo Quieto Angular	% Tiempo Quieto Promedio	% Tiempo Vel Máx
carro1	63.40	25.39	25.39	25.39	100.0
carro2	81.80	30.81	30.81	30.81	100.0
carro3	109.90	35.67	35.67	35.67	100.0
General	85.03	30.62	30.62	30.62	100.0

Cuadro 32: Tiempos y porcentajes en simulación física con PID.

ración con el controlador Lyapunov, reflejando una precisión limitada en el cumplimiento de la trayectoria. Sin embargo, el comportamiento fue homogéneo entre los distintos vehículos, lo que indica una estabilidad en el error a lo largo del recorrido, incluso realizando una trayectoria fluida en la superficie de la lona con irregularidades. Igualmente, no ocurrieron colisiones entre vehículos, lo cual demuestra que, incluso con las irregularidades, el sistema de prevención de colisiones mantiene su funcionalidad.

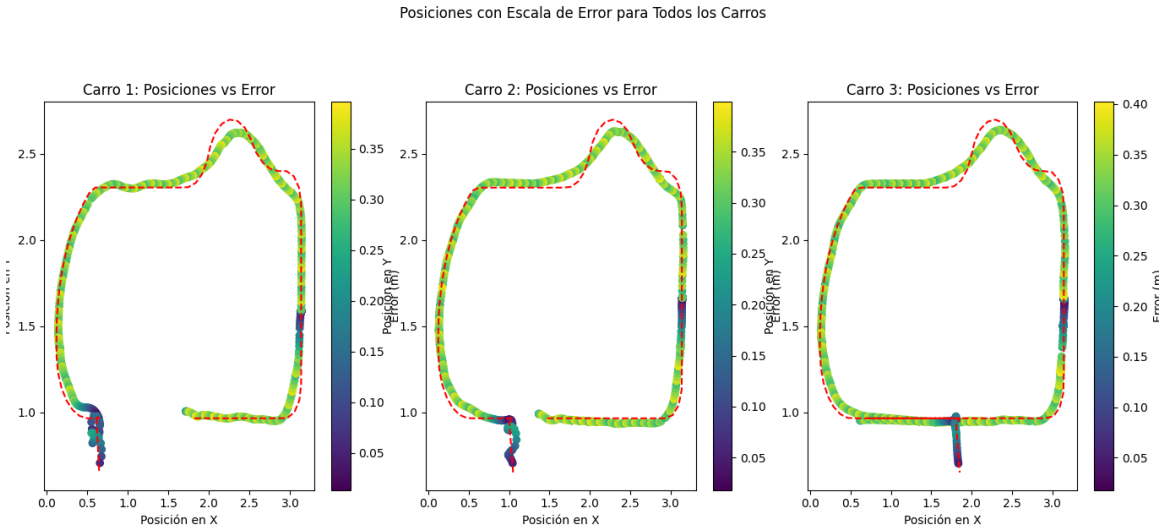


Figura 42: Posiciones de los vehículos con relación al error (m) - Controlador PID en simulación física.

En la Figura 43, se observa que el controlador PID mantiene una velocidad prácticamente constante en la mayoría de los tramos, con solo incrementos insignificantes en puntos con discontinuidades en la trayectoria. Esta constancia en la velocidad contribuye a un movimiento fluido, aunque la precisión en la trayectoria no sea tan alta como con el controlador Lyapunov.

9.10. Limitaciones de la simulación física

Durante las pruebas físicas en el ecosistema Robotat, se identificaron limitaciones en la gestión efectiva del número de vehículos autónomos. Con el controlador basado en Lyapunov, sólo fue posible implementar dos robots. Esto se debe a que este controlador exige un cumplimiento preciso del período de ejecución; cualquier retraso en la comunicación o procesamiento afecta directamente su desempeño. El ecosistema no permite mantener el período ideal, lo que provoca ineficiencias y riesgos de colisión al utilizar más de dos vehículos.

Por otro lado, el controlador PID con acercamiento exponencial, al ser menos dependiente del período de ejecución estricto, permitió la implementación de tres vehículos. Aunque un período de ejecución más corto mejora los resultados, este controlador mostró mayor tolerancia a retrasos en el procesamiento, facilitando la incorporación de vehículos adicionales sin comprometer significativamente el desempeño del sistema.

En síntesis, mientras que la implementación física con un solo vehículo fue estable para ambos controladores, la simulación con múltiples vehículos evidenció las limitaciones del controlador Lyapunov en entornos físicos. El PID con acercamiento exponencial demostró mayor flexibilidad y capacidad de manejo bajo condiciones de mayor carga del sistema.

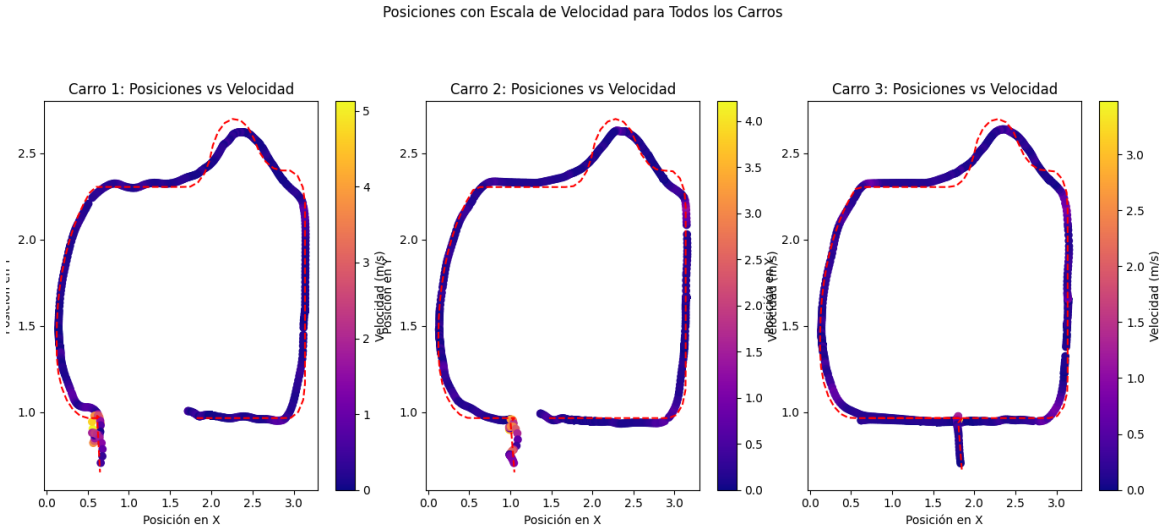


Figura 43: Posiciones de los vehículos con relación a la velocidad (m/s) - Controlador PID en simulación física.

CAPÍTULO 10

Conclusiones

- Se digitalizó un entorno vial a escala que replica características de las carreteras de la Ciudad de Guatemala, utilizando un grafo direccional para evaluar la navegación autónoma en condiciones realistas con simulaciones eficientes y controladas.
- Los algoritmos de búsqueda de rutas optimizaron la navegación en términos de distancia, generando rutas eficientes que respetan las direcciones de las vías, con un tiempo promedio de cálculo de 1.8 milisegundos. Los métodos de `CarroAutonomo`: `senal_alto`, `senal_semaforo`, `senal_bajarVel`, `evitarChoqueCarro` y `checkProxObstaculos` permitieron que el vehículo pudiera interactuar con el entorno para respetar señales de tránsito y evitar colisiones con otros vehículos y obstáculos. Los métodos `calcular_punto_deseado`, `determinar_nuevoPunto`, `controlador` y `enviarVelocidades` garantizaron que el vehículo replicara las trayectorias de forma segura y eficiente.
- Se implementaron y compararon los controladores Lyapunov y PID; ambos respondieron adecuadamente a las señales de tránsito e interacción con otros vehículos. El controlador Lyapunov destacó en precisión, con una semejanza promedio de 99.6 % en simulación virtual y 99.2 % en física, frente al PID con 93.4 % y 91.8 % para las simulaciones de múltiples vehículos. El PID evidenció mayor estabilidad porque mantuvo la velocidad máxima el 100 % del tiempo, mientras que el Lyapunov lo logró un 80.7 % en simulación virtual y 86.1 % en física.
- El sistema de validación con OptiTrack dentro del ecosistema Robotat permitió un monitoreo continuo de los vehículos autónomos retroalimentando la posición de los mismos en tiempo real. Esto favoreció el control porque permitió ajustar los controladores y estrategias de navegación, resultando en un desempeño que permitía a los vehículos autónomos replicar sus trayectoriaso.
- La simulación de tráfico con múltiples vehículos mostró interacciones seguras y realistas. Se identificaron limitaciones en el manejo de varios vehículos simultáneos, puesto que el controlador Lyapunov manejó hasta dos vehículos de forma estable, mientras que el PID gestionó hasta tres vehículos con mayor eficiencia en escenarios concurridos.

CAPÍTULO 11

Recomendaciones

- Optimizar el sistema de evasión de obstáculos en trayectorias curvas. Aunque el sistema actual funciona adecuadamente en tramos rectos, se recomienda mejorar el algoritmo para que sea más eficiente en curvas complejas, con el fin de evitar colisiones en situaciones de tráfico urbano más desafiantes.
- Ampliar el uso del controlador Lyapunov para entornos con múltiples vehículos. Dado que el controlador Lyapunov mostró mayor precisión en simulaciones virtuales, sería recomendable explorar su optimización para gestionar más de dos vehículos simultáneamente en simulaciones físicas, mejorando el desempeño del sistema en escenarios con alta densidad de tráfico.
- Integrar nuevos algoritmos de búsqueda de rutas con aprendizaje automático. Sería útil implementar algoritmos basados en inteligencia artificial o aprendizaje automático para mejorar aún más la capacidad del sistema de adaptarse a diferentes entornos viales y prever situaciones complejas, lo que podría aumentar la eficiencia de la planificación de rutas en tiempo real.
- Mejorar la precisión de la validación física con OptiTrack. La simulación física mostró limitaciones con múltiples vehículos, así que se recomienda optimizar el tiempo de procesamiento y la comunicación del sistema OptiTrack para mejorar el desempeño en simulaciones físicas con varios vehículos, asegurando una mayor precisión en el seguimiento de trayectorias.
- Incorporar sensores adicionales como LIDAR o cámaras estéreo para mejorar significativamente la detección de obstáculos y la precisión en la navegación, especialmente en entornos con condiciones climáticas adversas o infraestructura vial más compleja.
- Extender las capacidades de la interfaz gráfica. La GUI podría expandirse para incluir análisis de tráfico en tiempo real, permitiendo simular la interacción entre vehículos autónomos e integrar la interacción con vehículos no autónomos, peatones o ciclistas en el entorno simulado.

- Explorar nuevos enfoques de control adaptativo para mejorar la seguridad percibida por el pasajero. De esta forma se ajusta de forma dinámica el comportamiento del vehículo con tal que la trayectoria sea más suave y el comportamiento sea mejor.
- Se recomienda aumentar la inteligencia de los robots autónomos, reduciendo la dependencia del sistema computacional central. Esto permitiría una mayor velocidad operativa y un seguimiento de trayectorias más eficiente, al trasladar parte del procesamiento y toma de decisiones al propio vehículo, mejorando la eficiencia del sistema en general. Asimismo, el sistema central no tendría que tomar todas las decisiones de los vehículos y podría ser algo independiente.

CAPÍTULO 12

Bibliografía

[1] M. Daily, S. Medasani, R. Behringer y M. Trivedi, "Self-Driving Cars," *Computer*, vol. 50, n.º 12, págs. 18-23, dic. de 2017, ISSN: 1558-0814. DOI: 10.1109/MC.2017.4451204.

[2] C. P. Montoya, "Robotat: un ecosistema robótico de captura de movimiento y comunicación inalámbrica," 2023, Tesis de licenciatura.

[3] G. Fong, "Implementación de infraestructura a escala para la evaluación de algoritmos por visión de computador para vehículos autónomos," 2023, Tesis de licenciatura.

[4] C. F. A. Valdés, "Pruebas a escala de algoritmos básicos de visión de computadora y control para vehículos autónomos a escala," 2023, Tesis de licenciatura.

[5] *Waymo - Vehículos autónomos - Automóviles que se conducen a sí mismos - Transporte privado a pedido*, es, https://waymo.com/intl/es/, Accessed: 2024-7-7.

[6] X. Bai, Y. Luo, L. Jiang et al., "Bridging the Domain Gap between Synthetic and Real-World Data for Autonomous Driving," *ACM J. Auton. Transport. Syst.*, vol. 1, n.º 2, abr. de 2024. DOI: 10.1145/3633463. dirección: https://doi.org/10.1145/3633463.

[7] Q. Guo, L. Li y X. (Jeff) Ban, "Urban traffic signal control with connected and automated vehicles: A survey," *Transportation Research Part C: Emerging Technologies*, vol. 101, págs. 313-334, 2019, ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2019.01.026. dirección: https://www.sciencedirect.com/science/article/pii/S0968090X18311641.

[8] N. Buckman, A. Hansen, S. Karaman y D. Rus, "Evaluating Autonomous Urban Perception and Planning in a 1/10th Scale MiniCity," *Sensors*, vol. 22, n.º 18, 2022, ISSN: 1424-8220. DOI: 10.3390/s22186793. dirección: https://www.mdpi.com/1424-8220/22/18/6793.

[9] A. Bagherinezhad, M. Alizadeh y M. Parvania, "Rolling Horizon Approach for Real-Time Charging and Routing of Autonomous Electric Vehicles," *IEEE Open Access Journal of Power and Energy*, vol. 11, págs. 94-103, 2024. DOI: 10.1109/OAJPE.2023.3347972.

[10] Y. Shi, C. Shen, H. Wei y K. Zhang, *Advanced model predictive control for autonomous marine vehicles*. Springer, 2023.

[11] L. Carlone, K. Khosoussi, M. Ryll, G. Habibi, V. Tzoumas y R. Talak, *Visual Navigation for Autonomous Vehicles (VNAV)*, MIT OpenCourseWare, Course taught in the Department of Aeronautics and Astronautics, MIT, Fall 2020. Available online: https://ocw.mit.edu/courses/16-485-visual-navigation-for-autonomous-vehicles-vnav-fall-2020/, 2020.

[12] N. Li, "Game-theoretic and set-based methods for safe autonomous vehicles on shared roads," Tesis doct., 2021.

[13] R. Rabin, "Fully Autonomous Vehicles Hit the Streets: Stanford's Robert Rabin on Risks and Evolving Liability Issues," 2023, [En línea]. Disponible en: https://law.stanford.edu/2023/08/31/fully-autonomous-vehicles-hit-the-streets-stanfords-robert-rabin-on-risks-and-evolving-liability-issues/.

[14] M. Martínez y F. Soriguera, "Autonomous vehicles: theoretical and practical challenges," *Transportation research procedia*, vol. 33, págs. 275-282, 2018. DOI: 10.1016/j.trpro.2018.10.103. dirección: http://hdl.handle.net/2117/128503.

[15] P. Udhan, A. Ganeshkar, P. Murugesan, A. R. Permani, S. Sanjeeva y P. Deshpande, *Vehicle Route Planning using Dynamically Weighted Dijkstra's Algorithm with Traffic Prediction*, 2022. arXiv: 2205.15190 [math.OC].

[16] K. Y. Chen, *An Improved A Star Search Algorithm for Road Networks Using New Heuristic Estimation*, 2022. arXiv: 2208.00312 [cs.DS].

[17] A. Derrow-Pinion, J. She, D. Wong et al., "ETA Prediction with Graph Neural Networks in Google Maps," en *Proceedings of the 30th ACM International Conference on Information amp; Knowledge Management*, ép. CIKM 21, ACM, 2021. DOI: 10.1145/3459637.3481916. dirección: http://dx.doi.org/10.1145/3459637.3481916.

[18] E. Gospodinova, "Optimizing the Energy Efficiency of a Lighting Network using Graph Theory," vol. 12, págs. 291-299, 2024, [En línea]. Disponible en: https://www.wseas.com/journals/cr/2024/a565118-007(2024).pdf.

[19] A. Rohman, S. K. Nath y M. Mitra, "Dijkstra's Algorithm on Semigraph," vol. 33, págs. 196-200, 2024, [En línea]. Disponible en: https://doi.org/10.37394/232018.2024.12.19.

[20] M. Karimi y M. Eslamian, "A Resilience-Oriented Graph-Based Method for Restoration of Critical Loads in Distribution Networks Using Microgrids," *Journal of Operation and Automation in Power Engineering*, 2024, ISSN: 2322-4576. DOI: 10.22098/joape.2024.12110.1898. dirección: https://joape.uma.ac.ir/article_2783.html.

[21] P. E. Hart, N. J. Nilsson y B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, n.º 2, págs. 100-107, 1968. DOI: 10.1109/TSSC.1968.300136.

[22] A. Stentz, "Optimal and efficient path planning for partially-known environments," en *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 1994, 3310-3317 vol.4. DOI: 10.1109/ROBOT.1994.351061.

[23] S. M. LaValle, "Rapidly-exploring random trees : a new tool for path planning," *The annual research report*, 1998. dirección: https://api.semanticscholar.org/CorpusID:14744621.

[24] L. Kavraki, P. Svestka, J.-C. Latombe y M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, n.º 4, págs. 566-580, 1996. DOI: 10.1109/70.508439.

[25] E. Sásig, *Robot móvil diferencial: Navegación Autónoma*, https://www.udemy.com/course/robotica-con-python-y-arduino-robot-movil-diferencial/?couponCode=KEEPLEARNING, 2021.

[26] S. K. Malu y J. Majumdar, *Kinematics, localization and control of differential drive mobile robot*, [En línea]. Disponible en: https://globaljournals.org/GJRE_Volume14/1-Kinematics-Localization-and-Control.pdf, 2014.

[27] F. A. Salem a, *Dynamic and kinematic models and control for differential drive mobile robots*, [En línea]. Disponible en: https://inpressco.com/wp-content/uploads/2013/03/Paper6253-2632.pdf, 2013.

[28] D. Koditschek, "Lyapunov analysis of robot motion," en *Tutorial Workshop of 1987 IEEE International Conference on Robotics and Automation*, 1987, 1B-3.

[29] R. Grandia, A. J. Taylor, A. Singletary, M. Hutter y A. D. Ames, "Nonlinear model predictive control of robotic systems with control lyapunov functions," *arXiv preprint arXiv:2006.01229*, 2020.

[30] I. Carlucho, M. D. Paula y G. G. Acosta, "Double Q-PID algorithm for mobile robot control," *Expert Systems with Applications*, vol. 137, págs. 292-307, 2019, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2019.06.066. dirección: https://www.sciencedirect.com/science/article/pii/S0957417419304749.

[31] I. A. Hameed, L. H. Abbud, J. A. Abdulsaheb et al., "A New Nonlinear Dynamic Speed Controller for a Differential Drive Mobile Robot," *Entropy*, vol. 25, n.º 3, 2023, ISSN: 1099-4300. dirección: https://www.mdpi.com/1099-4300/25/3/514.

[32] A. Mallem, S. Nourredine y W. Benaziza, "Mobile robot trajectory tracking using PID fast terminal sliding mode inverse dynamic Control," págs. 1-6, 2016. DOI: 10.1109/CEIT.2016.7929057.

[33] Pololu.com, *3pi+ 32U4 OLED Robot Kit with 30:1 MP Motors (Standard Edition Kit)*, [En línea]. Disponible en: https://www.pololu.com/product/4978/specs.

[34] R. USA, *Pololu 3pi+ 32U4 OLED robot - Assembled Standard Edition (30:1 MP motors)*, [En línea]. Disponible en: https://www.robotshop.com/products/pololu-3pi-32u4-oled-robot-assembled-standard-edition-301-mp-motors.

[35] T. B. Moeslund, A. Hilton y V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Computer Vision and Image Understanding*, vol. 104, n.º 2, págs. 90-126, 2006, Special Issue on Modeling People: Vision-based understanding of a person's shape, appearance, movement and behaviour, ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2006.08.002. dirección: https://www.sciencedirect.com/science/article/pii/S1077314206001263.

[36] B. Siciliano y O. Khatib, *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer, 2016.

[37] *Primex 41*, en. dirección: https://optitrack.com/cameras/primex-41/.

[38] Guatemala, *Ley y Reglamento de Tránsito*, Accedido: 2024-05-12, 2015. dirección: https://transito.gob.gt/wp-content/uploads/2015/06/Ley-y-Reglamento-Transito.pdf.

[39] M. Scuriatti y F. P. Solorzano, *Rural roads – A lifeline for generating employment and fostering competitiveness in Guatemala*, https://blogs.worldbank.org/en/latinamerica/rural-roads-generating-employment-fostering-competitiveness-guatemala, 2024.

[40] *Infrastructure market in Guatemala*, https://www.tradecommissioner.gc.ca/guatemala/market-reports-etudes-de-marches/0006766.aspx?lang=eng, 2022.

[41] United Nations, "Road safety in action: UNDSS Guatemala and Belize organized driving training to elevate road safety standards,"

[42] C. Stachniss, D. Wilbers, N. Chebrolu et al., *Self-driving cars course*, Online course from StachnissLab, University of Bonn, 2021. dirección: http://ipb.uni-bonn.de/sdc-2021/index.html.

[43] M. Pedersen, *DXFtool*, Recuperado 21 julio, 2024, MATLAB Central File Exchange, 2024. dirección: https://www.mathworks.com/matlabcentral/fileexchange/66632-dxftool.

CAPÍTULO 13

Anexos

13.1. Repositorio de GitHub

El código implementado para este proyecto se puede encontrar en el siguiente repositorio de GitHub: `https://github.com/LPGarrido/Tesis_LPGJ.git`.