



UNIVERSIDAD POLITÉCNICA DE VICTORIA

**DISEÑO Y CONSTRUCCIÓN DE AMBIENTES EN
EL SIMULADOR DE ROBOTS GAZEBO PARA
VEHÍCULOS SIN CONDUCTOR**

**T E S I S A
QUE PARA OBTENER EL GRADO DE
INGENIERÍA EN TECNOLOGÍAS DE LA
INFORMACIÓN**

**PRESENTA:
ISAAC YAÑEZ WONG**

**DIRECTOR
DR. SAID POLANCO MARTAGÓN**

**CO-DIRECTOR
DR. MARCO ANTONIO NEGRETE VILLANUEVA
ORGANISMO RECEPTOR
UNIVERSIDAD NACIONAL AUTÓNOMA DE
MÉXICO**

CIUDAD VICTORIA, TAMAULIPAS, ABRIL DE 2021

Ciudad Victoria, Tamaulipas, a 4 de enero de 2021

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
DR. MARCO ANTONIO NEGRETE VILLANUEVA
PROFESOR INVESTIGADOR
COYOACÁN, CIUDAD DE MÉXICO

La Universidad Politécnica de Victoria tiene a bien presentar a sus finas atenciones al alumno **YANEZ WONG ISAAC**, con número de matrícula **1630062** del programa académico **INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN**, a fin de realizar su **ESTADÍA** desarrollando trabajos y actividades directamente relacionadas con el proyecto **"DISEÑO Y CONSTRUCCIÓN DE AMBIENTES PARA VEHÍCULOS SIN CONDUCTOR UTILIZANDO EL SIMULADOR GAZEBO"** para lo que deberá cubrir un total de **600** horas en el periodo comprendido del **4 de enero de 2021 al 16 de abril de 2021**



EN EL TEXTO DE LA CARTA DE ACEPTACIÓN QUE EMITA SU EMPRESA DEBERÁ MENCIONARSE LA MODALIDAD EN LA QUE EL ALUMNO DEBE REALIZAR LAS PRÁCTICAS PROFESIONALES, LAS CUALES PUEDEN SER PRESENCIAL, A DISTANCIA O MIXTA.

Es importante mencionar que el ingreso del alumno(a) a las instalaciones de su empresa, estará sujeto al cumplimiento de los protocolos derivados de la Contingencia COVID 19, al Reglamento Interno de Seguridad e Higiene y a la normatividad que Ustedes determinen.

Hacemos de su conocimiento que el alumno cuenta con la protección del **SEGURO FACULTATIVO IMSS** con número de afiliación **15-13-98-7170-9**.

Se expide este documento con la finalidad de formalizar la presentación y aceptación del alumno para realizar sus prácticas profesionales en la empresa mencionada. **NO TENDRÁ VALIDEZ PARA OTROS FINES.**

ATENTAMENTE


M.A. OTHÓN CANO GARZA
DIRECTOR DE VINCULACIÓN
M.S.I. MARIO HUMBERTO RODRIGUEZ CHÁVEZ
TITULAR DEL PROGRAMA
ACADÉMICO DE INGENIERÍA EN
TECNOLOGÍAS DE LA INFORMACIÓN



Ciudad de México, a 04 de enero de 2021

**FACULTAD DE INGENIERÍA
DIVISIÓN DE INGENIERÍA ELÉCTRICA
DEPARTAMENTO DE PROCESAMIENTO DE SEÑALES**

Asunto: Carta de aceptación

**M.A. OTHÓN CANO GARZA
DIRECTOR DE VINCULACIÓN
UNIVERSIDAD POLITÉCNICA DE VICTORIA
PRESENTE**

Por medio de la presente, le comunico que el alumno **ISAAC YAÑEZ WONG**, inscrito en la carrera de Ingeniería en Tecnologías de la Información, con matrícula **1630062**, en la Universidad Politécnica de Victoria, ha sido **ACEPTADO** para realizar su periodo de **ESTADÍA**, en el Departamento de Procesamiento de Señales, Facultad de Ingeniería, UNAM, bajo la dirección del Dr. Marco Antonio Negrete Villanueva.

El alumno Isaac Yañez Wong desarrollará el proyecto de investigación titulado "Diseño y Construcción de Ambientes para Vehículos sin Conductor utilizando el Simulador Gazebo", durante el periodo del 04 de enero al 16 de abril de 2021.

Se extiende la presente a petición del interesado para los fines y usos que considere pertinente.

Atentamente,

Dr. Marco Antonio Negrete Villanueva
Profesor Asociado de Tiempo Completo
Facultad de Ingeniería, UNAM.
Tel. 56223041
marco.negrete@ingenieria.unam.edu

NO OFICIAL



Ciudad de México, 16 de abril de 2021

**FACULTAD DE INGENIERÍA
DIVISIÓN DE INGENIERÍA ELÉCTRICA
DEPARTAMENTO DE PROCESAMIENTO DE SEÑALES**

Asunto: Carta de liberación

**M.A. OTHÓN CANO GARZA
DIRECTOR DE VINCULACIÓN
UNIVERSIDAD POLITÉCNICA DE VICTORIA
PRESENTE**

Por medio de la presente, hago constar que el alumno **ISAAC YAÑEZ WONG**, inscrito en la carrera de Ingeniería en Tecnologías de la Información, con matrícula **1630062**, en la Universidad Politécnica de Victoria, ha **CONCLUIDO SATISFACTORIAMENTE SU ESTADÍA**, en el Departamento de Procesamiento de Señales, Facultad de Ingeniería, UNAM, bajo mi dirección.

Durante su estadía, el alumno Isaac Yañez Wong desarrollará el proyecto titulado "Diseño y Construcción de Mundos en el Simulador de Robots Gazebo para Vehículos sin Conductor", durante el periodo del 04 de enero al 16 de abril de 2021. Cabe resaltar que los resultados obtenidos por el estudiante serán utilizados en el proyecto "Visión activa y robótica basada en comportamientos en el desarrollo de vehículos autónomos", que se desarrolla en la Facultad de Ingeniería de la UNAM, y en el proyecto "Hacia los autos autónomos en calles de México: manejo a la defensiva en presencia de conducción imprudente, peatones y baches", que se desarrolla en la Universidad Politécnica de Victoria.

Se extiende la presente a petición del interesado para los fines y usos que considere pertinente.

Atentamente.

Dr. Marco Antonio Negrete Villanueva
Profesor Asociado de Tiempo Completo
Facultad de Ingeniería, UNAM.
Tel. 56223041
marco.negrete@ingenieria.unam.edu

Vo. Bo.

CARTA DE ACEPTACIÓN DEL DOCUMENTO PARA SU IMPRESIÓN

Cd. Victoria, Tamaulipas a 8 de Abril de 2021

Isaac Yañez Wong
PRESENTE

Le comunico que el Programa Académico de Ingeniería en Tecnologías de la Información le ha otorgado la autorización para la impresión de su Tesina de Estadía Práctica cuyo título es:

**Diseño y Construcción de Ambientes en el Simulador de Robots
Gazebo para Vehículos sin Conductor**

ATENTAMENTE



Dr. Said Polanco Martagón
ASESOR INSTITUCIONAL

c.c.p Director de programa académico

EVALUACIÓN DE ESTADÍA

Rúbrica para evaluación de la presentación y el reporte de estadía

Nombre del alumno: **ISAAC YAÑEZ WONG**

Calificación final: **94**

Periodo: **ENERO-ABRIL 2021**

Ponderación	Aspecto a Evaluar	Competente 10	Independiente 9	Básico Avanzado 8	No Competente 5
40	Resultados y Actividades	Estrechamente relacionados al perfil de egreso de su programa académico ✓	Parcialmente relacionados al perfil de egreso de su programa académico	Escasamente relacionados al perfil de egreso de su programa académico	Escasamente relacionados al perfil de egreso de su programa académico
30	Exposición de las actividades de la estadía	Detalladas y sustentadas con respecto a los resultados que se obtuvieron	Detalladas y sustentadas parcialmente con respecto a los resultados que se obtuvieron ✓	Detalladas parcialmente con respecto a los resultados que se obtuvieron	Detalladas escasamente con respecto a los resultados que se obtuvieron
10	Material visual Lenguaje verbal	Uso el lenguaje y la terminología apropiadas; El material visual está organizado, adecuado y suficiente ✓	Uso el lenguaje y la terminología apropiadas El material visual está parcialmente organizado y es suficiente	Uso el lenguaje y la terminología apropiadas; El material visual está parcialmente organizado y es suficiente	Uso el lenguaje y terminología es inapropiada; El material visual no está organizado y es insuficiente
10	Exposición en Idioma Inglés	Pronunciation is clear so language is easily understood (2.5) Uses fluent connected speech, occasionally disrupted by search for correct form of expression (2.5) Uses topic related vocabulary without problems (2.5) Responds to questions using varied and descriptive vocabulary and language structures (2.5)	Pronunciation is understandable, but there are slight errors (2.25) Speech is connected but frequently disrupted by search for correct form of expression (2.25) Uses some topic related vocabulary sufficient to communicate ideas (2.25) Responds to questions using simple but accurate vocabulary and language structures (2.25)	Pronunciation is understandable most of the time, marked native accent and many errors (2) Speaks with simple sentences, sometimes not connected, but is understood (2) Uses basic vocabulary to communicate ideas (2) Partly responds to simple questions, with limited vocabulary and language structures (2)	Pronunciation makes language very difficult to understand (1) Uses one-word/two-word utterances (1) Unable to communicate ideas due to lack of vocabulary (1) Uses isolated words or sentence fragments to respond to questions (1)
5	Respuesta a los cuestionamientos de los evaluadores	Clara y satisfactoria	Clara y parcialmente satisfactoria ✓	Clara e insuficiente	Confusa e insuficiente
5	Autorización de tesina en tiempo y forma	Presenta en tiempo y forma	Presenta en tiempo y forma con algunos requisitos solicitados ✓	Presenta en tiempo y con algunas limitantes de los requerimientos solicitados.	Presenta fuera de tiempo y con los mínimos requerimientos solicitados.


Dr. Said Polanco Martagón
ASESOR INSTITUCIONAL


M.C. Jorge Omar Jasso Luna
EVALUADOR


M.C. Jorge Omar Jasso Luna
EVALUADOR INGLÉS

REGISTRO DE EVALUACION DE EXPOSICIÓN DE ESTADÍA

Siendo las 15:00 hrs del día 12 de Abril de 2021, el alumno **Isaac Yañez Wong**, del programa académico **Ingeniería en Tecnologías de la Información**, con matrícula **1630062**, presentó la exposición de la estadía realizada durante el cuatrimestre **enero-abril 2021**, en **Universidad Nacional Autónoma de México**, con el proyecto titulado **Diseño y Construcción de Ambientes en el Simulador de Robots Gazebo para Vehículos sin Conductor**.

Una vez concluido el proceso de evaluación, y con base a la rúbrica establecida para éste propósito, se determina que la calificación de la estadía es aprobatoria.



Dr. Said Polanco Martagón
ASESOR INSTITUCIONAL



M.C. Jorge Omar Jasso Luna
EVALUADOR



M.C. Jorge Omar Jasso Luna
EVALUADOR DE INGLÉS

Agradecimientos

Agradezco a los miembros de mi familia, a mis padres y a mis hermanos, ya que siempre han sido un fuerte apoyo para mí. Agradezco a la Universidad Politécnica de Victoria, y a mis amigos. Y finalmente, quiero agradecer el apoyo recibido por parte de UNAM-DGAPA con el número PAPIIT IA106520, para el desarrollo de este proyecto a lo largo de mi estadía.

Resumen

El acceso al público en general de un vehículo con la capacidad de conducción automática supone un gran avance en la seguridad vial, ya que de esta manera se podrían eliminar los factores de error humano, por eso es sumamente importante que se hagan las pruebas necesarias para comprobar que el sistema para los vehículos autónomos es óptimo.

En el desarrollo de este proyecto titulado Diseño y Construcción de Ambientes en el Simulador de Robots Gazebo para Vehículos sin Conductor, se trabajó con el simulador Gazebo en conjunto con ROS para crear mundos con características específicas para pruebas de vehículos autónomos, de igual manera se mencionan otros simuladores y se explica el porqué se eligió Gazebo. Se habla de la creación de mundos desde cero, al igual que las distintas tareas que se puede hacer dentro del simulador, como lo son las transformaciones de objetos y la población de los mundos con modelos. También se explican los sensores que son utilizados para el vehículo dentro de la simulación.

Las principales características que se consideraron para evaluar el correcto funcionamiento dentro de la simulación fue el comportamiento ante: peatones, señales de tráfico, intersecciones y el seguimiento de carriles.

En este documento se habla a detalle sobre las tareas que fueron hechas para este proyecto durante los meses de Enero y Abril de 2021, para el proceso de estadía de la Universidad Politécnica de Victoria.

Palabras clave: Simulador, Vehículo sin Conductor, Gazebo, Vehículo Autónomo

Summary

Access to the general public of a vehicle with automatic driving capacity represents a great advance in road safety, since in this way the factors of human error could be eliminated, which is why it is extremely important that the necessary tests be carried out to check that the system for autonomous vehicles is optimal.

In the development of this project entitled Design and Construction of Environments in the Gazebo Robot Simulator for Driverless Vehicles, we worked with the Gazebo simulator in conjunction with ROS to create worlds with specific characteristics for tests of autonomous vehicles, in the same way they are mentioned other simulators and why Gazebo was chosen is explained. It talks about creating worlds from scratch, as well as the different tasks that can be done within the simulator, such as object transformations and the population of worlds with models. The sensors that are used for the vehicle within the simulation are also explained.

The main characteristics that were considered to evaluate the correct functioning within the simulation were the behavior before: pedestrians, traffic signals, intersections and lane following.

This document talks in detail about the tasks that were done for this project during the months of January and April 2021, for the internship process of the Polytechnic University of Victoria.

Keywords: Simulator, Driverless Vehicle, Gazebo, Autonomous Vehicle

Índice

Agradecimientos	VII
Resumen	VIII
Summary	IX
Índice	x
1. Introducción	1
1.1. Objetivo General	2
1.2. Objetivo Particular	2
1.3. Alcances y limitaciones del Proyecto	2
1.4. Antecedentes de la institución	3
1.5. Antecedentes teóricos	3
1.6. Organización del Documento de Tesina	4
2. Marco Teórico	5
2.1. Vehículos sin conductor	5
2.2. Modelo cinemático del vehículo	5
2.3. ROS	6
2.4. El vehículo a escala AutoMiny	7
2.5. Open Dynamics Engine	8
2.6. Unreal Engine	8
2.7. Unity	9
2.8. Conclusión	10
3. Desarrollo del Sistema	11
3.1. El simulador Gazebo	11
3.2. Cámara RGB-D	11
3.3. Sensores LIDAR	12
3.3.1. Tipos de sensores LIDAR	12
3.4. Mundos en Gazebo	13
3.4.1. Construcción de modelos	13
3.4.2. Pasos para construir un modelo	13
3.4.3. Creación de mundos	13
3.4.4. Transformaciones	14
3.4.5. Población de modelos	14
3.5. Desarrollo de un mundo para detección y seguimiento de carril	15
3.6. Desarrollo de un mundo para detección de señales de tránsito	17
3.7. Detección de bordes Canny	17
3.8. Transformación de línea Hough	18

4. Pruebas y Resultados	20
4.1. Algoritmos	20
4.1.1. Segmentación de carril	20
4.1.2. Control para seguimiento de carril	22
4.1.3. Detección de semáforos	26
4.2. Detección de peatones	27
4.3. Conclusión	30
5. Discusión	31
5.1. Conclusión	31
5.2. Trabajo a Futuro	31
Índice de figuras	32
Índice de Tablas	33
Índice de algoritmos	34
Referencias	35

1. Introducción

La conducción automática supone un gran avance en cuanto a seguridad vial, la idea de que un auto se conduzca sin necesidad de interacción con una persona hace que, al ser un programa especialmente diseñado para conducir, se eviten los accidentes causados por errores humanos, factores como distracción, cansancio, poca experiencia en situaciones complicadas o falta de reflejos deberían ser suplidas por un autómata.

Hoy en día múltiples marcas de vehículos están incorporando en sus modelos más recientes las primeras funciones de seguridad, modelos como el BMW 530d, Volvo V60, Audi A7 entre otros, esto gracias a un sistema de frenado automático de emergencia. Esto se calcula mediante sensores, los cuales permiten frenar incluso antes de que el conductor tenga los reflejos de hacerlo, evitando así atropellos y reduciendo daños irreparables producidos por choques de alcance a simples rasguños, de igual manera mediante patrones de movimiento pueden manipular la dirección del auto para evitar salirse de la carretera, según un estudio de Euro NCAP se estimó que este sistema de frenado ha reducido en un 38 % de la frecuencia de choques por alcance[1].

Un inconveniente de esta tecnología es el no poder probar funciones avanzadas directamente en la vía pública debido al riesgo que un fallo podría representar a los demás conductores, por lo que es casi un requisito probar estos sistemas en un simulador, de esta forma se puede reducir al máximo la posibilidad de un error en el desempeño, y no implicaría ningún coste en caso que ocurra algo inesperado.

Hoy en día, los avances en la seguridad de los automóviles se enfocan en asistir y mejorar la experiencia de los conductores. Por ejemplo, minimizando al mismo tiempo grandes problemas de diseño que tenían modelos anteriores como el bloqueo de los frenos, visión en el ángulo muerto del piloto, bloqueo de la velocidad en ciertas zonas geográficas, control de velocidad crucero para reducir el cansancio de los pies y aparcamiento automático. Sin embargo, son pocas las que están enfocadas al controlar de forma total el automóvil debido al gran riesgo que representa el realizar una maniobra apresurada por parte de la computadora sobre el auto, ya que dependiendo de la velocidad a la que se vaya el vehículo, el modelo, peso y condiciones del auto, las consecuencias serán diferentes en cada caso, y en ocasiones, desastrosas.

Por otra parte, tratar de probar en el mundo real todas las posibles reacciones que pueden surgir, en un escenario con múltiples variables representadas por vehículos de diferente modelo, resultaría muy costoso y poco práctico. Es por eso que se requiere de un ambiente simulado, que permite reproducir escenarios, con una representación lo más parecida a la realidad.

El uso de simuladores puede arrojar resultados muy significativos en investigación y desarrollo tecnológico en vehículos sin conductor. El simulador Gazebo es el más adecuado para implementar estos ambientes simulados.

Dentro de Gazebo se hace la construcción de modelos, que son descripciones de ambientes

de trabajo para robots, que incluyen elementos simulados como objetos móviles y estáticos. Dichos ambientes serán orientados a evaluar algoritmos de navegación segura y conducción a la defensiva de un coche sin conductor. Por tanto, el ambiente deberá poseer tanto elementos estáticos (calle/carretera, casas, árboles) como objetos dinámicos como personas u otros coches.

1.1. Objetivo General

Diseñar e implementar ambientes apropiados en el simulador Gazebo para evaluar algoritmos de conducción autónoma de un vehículo sin conductor dotado con videocámara RGB y de profundidad y sensor láser, para mejorar la seguridad de la operación de dicho vehículo sin conductor.

1.2. Objetivo Particular

- Realizar un manual de construcción de mundos en Gazebo como referencia rápida para la reconfiguración de nuevos mundos.
- Diseñar y construir los mundos apropiados para evaluar la conducción del vehículo frente a las siguientes situaciones: a) peatón cruzando la calle, b) segundo coche rebasando en el ambiente, c) segundo coche rebasando y cambiando inesperadamente de carril, d) baches en la pista.
- Realizar evaluación inicial de la integración de los sistemas de conducción y los objetos dinámicos en el mundo.

1.3. Alcances y limitaciones del Proyecto

Alcances:

- Se podrá utilizar el simulador para pruebas de desempeño de algoritmos de control y visión computacional.
- Se simulará el ambiente y el hardware de modo que, a través de la plataforma ROS, sea transparente a los algoritmos de visión y control, esto con el objetivo de que sea fácil utilizar otros algoritmos o bien otro simulador.
- Este trabajo se utilizará en el proyecto "Visión Activa y Robótica Basada en Comportamientos en el Desarrollo de Vehículos Autónomos" patrocinado por la UNAM-DGAPA.
- Esta tesina servirá como documentación a otros estudiantes e investigadores que deseen incorporarse al proyecto.

Limitaciones:

- No se modelará la dinámica del vehículo, sólo la parte cinemática.
- El parecido de los sensores simulados con los sensores reales estará limitado por las capacidades del simulador Gazebo.

- No se considera ruido en el sensor RGB-D.
- Se considera que se tiene la capacidad de cómputo necesaria para correr los ambientes simulados.

1.4. Antecedentes de la institución

En el Laboratorio de Biorrobótica, de la Facultad de Ingeniería de la UNAM, se han desarrollado robots móviles autónomos de diversos tipos: robots de servicio doméstico, robots humanoides bípedos, robots para educación, y recientemente, vehículos a escala sin conductor. Entre los trabajos más importantes que se han desarrollado en el laboratorio están [2], [3], [4] y [5]. Actualmente, en el laboratorio se tiene el proyecto "Visión activa y robótica basada en comportamientos en el desarrollo de vehículos autónomos", proyecto patrocinado por la misma Universidad. También se colabora en el proyecto "Hacia los autos autónomos en las calles de México: manejo a la defensiva en presencia de conducción imprudente, peatones y baches" de la Universidad Politécnica de Victoria. Este trabajo será parte de ambos proyectos de investigación.

1.5. Antecedentes teóricos

El comienzo del desarrollo en el ámbito de los sistemas avanzados de control de vehículos surgió en la década de los sesenta, por parte del departamento de desarrollo de investigación de General Motors. Se desarrolló un sistema de control que controlaba el volante, el acelerador y los frenos de sus vehículos; el cual probado en circuitos privados. Sin embargo, desafortunadamente nunca se llegó a manufacturar para el mercado, probablemente porque la tecnología de esa época no logró producir un control lo suficientemente exacto para ser seguro [6].

Otra solución que se fue perfeccionando con el paso del tiempo fueron los vehículos de guiado automático o AGV por sus siglas en inglés, éstos se idearon con la mentalidad de la manipulación de vehículos de carga y de remolque sin necesidad de tener un conductor presente, de esta forma se garantiza la realización de tareas repetitivas con una alta cadencia. Estas tecnologías se implementan hoy en día en grandes almacenes y lugares en donde se transportan gran cantidad de materiales peligrosos [7].

Otro gran avance del control de vehículos modernos es el uso de sistemas electrónicos de dirección asistida, que pretende reemplazar los sistemas hidráulicos. Las simulaciones de estos sistemas presentan estructuras con un doble controlador que permitía controlar sistemas genéricos en conjunto con el motor y la inercia de la dirección [8].

De igual manera, en el reporte "Public perception of self driving cars: The case of Berkeley, California." [9] se hace una observación interesante, tomando en cuenta la vista del ojo público, a pesar de que los vehículos autónomos son un paso adelante para reducir los problemas con los que cuenta el transporte y mejorar la seguridad, muchas personas están reacias a el cambio que presentan estos vehículos, debido a la falta de control, entre otras cosas.

	Enfoque	Área de uso
Desarrollo de un sistema de detección y manipulación de objetos para un robot de servicio	Desarrollo de un sistema	Manipulación de objetos
Automatic car controls for electronic highways	Investigación	Viaje por carretera
Vehículos de Guiado Autónomo (AGV) en Aplicaciones Industriales: Una Revisión	Análisis del estado del arte	Transporte de objetos y materia prima
Robust two degree-of-freedom add-on controller design for automatic steering	Implementación de diseños	Dirección automática del vehículo
Public perception of self driving cars: The case of Berkeley, California.	Investigación	Transporte público y transporte privado

Tabla 1: Tabla comparativa de los antecedentes.

Como se puede observar, el uso de vehículos autónomos tiene un gran margen de áreas por cubrir, que van más allá de solamente el uso de vehículos personales, y que a pesar de las posibles opiniones mixtas que pueda encontrarse por parte del público general, la implementación de estos presenta un gran avance en la calidad de vida y puede reducir el tiempo que toma a algún proceso realizarse.

1.6. Organización del Documento de Tesina

El contenido de este documento es presentado en los siguientes capítulos:

- Capítulo 2, el cual se refiere al marco teórico referente a los conceptos que son precisos tener en cuenta para la correcta comprensión del contenido consiguiente.
- Capítulo 3, consta de las técnicas y herramientas implementadas a lo largo del desarrollo del presente proyecto, detallando los procesos y explicando detenidamente lo utilizado para la conclusión del mismo.
- Capítulo 4, se muestra el resultado obtenido tras el desarrollo realizado, mediante la experimentación se muestran diferentes escenarios y respuestas obtenidas.
- Capítulo 5, se dan las conclusiones consecuentes de la experimentación y testeo de lo desarrollado, además se ofrece trabajo futuro que aporta mejoras al proyecto de primera instancia.

2. Marco Teórico

En este capítulo se presentará una revisión del estado del arte referente a los vehículos autómatas, así como los diferentes sistemas auxiliares para la seguridad vehicular, y los componentes principales más utilizados por los sistemas inteligentes de navegación.

2.1. Vehículos sin conductor

Se define a partir de un vehículo que tenga un sistema de asistencia el cual sea capaz de tomar decisiones de cómo comportarse en el tráfico, respetando las normas de vialidad dentro de las limitaciones en las que fue programado [10].

La implementación de un vehículo inteligente puede realizarse bajo dos diferentes puntos de vista, vehículos con sistemas autónomos que están equipados con todos los componentes e inteligencia necesaria para operar, y vehículos con sistemas cooperativos, en donde toda la información necesaria para la operación se encuentra distribuida en el entorno. Estos dos enfoques suelen combinarse para beneficiarse de información adicional cuando se encuentra información adicional recolectada por los elementos cooperativos mejorando el rendimiento del sistema.

Los vehículos autónomos se enfocan en ciertas actividades que se pueden clasificar como:

- Asistencia pasiva. Esta función alerta al usuario de peligros y le da indicaciones para evitar riesgos al momento de conducir.
- Asistencia activa. Son acciones que incluyen tomar el control de los elementos responsables de la velocidad y la conducción del vehículo.
- Conducción autónoma. Son sistemas con la capacidad de tener el control completo del vehículo y realizar una conducción simulando a la propia de un humano, con la diferencia de buscar disminuir los errores que un humano podría cometer.

2.2. Modelo cinemático del vehículo

Es representado por un ambiente tridimensional generado en computadora el cual simula el recorrido de un vehículo de un punto a otro, esto nos ayuda a poner a prueba las reacciones que tendrá el autómata ante ciertas circunstancias, como planificación de movimientos, interacción ante obstáculos, peatones y otros vehículos. Esto facilita la maduración del algoritmo de planificación y evitar que sucedan errores en un ambiente real que pueda causar daños materiales o poner en riesgo vidas humanas [11].

El estado cinemático del robot se puede describir con un vector de tres variables (x, y, θ) , donde (x, y) corresponde a la posición en un plano y θ corresponde a la orientación del vehículo. Puesto que solo se toma en cuenta la parte cinemática, se consideran como señales de entrada la velocidad lineal v_r (considerando tracción en las llantas traseras) y el ángulo

de las llantas delanteras δ . El modelo en variables de estado que describe el movimiento está dado por:

$$\dot{x}_r = v_r \cos \theta \quad (1)$$

$$\dot{y}_r = v_r \sin \theta \quad (2)$$

$$\dot{\theta} = \frac{v_r}{l} \tan \delta \quad (3)$$

donde (x_r, y_r) corresponde a la posición del centro del eje trasero, v_r es la velocidad lineal de las llantas traseras y l es la distancia entre ejes. En el caso de un vehículo de tracción delantera, el modelo corresponde a:

$$\dot{x}_f = v_f \cos(\theta + \delta) \quad (4)$$

$$\dot{y}_f = v_f \sin(\theta + \delta) \quad (5)$$

$$\dot{\theta} = \frac{v_f}{l} \sin \delta \quad (6)$$

donde (x_f, y_f) es la posición del centro del eje delantero y v_r es la velocidad lineal de las llantas delanteras.

El simulador a desarrollar debe ser capaz de modelar el movimiento del vehículo con base en este modelo.

2.3. ROS

ROS (Robot Operating System, por sus siglas en inglés), se trata de un framework utilizado para la escritura de software para robots. Se compone de una colección de herramientas, librerías y convenciones que buscan simplificar la tarea de crear comportamientos de robot complejos y robustos a lo largo de distintas plataformas[12].

Debido a que crear software robusto para un robot de propósito general es difícil, ROS fue construido desde sus bases para alentar el desarrollo de software de robótica colaborativo.

ROS cuenta con los siguientes componentes base, una infraestructura de comunicación, ofrece una interfaz de envío de mensajes que provee comunicación entre procesos. De igual manera se cuenta con un middleware que sirve para dar una manera a las tareas de compartir información de la configuración a través de un almacenamiento global de valores clave.

En adición de los componentes middleware, ROS ofrece librerías específicas de robots, siendo algunas de ellas las siguientes[13]:

- Definiciones de mensajes estandarizados para robots.
- Librería de geometría robótica.
- Descripción de lenguaje robótico.
- Diagnósticos.

- Localización.
- Navegación.

ROS es una plataforma ampliamente utilizada en el desarrollo de robots móviles autónomos. En el Laboratorio de Biorrobótica de la UNAM se han desarrollado diversos robots bajo esta plataforma.

La UNAM ha trabajado con ROS para el desarrollo de robots de servicio, como el que se presenta en Semantic Reasoning in Service Robots Using Expert Systems[14]. Las personas toman como preferencia comunicarse haciendo uso del idioma hablado, por lo que es natural que al dirigirse a un asistente se haga de esta manera, ya sea que se dirijan a una persona o a un robot. Y debido a las mismas implicaciones que comprende un idioma hablado (flexibilidad, ambigüedades, excepciones y basados principalmente en las intenciones del hablador), una solución de comprensión de lenguaje natural no puede basarse solamente en el léxico y sintaxis. Este artículo presenta el módulo semántico en una arquitectura propuesta para robots de servicio, denominada VIRBOT. En el caso de la comprensión del lenguaje natural, mostramos que al combinar la IA simbólica con técnicas de procesamiento de señales digitales, se obtiene un desempeño de vanguardia.

Otro ejemplo es el artículo A Motion-Planning System for a Domestic Service Robot[15]. Los robots de servicio están diseñados para ayudar a los humanos en ambientes no industriales, como pueden ser los hogares u oficinas. Para poder lograr esta tarea, los robots de servicio deben de contar con habilidades como reconocimiento de objetos, manipulación de objetos, detección y reconocimiento de rostros, reconocimiento y síntesis de lenguaje y todavía más importante, la navegación en entornos dinámicos, entre otros. Este artículo describe un sistema de planificación de movimiento completamente implementado que comprende desde algoritmos de planificación de movimiento y trayectoria hasta representación espacial y navegación activa basada en el comportamiento. El sistema propuesto se implementa en Justina, un robot de servicio doméstico cuyo diseño se basa en el ViRBot, una arquitectura para operar robots virtuales y reales que abarca varias capas de abstracción, desde el control de bajo nivel hasta la planificación simbólica.

2.4. El vehículo a escala AutoMiny

Se trata de un vehículo autónomo desarrollado por la universidad Freie Universität Berlin para motivos educacionales y de investigación[16]. Es un vehículo con un sistema completo con sensores como una cámara estéreo infrarroja y LIDAR, sensores de retroalimentación, para el ángulo de posición del volante y un codificador de motor, un CPU y GPU poderosos al igual que LEDs para simular las luces de un vehículo. Autominy está diseñado para no requerir sensores o computo externos, pero puede ser controlado de manera remota o programado para funcionar de manera autónoma. Este vehículo es ejecutado con ROS melodic.

2.5. Open Dynamics Engine

Un motor de simulación que se utiliza es Open Dynamics Engine, se trata de una librería de código abierto usada para la simulación de dinámicas de cuerpos rígidos. Es sencillo de usar e independiente de alguna plataforma, con API de C/C++. Cuenta con articulaciones avanzadas e integración de detección de colisiones con fricción. Es sumamente útil para simular vehículos, objetos en ambientes de realidad virtual y criaturas virtuales[17].

2.6. Unreal Engine

Unreal Engine es un conjunto de herramientas de desarrollo para el trabajo con tecnología en tiempo real. Desde visualizaciones de diseño y experiencias cinematográficas hasta juegos de alta calidad en PC, consolas, dispositivos móviles, VR y AR[18].

Algunos de los elementos de simulación con los que cuenta son:

- Partículas Niágara y efectos visuales.
- Herramientas de vestimenta.
- Físicas de caos y sistemas de destrucción.
- Cabello y piel a base de hebras.

Debido a la gran versatilidad que se puede obtener con este motor, ha sido implementado en otras áreas fuera de los juegos por computadora, como por ejemplo en el documento de Qiu y Yuille, comparten que:

“Los gráficos por computadora no solo pueden generar imágenes sintéticas y verdad fundamental, sino que también ofrecen la posibilidad de construir mundos virtuales en los que: (i) un agente puede percibir, navegar y tomar acciones guiado por algoritmos de IA, (ii) las propiedades de los mundos pueden modificarse (por ejemplo, material y reflectancia), (iii) se pueden realizar simulaciones físicas y (iv) se pueden aprender y evaluar algoritmos.”[19]

Un ejemplo del uso práctico de Unreal Engine en la robótica es el artículo USARSim: a robot simulator for research and education[20], donde se muestra un simulador de robot de alta fidelidad de código abierto que puede ser usado tanto para educación como para investigación. La diferencia principal con otros simuladores es que constituye un motor de simulación usado principalmente para competencias de robots virtuales dentro de la iniciativa RoboCup. Debido al alarmante decremento en el número de estudiantes en programas de ingeniería y ciencia en Estados Unidos, han surgido una gran cantidad de iniciativas para revertir esta tendencia. Y el éxito de iniciativas como RoboCup, una competencia enfocada a estudiantes que se realiza cada año, sirve como punto de entrada a entusiastas alrededor del mundo, demostrando que la robótica tiene un gran papel en involucrar a los estudiantes. USARSim es construido sobre el motor Unreal Engine 2.0.

También se encuentra el artículo A game engine based simulation of the NIST urban search and rescue arenas[21], donde se explica el desarrollo de simulaciones interactivas por parte del Instituto Nacional de Estándares y Tecnología(NIST, por sus siglas en inglés). Se tiene un ambiente estandarizado de desastres que consisten en tres escenarios de dificultad progresiva, arenas amarillas, naranjas y rojas. Las tareas se enfocan en la conducta de robots, y la interacción física con ambientes llenos de escombros estandarizados pero desordenados, donde la simulación va a ser usada para probar y evaluar el diseño de interfaces de teleoperación. Para el desarrollo de este artículo se usaron varios motores gráficos para el desarrollo de videojuegos, entre ellos Unreal Engine.

2.7. Unity

Es una plataforma de desarrollo 3D en tiempo real que permite a artistas crear experiencias interactivas, se encuentra disponible en múltiples sistemas operativos[22]. Cuenta con un componente de sistema de partículas, que simula entidades de fluido, nubes y llamas al generar y animar una gran cantidad de imágenes 2D en la escena. También se puede trabajar con el grafo de efectos visuales que permite controlar los efectos por medio de una lógica visual basada en nodos. Se puede usar para efectos simples como simulaciones complejas, por ejemplo, crear uno o varios sistemas de partículas.

Unity también ha sido usado en una gran cantidad de trabajos académicos, uno de ellos siendo “Robot design using Unity for computer games and robotic simulations” [23], donde se presenta un enfoque para diseñar un robot, mientras se implementa el entorno de desarrollo Unity como la herramienta principal para dicho robot virtual. También se hace énfasis en las colecciones de recursos que se encuentran como paquetes de Unity, los cuales al ser usados mejoran considerablemente el flujo de trabajo.

A continuación se muestran un par de ejemplos de la implementación de Unity para investigaciones relacionadas con la simulación por computadora:

La interacción robot-humano es un campo interdisciplinario que integra un gran rango de conocimientos en un solo proyecto, lo que puede resultar muy complicado. Para el artículo The Robot Engine – Making The Unity 3D Game Engine Work For HRI[24] se construyo “The Robot Engine” haciendo uso del motor Unity. Una ventaja de trabajar con Unity es que ofrece a las personas sin conocimiento de programación el uso de un conjunto de herramientas para animación y la interacción con herramientas de diseño para programar de manera visual y animar robots. Para este artículo se analizaron múltiples técnicas de animación que son comunes en los videojuegos de computadora que fueran capaces de hacer los movimientos de robot más naturales y convincentes.

También se tiene el artículo Navigation Simulation of a Mecanum Wheel Mobile Robot Based on an Improved A* Algorithm in Unity3D[25], donde se explica que la simulación por computadora es una manera efectiva para la investigación de algoritmos para navegación robótica. En este artículo se propone el uso de Unity para la implementación de simulación en tiempo real, simulación tridimensional, algoritmos de simulación de navegación visual, entre

otros. Trabajando con estas herramientas se puede crear de manera rápida un prototipo de un robot virtual, que además puede ser importado su modelo 3D, junto con sus articulaciones virtuales, sensores y su sistema de navegación. De esta manera, los resultados que se muestran en el artículo muestran que el desarrollo en Unity es factible, eficiente y flexible.

2.8. Conclusión

El desarrollo de vehículos autónomos es un problema abierto y la falta de hardware se puede solucionar con simuladores, Gazebo es una opción debido a las múltiples características que ofrece, como por ejemplo la posibilidad de generar información para los sensores, con la opción de generar ruido, utilizar cámaras 2D o 3D, la implementación de sensores Kinect, sensores de contacto, el acceso a una variedad de motores de física como DART[26], Simbody[27] y la libertad de desarrollar plugins para robots, sensores o el control ambiental, y que se puede usar para probar algoritmos de segmentación de carril, evasión y detección de señales de tránsito. Se escogió el simulador Gazebo para llevar a cabo este proyecto, Gazebo cuenta con un buen motor de física al igual que los motores Unity y Unreal, pero con la diferencia que Gazebo cuenta con un amplio trabajo de la comunidad de desarrolladores para integrar el simulador con la plataforma ROS; Unity y Unreal al no contar con esto presenta dificultades para el desarrollo del proyecto.

3. Desarrollo del Sistema

3.1. El simulador Gazebo

Gazebo es un simulador 3D multi robot, nos permite simular con gran precisión y eficiencia una gran variedad de robots, objetos y sensores complejos brindando una retroalimentación realista de choques entre objetos y respuesta de sensores [28].

Sus características principales son:

- Es un software libre lo cual permite ser adaptado a las necesidades de cada usuario.
- Su capacidad de simulación sumamente realista nos permitirá una representación muy cercana a la realidad de físicas en cuerpos sólidos, simular el comportamiento de robots y sus interacciones con otros objetos o viceversa.
- Creación de escenarios de simulación personalizados, manipulando las características de los entornos, y las interacciones que cada superficie puede llegar a tener con un contacto directo, obstáculos, gravedad, desplazamiento, y las características de cada robot las cuales pueden modificarse de forma independiente.
- La facilidad de incorporar sensores como láser, medidores de luz, cámaras, sensores de contacto, seguidores de patrones, todos simulados de forma precisa y con la facilidad de ser agregados en forma de plugins.

3.2. Cámara RGB-D

Sistema de visión artificial de bajo costo que permite la detección de obstáculos interpuestos en una trayectoria permitiendo su evasión generando un modelo 3d de la escena con una representación de puntos [29].

Este dispositivo se compone de dos cámaras y un láser infrarrojo:

- Cámara RGB: Es una cámara convencional que captura información en color del entorno que observa.
- Cámara Infrarroja: La cámara infrarroja recoge el patrón y por hardware se calcula la profundidad de cada punto. Para ello, la cámara tiene un patrón de puntos memorizado para una profundidad conocida. Al poner objetos delante, el patrón aparece “desplazado”, con este desplazamiento se calcula la profundidad.
- Láser infrarrojo: Dispositivo que proyecta un patrón de puntos sobre el escenario que tiene enfrente, los rayos chocan contra los objetos del entorno y vuelven a la cámara creando un patrón dependiendo del momento en que regresan a la cámara generando una imagen con los patrones de profundidad.

3.3. Sensores LIDAR

Es un sensor que mediante la emisión de haces de rayos infrarrojos y una lente receptora, recibe el reflejo de los rayos emitidos cuando impactan con los objetos que se encuentran alrededor y estas a su vez son captadas por el lente, esto genera una nube de puntos del entorno, esto genera una imagen tridimensional que ubica que se está actualizando constantemente, esto permite saber cómo se desplazan dichos objetos [30].

3.3.1. Tipos de sensores LIDAR

Por tipo de láser:

- LIDAR de pulsos. El proceso para la medición de la distancia entre el sensor y el terreno se lleva cabo mediante la medición del tiempo que tarda un pulso desde que es emitido hasta que es recibido. El emisor funciona emitiendo pulsos de luz.
- LIDAR de medición de fase. En este caso el emisor emite un haz láser continuo. Cuando recibe la señal reflejada mide la diferencia de fase entre la emitida y la reflejada. Conocida ésta solo hay que resolver el número de longitud de ondas enteras que ha recorrido (ambigüedades).

Por tipo de escaneado:

- Líneas. Dispone de un espejo rotatorio que va desviando el haz láser. Produce líneas paralelas en el terreno como patrón de escaneado. El inconveniente principal de este sistema es que al girar el espejo en una sola dirección no siempre tenemos mediciones.
- Zigzag. En este caso el espejo es rotatorio en dos sentidos (ida y vuelta). Produce líneas en zigzag como patrón de escaneado. Tiene la ventaja de que siempre está midiendo pero al tener que cambiar de sentido de giro la aceleración del espejo varía según su posición. Esto hace que en las zonas cercanas al límite de escaneado lateral (donde varía el sentido de rotación del espejo), la densidad de puntos escaneados sea mayor que en el nadir.
- De fibra óptica. Desde la fibra central de un cable de fibra óptica y con la ayuda de unos pequeños espejos, el haz láser es desviado a las fibras laterales montadas alrededor del eje. Este sistema produce una huella en forma de una especie de circunferencias solapadas. Al ser los espejos pequeños, la velocidad de toma de datos aumenta respecto a los otros sistemas pero el ángulo de escaneado (FOV) es menor.
- Elíptico (Palmer). En este caso el haz láser es desviado por dos espejos que producen un patrón de escaneado elíptico. Como ventajas del método podemos comentar que el terreno es a veces escaneado desde diferentes perspectivas aunque el tener dos espejos incrementa la dificultad al tener dos medidores angulares.

3.4. Mundos en Gazebo

3.4.1. Construcción de modelos

Un modelo va desde alguna figura simple hasta un robot de más complejidad. Se refiere a la etiqueta SDF “model”, que es básicamente un conjunto de enlaces, uniones, objetos de colisión, elementos visuales y complementos. La complejidad del archivo del modelo depende del modelo que se desee construir.

Componentes de los modelos SDF:

- Enlaces: un enlace contiene las propiedades físicas de un cuerpo del modelo.
- Colisión: un elemento de colisión encapsula una geometría que se utiliza para la comprobación de colisiones.
- Visual: un elemento visual se utiliza para visualizar partes de un enlace.
- Inercial: describe las propiedades dinámicas del enlace, como la masa y la matriz de inercia rotacional.
- Sensor: un sensor recopila datos del mundo para usarlos en complementos.
- Luz: un elemento de luz describe una fuente de luz adjunta a un enlace.
- Articulaciones: una articulación conecta dos enlaces.
- Complementos: es una biblioteca compartida creada por un tercero para controlar un modelo.

3.4.2. Pasos para construir un modelo

Recolectar las estructuras Este paso implica recopilar todos los archivos de estructura 3D necesarios para construir el modelo. Gazebo requiere que los archivos de estructura estén formateados como STL, Collada u OBJ, siendo Collada y OBJ los formatos preferidos.

Archivo de modelo SDF Se comienza por crear un archivo de modelo extremadamente simple o copiando de un archivo de modelo existente.

Agregar al archivo del modelo SDF Con un archivo .sdf funcional, se comienza a agregar más complejidad. Con cada nueva incorporación, se carga el modelo utilizando el cliente gráfico para asegurar que su modelo sea correcto.

3.4.3. Creación de mundos

Para comprender mejor el uso y la creación de mundos en Gazebo, se deben tener en cuenta los siguientes tres términos^[31]:

- Mundo: término utilizado para describir una colección de robots, objetos y parámetros globales.

- Estático: son objetos que solo tienen geometría de colisión.
- Dinámico: son objetos que tienen tanto inercia como geometría de colisión.

Existen dos maneras de agregar objetos, una es haciendo uso de figuras sencillas que el mismo simulador brinda (como se puede observar en la figura 1), o agregarlas desde la base de datos de modelos.

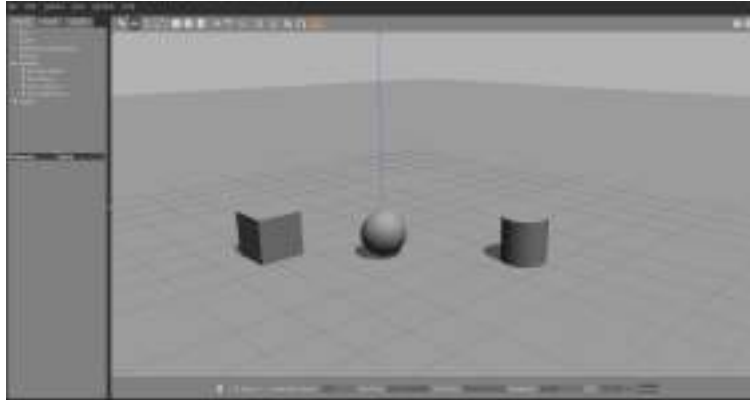


Figura 1: Figuras que se encuentran en el simulador Gazebo[31].

3.4.4. Transformaciones

El simulador Gazebo también permite realizar las siguientes transformaciones, traslación, rotación, escala[31].

- *Traslación* La herramienta de traslación permite mover los objetos a lo largo de los ejes x, y y z. Aunque también se puede lograr esto al arrastrar los objetos haciendo click en ellos.
- *Rotación* La herramienta de rotación permite orientar un modelo alrededor de los ejes x, y y z. Se muestran tres marcadores visuales en forma de anillo sobre el objeto, que permiten rotar el objeto alrededor de los ejes x, y, z.
- *Escala* La herramienta de escala permite cambiar el tamaño de un modelo en las direcciones x, y y z. Se muestra un marcador visual de tres ejes sobre el objeto, que permite escalar las dimensiones x, y, z del objeto.

3.4.5. Población de modelos

Una población consiste en una colección de modelos idénticos, y esto se puede lograr haciendo uso de la etiqueta “población”. Cada población debe tener un nombre único, y esto se especifica mediante el atributo de nombre. Dentro de la etiqueta de población, puede ver cómo seleccionar un modelo usando la etiqueta ”model”. El tipo de población más común consiste en objetos inanimados como árboles, rocas y edificios[32]. En la figura 2 se pueden

ver los tipos de distribución.

Tipos de distribución:

- Random: Modelos colocados al azar.
- Uniforme: Modelos colocados en un patrón de cuadrícula pseudo-2D.
- Cuadrícula: Modelos colocados uniformemente en un patrón de cuadrícula 2D.
- Linear-x: Modelos colocados uniformemente en una fila a lo largo del eje x global.
- Linear-y: Modelos colocados uniformemente en una fila a lo largo del eje y global.
- Linear-z: Modelos colocados uniformemente en una fila a lo largo del eje z global.



Figura 2: Tipos de distribución[32]

3.5. Desarrollo de un mundo para detección y seguimiento de carril

El seguimiento de carril es importante porque es una de las tareas más básicas, en la sección de resultados este es uno de los algoritmos que se va a probar, y para poder desarrollar algoritmos de detección y seguimiento de carril es necesario tener un mundo adecuado para estas pruebas, por lo que se desarrolló un mundo que únicamente consiste en un circuito cerrado. Para esto se cuenta con modelos simples.

Para construir este modelo se está haciendo uso de dos modelos, uno de una carretera recta, y un modelo de una carretera curva. En las imágenes 3 y 4 se pueden ver las texturas utilizadas para estos modelos.



Figura 3: Textura del modelo de la carretera recta.



Figura 4: Textura del modelo de la carretera curva.

Se hace uso de técnicas que fueron descritas previamente para la construcción del mundo, pero para esto simplemente se está poniendo varias copias del modelo en diferentes posiciones y rotaciones.

3.6. Desarrollo de un mundo para detección de señales de tránsito

Es importante tener un mundo para detecciones de señales de tránsito porque es una tarea fundamental en el desarrollo de vehículos autónomos por lo que se requiere de una simulación para poder probar estos algoritmos.

Para construir este mundo se usaron los siguientes modelos:

- Carretera recta.
- Carretera curva.
- Paso de cebra.
- Semáforo.
- Signo de alto.

Finalmente, para la creación del mundo se utilizaron las técnicas que fueron previamente descritas para la población de mundos, donde se colocaron copias de los modelos mencionados con diferentes rotaciones y traslaciones.

3.7. Detección de bordes Canny

Se trata de un algoritmo de múltiples pasos popular desarrollado por John F. Canny, para la detección de bordes[33].

Reducción de ruido Dado que la detección de bordes es susceptible al ruido en la imagen, el primer paso es eliminar el ruido en la imagen con un filtro gaussiano de 5x5.

Encontrar el gradiente de intensidad de la imagen Posteriormente, la imagen una vez suavizada, es filtrada con un kernel de Sobel para las direcciones horizontal y vertical, con esto se obtiene la primera derivada en horizontal (G_x) y vertical (G_y). Se puede encontrar el gradiente del borde, el cual siempre es perpendicular a los bordes, y la dirección de cada píxel de la siguiente manera[33]:

$$Edge_Gradiente(G) = \sqrt{G_x^2 + G_y^2} \quad (7)$$

$$Angle(\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \quad (8)$$

Supresión no máxima Obteniendo la magnitud y dirección del degradado, se hace un escaneo de la imagen para remover los píxeles no deseados que pueden no ser parte del borde. Para lograr esto, en cada píxel, el píxel se comprueba si es un máximo local a su alrededor en la dirección del gradiente.

Umbral de histéresis En este paso se escoge qué bordes son realmente bordes y cuáles no lo son. Para esto se necesitan dos valores del umbral, minVal y maxVal. Cualquier borde que

tenga un valor mayor a maxVal se trata de un borde, y el que tenga un valor menor a minVal es descartado. Los que se encuentran entre estos dos valores son evaluados de acuerdo a su relación, si están conectados con un píxel que se trate de un borde, entonces se considera también como parte del mismo, de lo contrario es descartado.

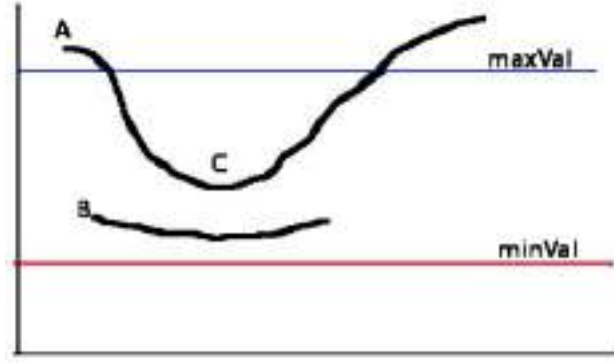


Figura 5: Umbral de histéresis[33].

3.8. Transformación de línea Hough

Se trata de una transformación que se utiliza para detectar líneas rectas. Para que pueda ser utilizada se tiene que detectar primero las líneas de una imagen[34]. Una línea en una imagen puede ser expresada con dos variables, por ejemplo, en un sistema de coordenadas polares pueden ser (r, θ) .

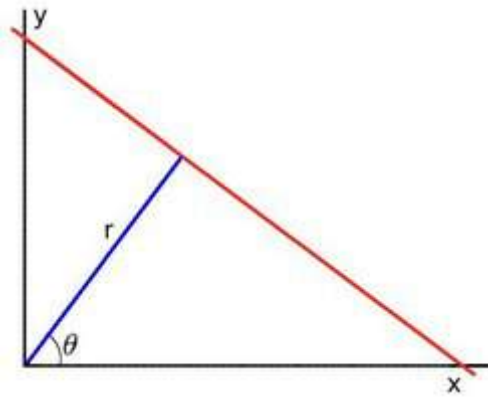


Figura 6: Sistema de coordenadas polares[34].

Para la transformación de Hough se puede escribir la siguiente ecuación:

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right) \quad (9)$$

Ordenando los términos $r = x\cos\theta + y\sin\theta$

Para cada punto (x_0, y_0) , se puede definir que la familia de líneas que pasa por un punto como:

$$r_\theta = x_0.\cos\theta + y_0.\sin\theta \quad (10)$$

Lo que significa que para cada par (r_θ, θ) representa cada línea que pasa por (x_0, y_0)

4. Pruebas y Resultados

4.1. Algoritmos

4.1.1. Segmentación de carril

El único propósito del detector de carril es detectar los carriles de una carretera a partir de la imagen original de la cámara de un coche, haciendo uso de librerías de OpenCV para lograr esto.

En la figura 7 se puede observar en la imagen original (a) que las líneas son de color blanco, y es con ese atributo que se pueden distinguir y segmentar los carriles, ya que es la parte que cuenta con la mayor cantidad de píxeles de ese color. En la imagen binaria (b) se muestra en color blanco todo lo que en la imagen original es de color blanco, y en color negro todo lo que no es blanco en la imagen original. Continuando con la segmentación, el siguiente paso es hacer un filtrado de píxeles color blanco que tengan una densidad mínima, en la imagen filtrada (c) se puede observar el resultado de esto. Y finalmente para poder identificar los carriles por medio de las líneas, el primer paso es obtener los contornos de la imagen filtrada, se requiere hacer uso de la imagen filtrada y los umbrales correspondientes para obtener la imagen de contornos (d).



(a) Imagen original



(b) Imagen binaria



(c) Imagen filtrada



(d) Imagen de contornos

Figura 7: Imágenes procesadas

Lo que interesa es obtener la geometría de la figura 8, segmentando la carretera, para de esta manera aprovechar la geometría y reducir el campo de visión para obtener solamente la imagen donde aparecen los carriles.

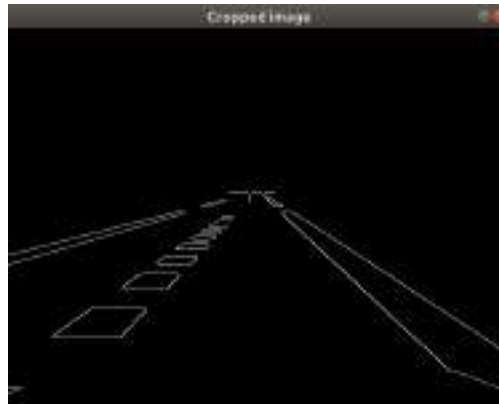


Figura 8: Imagen de los contornos de la sección de interés.

En el algoritmo 1 se muestran los pasos para obtener la imagen de la figura 8.

Algoritmo 1 Algoritmo para obtener los contornos de la imagen.

Entrada: Imagen de contornos.

- 1: altura = Se obtiene la altura de la imagen de entrada.
 - 2: ancho = Se obtiene el ancho de la imagen de entrada.
 - 3: mask = Se crea un vector de ceros con la misma dimensión de la imagen de entrada.
 - 4: triangle = Se crea un vector con los valores de altura.
 - 5: Se rellena un polígono con los valores (mask, triangle).
 - 6: Se obtiene la suma de los valores a partir de la imagen de entrada y mask.
-

La imagen obtenida es la necesaria para calcular y dibujar las líneas de los carriles sobre la imagen original. Lo primero que se tiene que hacer es calcular las líneas. Lo cual se logra por medio de una transformada de Hough.

Con el procedimiento anterior, los resultados son los que se pueden observar en la figura 9.



Figura 9: Segmentación de los carriles de la carretera.

4.1.2. Control para seguimiento de carril

El seguidor de carril se trata de un programa que hace uso de visión computacional para mover un vehículo a través de la segmentación de carriles de una carretera. Se usan los carriles previamente segmentados para que el vehículo pueda avanzar sin salir de su carril.

Dentro de la simulación se pretende que el vehículo sea capaz de seguir una línea recta sin salirse, pero un detalle importante es que al vehículo se le ha indicado que tenga una tendencia a salir del camino. Esta simulación se lleva a cabo dentro del simulador Gazebo. Para esto se requiere dos paquetes de Python:

- CvBridge: Convierte los mensajes de imágenes de ROS a imágenes de OpenCV.
- Cv2: Librería diseñada para resolver problemas de visión computacional.

A continuación se presenta el algoritmo para obtener la imagen proporcionada por el vehículo:

Algoritmo 2 Algoritmo para obtener la imagen del vehículo.

Entrada: Imagen proporcionada por el vehículo.

- 1: bridge = Se convierte la imagen obtenida en ROS a una imagen de OpenCV.
 - 2: La imagen se convierte a escala BGR (blue-red-green).
 - 3: Se muestra la imagen original.
 - 4: Cada frame es mostrado por aproximadamente 33 mili segundos.
-

En la figura 10 se muestra el resultado obtenido.



Figura 10: Imagen original de la cámara del vehículo.

Una vez que se obtiene la imagen de la figura 4, se puede segmentar las líneas de los carriles, basándose en que estos son los únicos elementos de color blanco.

En el siguiente algoritmo se muestra la segmentación de líneas:

Algoritmo 3 Algoritmo para la segmentación de líneas.

Entrada: Imagen proporcionada por el vehículo.

- 1: Se declara el valor del umbral inferior, gris claro.
 - 2: Se declara el valor del umbral superior, blanco.
 - 3: Se toma la imagen de entrada y se convierte a blanco y negro.
 - 4: Se realiza una transformación morfológica.
 - 5: Se crea un filtro para eliminar el ruido de la imagen.
-

En la figura 11 se muestra el resultado de la segmentación.



Figura 11: Imagen binaria. En color blanco se muestra la segmentación del color de interés.

En la figura 12 se muestra que con la perspectiva de las cámaras, las líneas de los carriles forman un triángulo.



Figura 12: Forma de triángulo de los carriles vistos desde la perspectiva de la cámara.

Para la detección del carril se hace uso del algoritmo 1. Una vez que se tiene la imagen en la que las líneas de los carriles se aproximan a ser solamente líneas rectas. Para lograr esto, se usa la transformada de línea de Hough estándar y probabilística. En el algoritmo 4 se muestra el proceso.

Algoritmo 4 Algoritmo para la transformada de Hough.

Entrada: Imagen recortada.

- 1: Se crea la transformación de Hough con valores ya definidos.
 - 2: Se aplica la transformación a la imagen de entrada.
-

Una vez que se obtiene este punto, se dibujan en una imagen para ver la ubicación, se explica en el algoritmo 6.

Algoritmo 5 Algoritmo para dibujar las líneas de los carriles.

Entrada: Imagen con filtro, líneas promedio.

- 1: Se representan los 3 canales RGB de la imagen.
 - 2: Se define el color azul.
 - 3: Se define el grosor de la línea.
 - 4: Se define el color de los puntos.
 - 5: Se define el tamaño de los puntos.
 - 6: Se crea la imagen donde se muestran las líneas y los puntos, combinando los datos de entrada.
 - 7: Se muestra la imagen.
-

En la figura 13 se muestra la imagen obtenida del algoritmo 5.

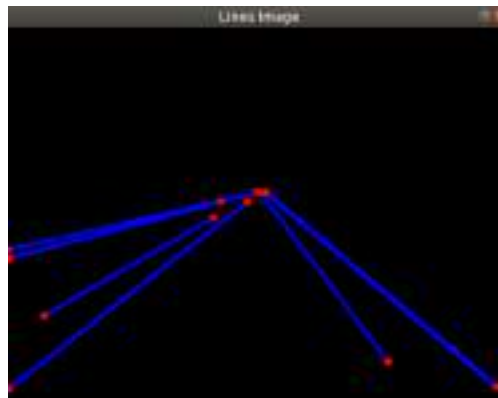


Figura 13: Representación de las líneas de los carriles por medio de líneas rectas.

Hasta este punto se tiene todo lo relacionado con la visión computacional, el siguiente paso es el movimiento del vehículo. El control que se va a utilizar corrige la dirección del vehículo, al comparar la inclinación de las líneas azules con la dirección del vehículo. Las coordenadas de los puntos de las líneas azules son el recurso más importante, ya que con esto se calcula el ángulo de inclinación de cada línea. Para realizar el cálculo de la pendiente de las líneas azules se utiliza la siguiente fórmula aritmética:

$$m = \frac{y_2 - y_1}{x_2 - x_1} \quad (11)$$

Si el ángulo de inclinación del vehículo o de las líneas azules es mayor a 36.5° quiere decir que el vehículo está centrado, entonces le asignamos una velocidad relativamente alta y un

ángulo al volante que le permita estar alineado para que sea posible apreciar su desplazamiento. El vehículo está bien centrado cuando su ángulo de inclinación es de aproximadamente 38° , entonces aceptamos un error de 1.5° . Si el ángulo es menor o igual a 36.5° , modificamos el ángulo del volante para que quede centrado, así mismo disminuimos su velocidad para que tenga oportunidad de centrarse y no salirse del carril.

Algoritmo 6 Algoritmo para centrar el vehículo.

Entrada: Valor de las líneas.

```

1:
  Si lineas sea igual a 0.
    Hacer Imprimir mensaje.
  Si no
    Se crea una lista donde se almacena el angulo en
    grados de inclinacion de cada linea.
    Para cada linea en lineas Hacer
      Asignar los valores x1,x2,y1,y2.
      Calcular la pendiente.
      Calcular el angulo de inclinacion en grados.
      Agregar la pendiente a la lista.
      Regresar la lista.
    Se verifica que se tenga algun valor en la lista
    de angulos.
    Se almacenan los valores de mayor magnitud.
    Si el angulo es distinto a cero Hacer
      Agregar el angulo a la lista.
      Ordenar los valores de forma descendente.
      Si el valor absoluto es 36.5 Hacer
        Imprimir 'El vehiculo esta
        alineado '.
        Disminuir la velocidad.
        Desviar el coche 90 grados a
        la izquierda.
      Si no
        Imprimir 'El vehiculo no esta
        alineado '.
        Calcular el error.
        Publicar la velocidad y direccion
        del vehiculo.

```

Se puede observar el funcionamiento del controlador en ejecución con la simulación en un vehículo poco inclinado con respecto a las líneas de los carriles en la figura 14.



Figura 14: Se corrige la dirección del vehículo.

4.1.3. Detección de semáforos

El objetivo de este algoritmo es la detección de cualquiera de los tres colores que produce un semáforo vial, verde, ámbar o rojo. Como demostración se va a escoger el color verde y será identificado dentro de un círculo rosa, como se puede observar en la figura 15.



Figura 15: Luz verde del semáforo identificada por el círculo rosa.

Para obtener el código de color se puede hacer uso de alguna herramienta que se encuentre en la web, que sea capaz de proporcionar el código HSV(Hue, Saturation and Value). Se debe de tener en cuenta que no todos los colores verdes tienen el mismo código de color, por lo que se tiene que hacer una experimentación con el código obtenido para ajustarlo a las necesidades.

El proceso para obtener la imagen binaria donde aparece el color verde ya segmentado se realiza lo siguiente:

Algoritmo 7 Algoritmo para segmentar el color verde en la imagen.

Entrada: Imagen de entrada.

- 1: Se toma la imagen de entrada y se asignan los valores del color.
 - 2: Se elimina el ruido de la imagen de entrada por medio de una transformación morfológica.
 - 3: Se define el kernel que se va a utilizar.
 - 4: Se hace la visualización de la imagen ya filtrada.
-

En la figura 9 se observa que se han detectado dos elementos de color verde. Para tener una aproximación mayor a la realidad, lo que se hace es darle prioridad al semáforo que corresponde al carril en el que se transita, por lo que se implementa un algoritmo para detectar únicamente los elementos circulares, los cuales son resaltados con el color rosa.

Para detectar los elementos circulares se hace uso del algoritmo llamado “HoughCircles”. El proceso es mostrado a continuación.

Algoritmo 8 Algoritmo para detectar los elementos circulares.

Entrada: Imagen segmentada.

- 1:
Se toma la imagen segmentada y se convierte a escala de grises.
Se desenfoca la imagen.
Se obtienen las filas de las imagenes.
Se hace una transformacion circular de Hough con la imagen desenfocada.
Si existe algun circulo Hacer
 Para cada circulo en circulos Hacer
 Calcular el centro del circulo.
 Dibujar el centro del circulo.
 Calcular el radio del circulo.
 Dibujar el perimetro del circulo.
-

4.2. Detección de peatones

El detector de peatones se trata de un programa que hace uso de algoritmos como YOLOv3(You Only Look Once)[35]. Es importante tener en cuenta que se deben de realizar tres pasos importantes para hacer este procedimiento.

- Entrenar las imágenes para el modelo de reconocimiento de personas en YOLOv3.
- Cargar las imágenes en un servidor de Google.
- Probar el modelo generado en el paso anterior para ver si realmente funciona.

El primer paso es preparar el dataset, el cual contiene muchas imágenes del objeto al que interesa hacer reconocimiento. Algo importante es señalar que entre más imágenes existan,

y que sean diferentes(distintos escenarios, ángulos, iluminación, etc.), el resultado será mejor.

Una vez que ya se tienen las imágenes, es necesario hacer uso del programa gratuito LabelImg[36], este programa sirve para preparar las imágenes y tenerlas listas para entrenar el modelo de reconocimiento de objetos con YOLOv3. Una vez que se tenga este programa, se observa que existe un archivo ejecutable para poder abrir el programa, el cual muestra la interfaz, como se puede observar en las figuras 16 y 17, respectivamente.

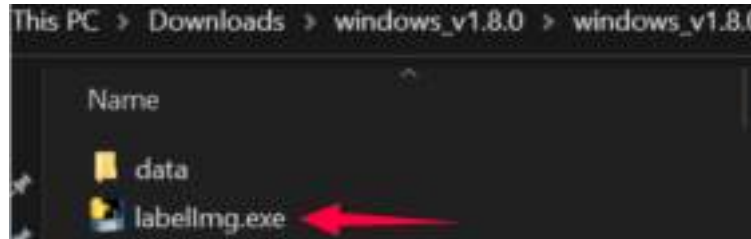


Figura 16: Archivo ejecutable del programa LabelImg.

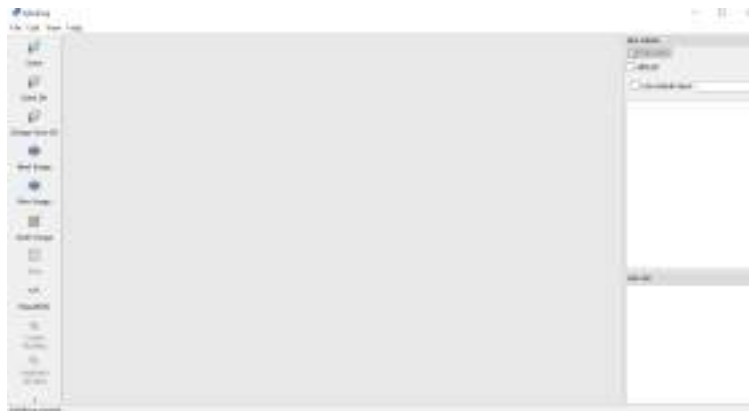


Figura 17: Interfaz del programa: LabelImg.

Una vez que se haya ejecutado este programa:

- Se selecciona el directorio donde se encuentran las imágenes.
- Se selecciona el directorio donde se guardarán los archivos para el modelo de entrenamiento.
- Se configura LabelImg para trabajar con el formato que requiere YOLOv3.

Con estos pasos es posible comenzar a preparar las imágenes. Para etiquetar una imagen de acuerdo a una categoría, el programa muestra una ventana con una lista de sugerencias. También permite crear una etiqueta en caso que el sujeto de interés no se encuentre en la lista, como se muestra en la figura 18.



Figura 18: Selección del objeto de interés.

Una vez que se haya hecho el etiquetado, se guarda la etiqueta con la relación con la imagen seleccionada. Se hace este proceso con cada imagen que se necesite. Para cada imagen se siguen estos pasos, se genera un archivo de texto, el cual está asociado a la imagen. Cuando se finaliza este proceso, se toman todas las imágenes con sus respectivos archivos de texto, y son comprimidos. Con esto se obtiene el dataset que se usará para entrenar el modelo de reconocimiento de personas en YOLOv3.

Posteriormente se hace uso de Google Colab, que es un servicio que ofrece Google para que los usuarios puedan usar GPUs gratuitas y listas en cualquier momento. Aquí se carga el archivo comprimido que se hizo anteriormente. También se utiliza un archivo con extensión “ipynb” para realizar el entrenamiento. Después de que ambos archivos se hayan cargado se realiza la ejecución. Una vez que se termina este proceso, se generan archivos “.weights”, el cual contiene información completa del entrenamiento. Finalmente se hace uso de un archivo de python llamado “pedestrian_detector” que es compatible con YOLOv3, se edita este archivo para configurarlo con los otros archivos que se han generado. Al ejecutar el programa aparecerá la ventana de figura 19 en la que se detecta a la persona que aparece en la imagen.



Figura 19: Resultado final del programa detector de personas.

4.3. Conclusión

Como parte final de este capítulo se presenta la conclusión a modo de discusión. El primer punto a comentar es la comodidad que se presenta al utilizar simuladores, y esto es un punto clave para cualquier estudiante o investigador, ya que no es necesario que se cuente con un presupuesto alto, o que se tenga que adquirir hardware específico para este tipo de tareas, como lo son los sensores, que usualmente son caros, y en comparación, el costo de una computadora es menor que el costo del hardware para un vehículo autónomo.

Por otro lado, hablando un poco sobre el simulador Gazebo, y las tareas realizadas durante el desarrollo de este proyecto, se pudieron observar distintas características, pero las que sobresalen y pueden ser consideradas como ventajas son las siguientes:

- Facilidad de conexión con ROS.
- El uso del motor de físicas ODE.
- Facilidad de importar desarrollo de terceros (Entre ellos se encuentran cosas como modelos hechos en CAD, Maya, Workbench, etc.).
- La posibilidad de implementar plugins.

Y finalmente, se pueden observar dos implementaciones prácticas de lo mostrado a lo largo de este documento. Existen dos proyectos, por parte de la UNAM se encuentra el proyecto UNAM-DGAPA con el número PAPIIT IA106520, donde el equipo de trabajo está conformado por tres integrantes, y también está un proyecto por parte de la UPV, que actualmente está en su primer año de desarrollo y se titula "Hacia los autos autónomos en calles de México: manejo a la defensiva en presencia de conducción imprudente, peatones y baches" donde el equipo de trabajo es todavía mayor. El proyecto de la UPV recibió como préstamo un vehículo a escala por parte de la UNAM, pero debido a la falta de algunas configuraciones no se ha podido probar algoritmos en dicho vehículo, todo el trabajo que se ha realizado ha sido hecho en simulador. Y en ambos proyectos, gracias al uso de simuladores que se ha podido continuar con el desarrollo algoritmos, además que se ha podido trabajar al mismo tiempo debido a que no es necesario que utilicen el hardware.

Cabe recalcar que no se está probando una hipótesis de un descubrimiento científico, si no que, se está probando un desarrollo tecnológico. Y en este documento se propone que el simulador y los ambientes van a ser útiles para el desarrollo de algoritmos de navegación autónoma. Esto se puede comprobar con los algoritmos que fueron implementados y descritos previamente.

5. Discusión

En este último capítulo del documento se va a hablar de las conclusiones obtenidas después de haber finalizado con el trabajo hecho durante el período de estadía, al igual que se menciona el trabajo a futuro para continuar con este proyecto.

5.1. Conclusión

Los vehículos terrestres son uno de los medios principales de transporte para la mayoría de las personas, y gracias a las tecnologías actuales se puede lograr la navegación autónoma por parte de estos, pero para obtener este objetivo es necesario que se hagan pruebas en distintos ambientes, ya que de lo contrario podría resultar en graves accidentes sobre la población.

Tomando esto en cuenta, se concluye que el uso de simuladores sí es útil en el desarrollo de vehículos sin conductor, pues facilita el trabajo en equipo y permite avanzar con el trabajo sin la necesidad de un equipo físico elaborado. Además de que se pudieron observar las cualidades con las que cuenta el simulador Gazebo, y se pudo comprobar el resultado de los algoritmos implementados en los mundos creados.

Y que con un sistema que simule los sensores y la cinemática del vehículo es suficiente para que el traslado de los desarrollos al robot real no sea tan complicado.

5.2. Trabajo a Futuro

Con el constante avance en las tecnologías, el trabajo a futuro puede involucrar distintas cosas, pero principalmente se tiene contemplado el uso de otros sensores, como lo serían los sensores sonares, o los sensores Lidar 3D, para que pueda realizar barridos en un plano a distintas alturas, para que de esta manera complementen los que fueron tomados en cuenta para este proyecto. También se puede seguir investigando o desarrollando otros simuladores, para tener un ambiente de pruebas que pueda resultar más útil para las personas involucradas.

Índice de figuras

1.	Figuras que se encuentran en el simulador Gazebo[31].	14
2.	Tipos de distribución[32]	15
3.	Textura del modelo de la carretera recta.	16
4.	Textura del modelo de la carretera curva.	16
5.	Umbral de histéresis[33].	18
6.	Sistema de coordenadas polares[34].	18
7.	Imágenes procesadas	20
8.	Imagen de los contornos de la sección de interés.	21
9.	Segmentación de los carriles de la carretera.	21
10.	Imagen original de la cámara del vehículo.	22
11.	Imagen binaria. En color blanco se muestra la segmentación del color de interés.	23
12.	Forma de triángulo de los carriles vistos desde la perspectiva de la cámara.	23
13.	Representación de las líneas de los carriles por medio de líneas rectas.	24
14.	Se corrige la dirección del vehículo.	26
15.	Luz verde del semáforo identificada por el círculo rosa.	26
16.	Archivo ejecutable del programa Labellmg.	28
17.	Interfaz del programa: Labellmg.	28
18.	Selección del objeto de interés.	29
19.	Resultado final del programa detector de personas.	29

Índice de tablas

1. Tabla comparativa de los antecedentes.	4
---	---

Índice de algoritmos

1.
 - para obtener los contornos de la imagen.212.
 - para obtener la imagen del vehículo.223.
 - para la segmentación de líneas.234.
 - para la transformada de Hough.245.
 - para dibujar las líneas de los carriles.246.
 - para centrar el vehículo.257.
 - para segmentar el color verde en la imagen.278.
 - para detectar los elementos circulares.27

Referencias

- [1] Euro NCAP. “2013 PRUEBAS DE FRENADO DE EMERGENCIA AUTÓNOMO”. En: (ene. de 2014).
- [2] Jesus Savage y col. “Semantic reasoning in service robots using expert systems”. En: *Robotics and Autonomous Systems* 114 (2019), págs. 77-92. ISSN: 0921-8890. DOI: <https://doi.org/10.1016/j.robot.2019.01.007>. URL: <https://www.sciencedirect.com/science/article/pii/S0921889018302501>.
- [3] Marco Negrete, Jesus Savage y Luis Angel Contreras-Toledo. “A Motion-Planning System for a Domestic Service Robot”. En: *SPIIRAS Proceedings* 5.60 (2018), págs. 5-38. DOI: [10.15622/sp.60.1](https://doi.org/10.15622/sp.60.1). URL: <http://proceedings.spiiras.nw.ru/index.php/sp/article/view/3932>.
- [4] Jesus Savage y col. “Construction of roadmaps for mobile robots’ navigation using rgb-d cameras”. En: *Intelligent Autonomous Systems 13*. Springer, 2016, págs. 217-229.
- [5] J. Savage y col. “Configurable Mobile Robot Behaviors Implemented on FPGA Based Architectures”. En: *2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. 2016, págs. 317-322. DOI: [10.1109/ICARSC.2016.29](https://doi.org/10.1109/ICARSC.2016.29).
- [6] General Motors Research Laboratories Report. “Automatic car controls for electronic highways.” En: (1960).
- [7] Varios Autores. *VEHÍCULOS DE GUIADO AUTÓNOMO (AGV) EN APLICACIONES INDUSTRIALES: UNA REVISIÓN*. <https://revistas.elpoli.edu.co/index.php/pol/article/view/1478/1208>. 2019.
- [8] IEEE Transactions on Control Systems Technology. “Robust two degree-of-freedom add-on controller design for automatic steering.” En: (2002).
- [9] Daniel F. Howard y Danielle Dai. “Public Perceptions of Self-Driving Cars: The Case of Berkeley, California”. En: 2014.
- [10] Barbara Lenz Hermann Winner Markus Maurer J. Christian Gerdes. *Autonomous Driving, Technical, Legal and Social Aspects*. SpringerOpen, 2015.
- [11] B. Paden y col. “A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles”. En: *IEEE Transactions on Intelligent Vehicles* 1.1 (2016), págs. 33-55.
- [12] Open Robotics. *About ROS*. <https://www.ros.org/about-ros/>. 2021.
- [13] Open Robotics. *Core Components*. <https://www.ros.org/core-components/>. 2021.
- [14] Jesus Savage y col. “Semantic reasoning in service robots using expert systems”. En: *Robotics and Autonomous Systems* 114 (abr. de 2019).
- [15] Marco Negrete, Jesus Savage y Luis Contreras-Toledo. “A Motion-Planning System for a Domestic Service Robot”. En: *SPIIRAS Proceedings* 5 (oct. de 2018), pág. 5. DOI: [10.15622/sp.60.1](https://doi.org/10.15622/sp.60.1).
- [16] AutoMiny. *About*. <https://autominy.github.io/AutoMiny/about/>. 2020.
- [17] Russ Smith. *Open Dynamics Engine*. <https://www.ode.org/>. 2004.

- [18] Epic Games. *Unreal Engine*. Ver. 4.22.1. 25 de abr. de 2019. URL: <https://www.unrealengine.com>.
- [19] Weichao Qiu y Alan Yuille. “UnrealCV: Connecting Computer Vision to Unreal Engine”. En: *Computer Vision – ECCV 2016 Workshops*. Ed. por Gang Hua y Hervé Jégou. Cham: Springer International Publishing, 2016, págs. 909-916. ISBN: 978-3-319-49409-8.
- [20] S. Carpin y col. “USARSim: a robot simulator for research and education”. En: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. 2007, págs. 1400-1405. DOI: [10.1109/ROBOT.2007.363180](https://doi.org/10.1109/ROBOT.2007.363180).
- [21] Jijun Wang, Lewis y Gennari. “A game engine based simulation of the NIST urban search and rescue arenas”. En: *Proceedings of the 2003 Winter Simulation Conference, 2003*. Vol. 1. 2003, 1039-1045 Vol.1. DOI: [10.1109/WSC.2003.1261528](https://doi.org/10.1109/WSC.2003.1261528).
- [22] Unity. *Plataforma de Unity*. 20 de feb. de 2021. URL: <https://unity.com/es/products/unity-platform>.
- [23] W. A. Mattingly y col. “Robot design using Unity for computer games and robotic simulations”. En: *2012 17th International Conference on Computer Games (CGAMES)*. 2012, págs. 56-59. DOI: [10.1109/CGames.2012.6314552](https://doi.org/10.1109/CGames.2012.6314552).
- [24] Christoph Bartneck y col. “The robot engine — Making the unity 3D game engine work for HRI”. En: ago. de 2015. DOI: [10.1109/ROMAN.2015.7333561](https://doi.org/10.1109/ROMAN.2015.7333561).
- [25] Yunwang Li y col. “Navigation Simulation of a Mecanum Wheel Mobile Robot Based on an Improved A* Algorithm in Unity3D”. En: *Sensors* 19 (jul. de 2019), pág. 2976. DOI: [10.3390/s19132976](https://doi.org/10.3390/s19132976).
- [26] Georgia Tech y Carnegie Mellon University. *Overview*. <http://dartsim.github.io/>. 2020.
- [27] Varios Autores. *Simbody: Multibody Physics API*. 2021.
- [28] Varios Autores. *Why Gazebo?* <http://gazebo.org/>. 2020.
- [29] Humberto Loaiza-Correa y andres felipe suarez sanchez. “Esquema de Navegación Reactiva con Sensores RGB-D”. En: *UIS Ingenierías* 14 (ene. de 2015), pág. 7.
- [30] Varios Autores. *¿Cómo funciona el LiDAR?* <https://www.yellowscan-lidar.com/es/knowledge/how-lidar-works/>. 2018.
- [31] Gazebo. *Building a world*. http://gazebo.org/tutorials?tut=build_world. 2014.
- [32] Gazebo. *Population of models*. http://gazebo.org/tutorials?tut=model_population&cat=build_world. 2014.
- [33] OpenCV. *Canny Edge Detection*. https://docs.opencv.org/master/da/d22/tutorial_py_canny.html. 2021.
- [34] OpenCV. *Hough Line Transform*. 2021.
- [35] Joseph Redmon y Ali Farhadi. “YOLOv3: An Incremental Improvement”. En: *arXiv* (2018).
- [36] Tzutalin. *LabelImg*. <https://github.com/tzutalin/labelImg>. 2015.