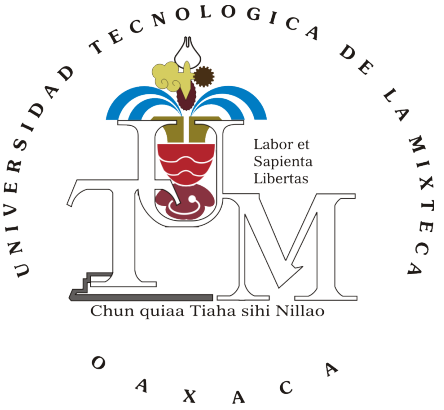


UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

DIVISIÓN DE ESTUDIOS DE POSGRADO

MAESTRÍA EN ROBÓTICA

GENERACIÓN DEL MAPA MÉTRICO DE UNA PISTA DE COMPETENCIAS PARA LA NAVEGACIÓN DEL ROBOT MÓVIL AUTONOMOS MINI V2

TESIS

PARA OBTENER EL GRADO DE

MAESTRA EN ROBÓTICA

PRESENTA:

ING. ESTHER GUADALUPE DIAZ SARMIENTOS

DIRECTOR DE TESIS:

DR. ALBERTO ELÍAS PETRILLI BARCELÓ

CO-DIRECTOR DE TESIS:

DR. JUAN HUMBERTO SOSSA AZUELA

HUAJUAPAN DE LEÓN, OAXACA, MÉXICO, OCTUBRE DE 2020

Tesis presentada en octubre de 2020 ante los sinodales:

- Dr. José Anibal Arias Aguilar
  Profesor-Investigador UTM
- Dr. Rosebet Miranda Luna
  Profesor-Investigador UTM
- Dr. Arturo Téllez Velázquez
  Profesor-Investigador UTM
- Dr. Rogelio de Jesús Portillo Vélez
  Profesor-investigador Universidad Veracruzana

Dedicatoria

Dedicado a Dios, a mi madre Esperanza, a mi padre Jesus, a mi amor Mauricio, a mi familia y a mis amigos.

Agradecimientos

A Dios por darme vida, sabiduría y fortaleza para realizar esta meta.

A mi madre por apoyarme siempre, por sus consejos, enseñanzas y hasta regaños. Se que su anhelo más grande es ver a sus hijos cumplir sus sueños.

A mi padre, mi ángel guardián que me cuida desde el cielo y que se me acompaña siempre.

A Mauricio, mi compañero de vida, mi apoyo constante y mi razón para seguir. Pasamos por tanto, desde el inicio hasta el final, a pesar de la distancia y los problemas no me dejaste rendirme.

A mis hermanos, Jesús, Neftalí, Soraya y Yahir, cada uno a su manera me impulsan a ser mejor cada día. A mi sobrina, mi niña que aunque lejos, siempre en mis pensamientos.

Al Dr. Alberto Elías Petrilli Barceló por su apoyo y conocimientos en la realización de este trabajo y en toda la maestría. Le agradezco su tiempo, sus enseñanzas y consejos, así como la confianza en mi capacidad para realizar este proyecto.

Al Dr. Humberto Sossa Azuela por recibirme en su laboratorio, darme todas las facilidades para realizar mis estancias y por compartirme un poco de lo mucho que sabe.

A los sinodales, Dr. José Anibal Arias Aguilar, Dr. Rosebet Miranda Luna, Dr. Arturo Téllez Velázquez y Dr. Rogelio de Jesús Portillo Vélez, por su tiempo para la revisión de este trabajo, por sus consejos y enseñanzas.

A Berith y Yubesny, su amistad es de lo más preciado que obtuve en esta maestría. Gracias por su apoyo y por siempre tener tiempo para escucharme. He aprendido mucho de ustedes y espero seguir haciéndolo. Sin duda alguna las mañanas de café, las largas pláticas y las anécdotas vividas hicieron más ameno el día a día.

A mis compañeros de generación, compañeros de laboratorio y a todos los amigos que conocí en el transcurso de la maestría. Gracias por permitirme aprender de ustedes y por los buenos momentos compartidos.

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

Al Centro de Investigación en Computación del IPN por recibirme en sus instalaciones y permitirme desarrollar parte de este trabajo.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por el apoyo económico brindado para mis estudios de maestría y la realización de una estancia nacional.

Resumen

Este trabajo describe el desarrollo de un algoritmo para la reconstrucción de un mapa métrico generado a partir de las imágenes capturadas por la cámara Intel Real Sense a bordo del robot AutoNOMOS mini. Se presenta primeramente el pre-procesamiento de las imágenes, así como su corrección de perspectiva. De igual forma se describen dos técnicas para detección de líneas de carril, diferenciadas en su método y aplicación. Se presenta también un algoritmo para la asociación de componentes en imágenes consecutivas, así como un algoritmo para el cálculo de la matriz de transformación entre dos conjuntos de puntos. Finalmente, usando la información combinada de asociación de datos con la estimación de la orientación del robot AutoNOMOS, se reconstruye un mapa de una pista real y de un entorno virtual.

Abstract

This work describes the development of an algorithm for the recovery of a metric map generated from images captured by the Intel Real Sense camera on board of the AutoNOMOS mini robot. The image processing is presented first, as well its a perspective correction. Similarly, two techniques for detecting lane lines are also described, differentiated in their method and application.An algorithm for the association of components in consecutive images is also presented, as well as an algorithm for the calculation of the transformation matrix between two sets of points. Finally, using the combined information and data association with the estimation of the orientation of the AutoNOMOS robot, a map is reconstructed.

Índice

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

La presencia de la robótica en diversas áreas del mundo actual, desde la producción industrial hasta actividades cotidianas, ha tenido un crecimiento acelerado en los últimos años. La industria automotriz no ha estado exenta de este desarrollo, ya que actualmente se han integrado en nuevos modelos de automóviles mejores sensores, controles de velocidad y asistentes de estacionamiento así como Sistemas Avanzados de Asistencia a la Conducción(ADAS).

La realización de competencias tanto con vehículos autónomos reales como a escala constituye una excelente oportunidad de desarrollo de tecnologías que contribuyen al avance en la industria automotriz. Estas competencias involucran tareas a resolver sin intervención humana.

Una parte importante de la autonomía de un vehículo, o robot en general, es el conocimiento del entorno. Esto permite la planeación para navegación y realización de tareas. Los sensores son los que proporcionan la información del entorno siendo las cámaras muy utilizado para ello. El ambiente de trabajo del robot puede ser conocido a priori o ser sensado mientras durante la navegación.

Los mapas constituyen una herramienta importante en el conocimiento del entorno y permiten planeación de rutas, localización y navegación. Dentro de los mapas existen clasificaciones, siendo los mapas métricos menos complejos en cuanto a construcción ya que se constituyen de información espacial basados en formas básicas como líneas y curvas.

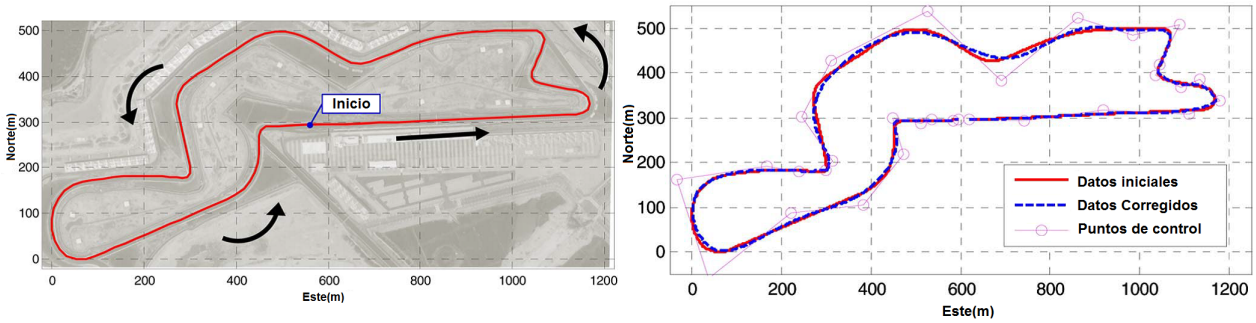
En este documento se describe el desarrollo de una metodología para la reconstrucción de una pista de competencias por medio de un mapa de tipo métrico. Incluye las etapas de detección de carriles utilizando dos técnicas diferentes: método de ventanas y detección de componentes. También la descripción de la asociación de imágenes por medio de la detección de esquinas y por último la reconstrucción de mapa utilizando las transformaciones sucesivas de imágenes consecutivas, combinando información de imágenes y medición de orientación.

1.1. Antecedentes

En los últimos años se ha acrecentado el interés en los vehículos autónomos con el fin de reducir riesgos de accidentes, disminuir la contaminación, aumentar la eficiencia en los transportes, entre otros beneficios. El desarrollo de esta tecnología tiene como aliciente que compañías como Google buscan poner al alcance del público estos vehículos.

Uno de los temas que en este ámbito ha recibido mucha atención es la detección de los carriles en carreteras donde se han implementado herramientas como redes neuronales y filtros estadísticos ((Aly, 2008), (López et al., 2010), (Guo et al., 2010), (Bae and Song, 2011) y (Küçükyildiz and Ocak, 2014)). La construcción de mapas es un tema relevante en el desarrollo de vehículos autónomos, de forma separada o en técnicas de SLAM(Mapeo y Localización simultáneos). En esta área, (Thrun et al., 2002) es un referente importante con una investigación amplia en cuanto a tipos de mapas. Thrun describe de forma amplia los distintos tipos de algoritmos en la generación de mapas donde destacan los descritos por medio del filtro de Kalman, ocupación de rejillas y basados en objetos.

En (Jo and Sunwoo, 2014) se realiza la reconstrucción de carreteras usando el GPS y los sensores a bordo de orientación y velocidad, utilizando *B-splines*. Cabe señalar que *B-spline* es la representación de una curva construida por puntos de control. (Jo and Sunwoo, 2014) dividen el algoritmo de generación del mapa en tres etapas: adquisición de datos, fusión de datos de los distintos sensores y reconstrucción de carretera. Para fusionar las mediciones se utiliza un filtro estadístico llamado RTS(Rauch-Tung-Striebel), cuya linealización es idéntica a la usada en el filtro de Kalman extendido. La figura 1.1 muestra el resultado de la generación del mapa.



(a) Modelo Real

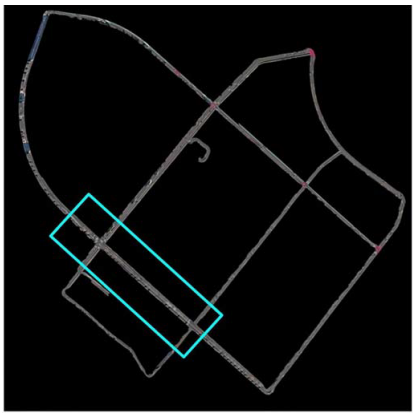
(b) Modelo obtenido

Figura 1.1: Generación de un mapa usando *B-splines* (Jo and Sunwoo, 2014).

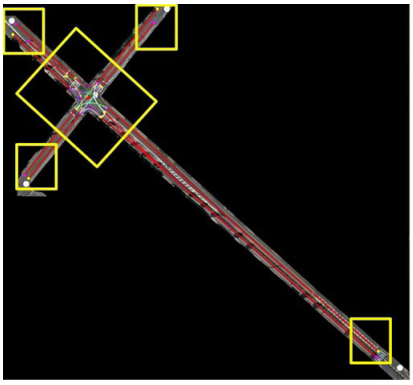
(Guo et al., 2016) propone un sistema de dos módulos para la reconstrucción de carreteras por medio de curvas clotoides. El primer módulo determina los segmentos de carretera a recorrerse para la adquisición de imágenes y datos, a partir de un mapa global. Las imágenes obtenidas son transformadas con homografía, para ser usadas en el segundo módulo que corresponde a la reconstrucción de los carriles. La recopilación de datos se llevo a cabo usando

tres vehículos equipados con un GPS de bajo costo y una cámara CMOS. Una porción del mapa generado por este trabajo se muestra en la figura 1.2

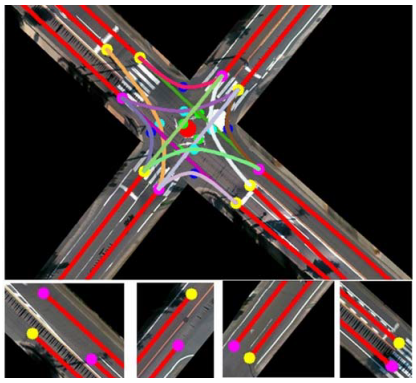
(a) Trayectorias del vehículo.



(b) Imágen generada.



(c) Ampliación del área cian.



(d) Ampliación de las areas amarillas.

Figura 1.2: Resultados de (Guo et al., 2016).

En (Gwon et al., 2017) se propone el uso de un sensor láser Lidar 3D en combinación con un GPS para la reconstrucción de carreteras. Proponen un proceso de generación de mapas de tres pasos: 1) adquisición de datos, 2) procesamiento de datos y 3) reconstrucción de carreteras. Para la última etapa se utilizaron aproximaciones polinomiales por segmentos con 12 coeficientes, haciendo uso del filtro de Kalman, cuyo desempeño fue mejor que el obtenido con B-splines y clotoides.

El sistema presentado en (Joshi and James, 2015) combina información a priori obtenida de mapas del proyecto Open Street, con información de un sensor Lidar. Su primer paso es la localización del vehículo empleando técnicas de SLAM con información de un sensor GPS. El algoritmo de estimación y reconstrucción de carriles corresponde a la segunda etapa y se divide en tres fases: preprocesamiento de los datos y obtención de marcadores, estimación de distribución usando un filtro de partículas y estimación a posteriori del número y posición de los carriles. Muestra de los resultados cualitativos obtenidos en Joshi and James (2015) se

pueden observar en la figura 1.3 donde las líneas rojas representan los centros de los carriles y las otras líneas, marcadores de los carriles.

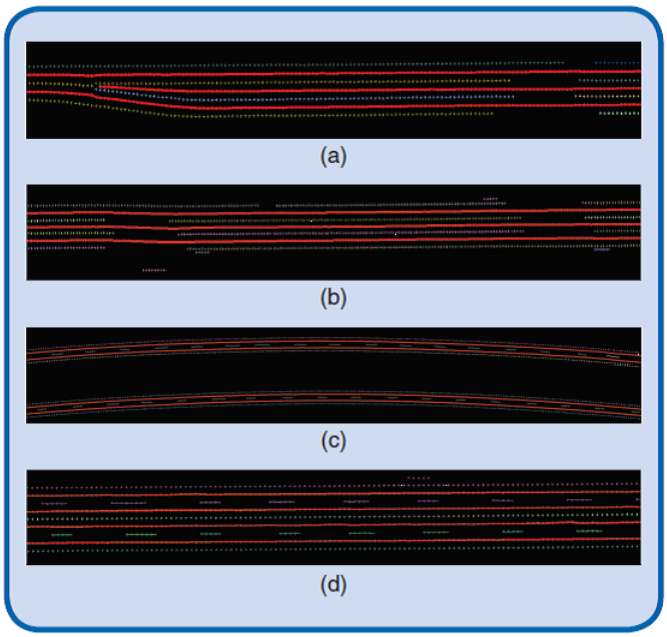


Figura 1.3: Resultados cualitativos obtenidos en (Joshi and James, 2015).

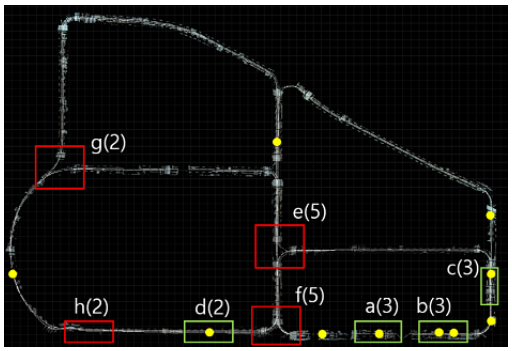
(Jeong et al., 2017) utiliza submapas que contienen líneas de carril para la construcción de un mapa global, como parte del algoritmo Road-SLAM. En su procesamiento utiliza técnicas de corrección de perspectiva, binarización y división de nube de puntos, así como un clasificador *Random Forest* que divide todas las marcas presentes en las imágenes en seis clases(marcas del camino, números, flechas, carriles, cruces de peatones y otros). En la figura 1.4 puede verse la comparación entre el mapa real y el obtenido por este método.

Como parte importante de la localización de un vehículo (Vivacqua et al., 2018) realiza la reconstrucción de carreteras tomando en cuenta las marcas visuales de los carriles. Este resulta de la combinación de la detección de las lineas de carriles con un mapa global conocido. Para su construcción se considera información inercial del vehículo y de GNSS (Sistema de Satélites de Navegación Global). El mapa de este trabajo se realiza con referencia al vehículo. En la figura 1.5 se observa el resultado de este trabajo.

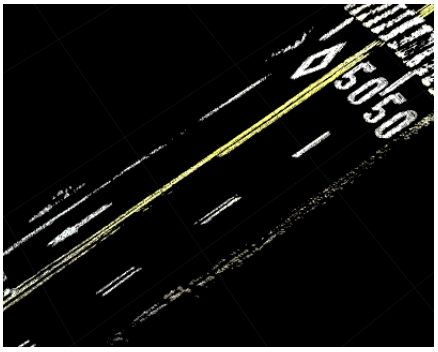
Los sistemas de visión son una herramienta importante en la generación de mapas. (Bender et al., 2014) generó una librería denominada *lanelets*, cuyo enfoque es una combinación entre mapas topológicos y métricos. Los *lanelets* son pequeños segmentos de carretera con información, que unidos son usados para modelarlas.

Las competencias de vehículos a escala tienen como elemento importante el recorrido de un circuito completo sin salirse de las líneas delimitadoras del carril. Una competencia importante en este ámbito es la copa Carolo realizada en la Technische Universität Braunschweig de Alemania cada año. En el trabajo de Karouach and Ivanov (2016) hacen un recuento de

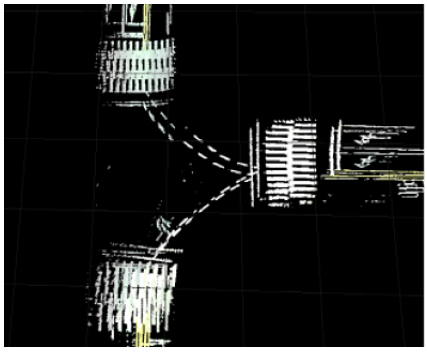
(a) Imagen aérea de la sección mapeada.



(b) Mapa construido con submapas.



(c) Sección (a) del mapa estimado.



(d) Sección (e) del mapa estimado.

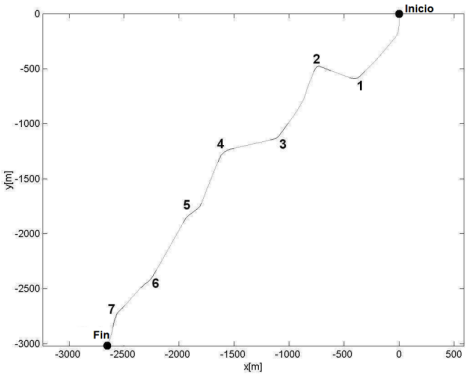
Figura 1.4: Construcción de mapa de (Jeong et al., 2017).

los algoritmos utilizados por cada equipo en la competencia del 2016, tal como se enlista en la tabla 1.1.

(Krüger et al., 2015) implementaron un reconocimiento de carriles con *SequentialRansac*. Como el mismo autor lo enuncia, un requisito importante es que el vehículo reconozca y obtenga una representación de los carriles, para seguirlos y participar en cualquiera de las categorías de la copa Carolo. Dentro de esta misma competencia, (Rehder and Albrecht, 2015) desarrollo una metodología para la reconstrucción de un circuito utilizando un mapa métrico basado en ocupación de rejillas. Su sistema realiza construcción de un mapa local por medio de la estimación de movimiento(desplazamiento y orientación) y la extracción de características. En la primera etapa utiliza un Filtro de Kalman Extendido para fusionar las mediciones de encoders, giroscopios y acelerómetros, mientras que en la segunda utilizando regiones de interés, umbralización y un filtro de mediana obtiene las marcas de carriles de las imágenes. Los mapas locales son agregados a un mapa global por medio de un algoritmo de asociación de datos y una optimización utilizando los errores concatenados de odometría y medición. Los resultados de este sistema pueden verse en la figura 1.6.

En México, dentro del TMR (Torneo Mexicano de Robótica) y el evento Talent, Land existe una categoría denominada AutoModelCar. Dicha categoría contempla coches a escala que resuelven pruebas de recorridos de circuitos completos con y sin obstáculos. (Bravo Co-

(a) Mapa base obtenido de Google Maps.

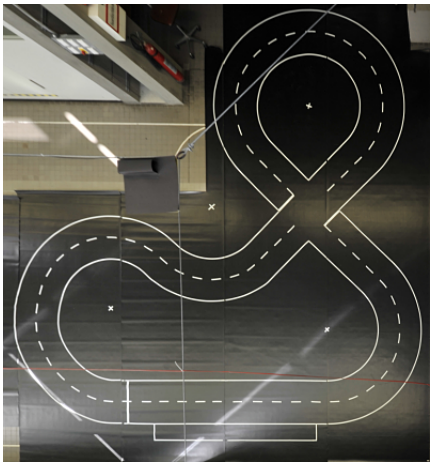


(b) Mapa construido por el sistema de estimacion.

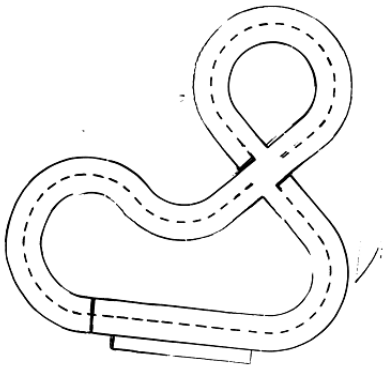
(c) Imagen satelital de la curva 2.

(d) Mapa generado a partir del sistema de visión.

Figura 1.5: Resultados de(Vivacqua et al., 2018).



(a) Pista real.



(b) Mapa construido utilizando submapas.

Figura 1.6: Comparación entre imagen real y mapa construido en (Rehder and Albrecht, 2015).

nejo, 2018) desarrollo un sistema de navegación para la plataforma AutoNOMOS mini V1, donde realiza detección de carriles, generación y seguimiento de rutas, detección y evasión de obstáculos y estacionamiento en paralelo. Dentro de la detección de carriles utiliza una técnica basada en mínimos locales para identificar las líneas, posteriormente reconstruye el carril utilizando regresión no lineal.

(Vanegas Sánchez, 2018) diseñó una metodología basada en redes neuronales convolucionales para el control de dirección de la plataforma AutoNOMOS mini V1 en ambientes reales y virtuales. Estos dos trabajos ((Bravo Conejo, 2018) y (Vanegas Sánchez, 2018)) corresponden a desarrollos centrados solo en la navegación.

Equipo	Pre-procesamiento de la imagen	Extracción de características	Validación de características	Cálculo de trayectoria
Berlin United	Escala de grises Región de interés Mapeo perspectivo inverso	Detección de bordes	RANSAC	Punto objetivo
CDLC	Escala de grises Región de interés Mapeo perspectivo inverso	Detección de puntos	RANSAC Ajuste polinomial	Punto objetivo
eWolf	Escala de grises	Detección de bordes	?	?
GalaXIs	Escala de grises Correción vista esférica a panorámica. Mapeo perspectivo inverso	Detección de bordes	Ajuste polinomial	Línea central
ISF Löwen	Escala de grises Corrección vista esférica a panorámica. Mapeo perspectivo inverso	Detección de bordes	Ajuste geométrico	Punto objetivo
it:movES	Escala de grises Región de interés	Detección de bordes	?	?
KITCar	Escala de grises Imagen binaria	Escaneo de líneas	Ajuste polinomial	Línea central
NaN	Escala de grises Imagen binaria(OTSU)	?	Ajuste polinomial	Punto objetivo
Ostfalia	Escala de grises Contraste brillo Región de interés	Detección de bordes	Ajuste geométrico	Punto objetivo
Pegasus	Escala de grises Región de interés Imagen binaria	Escaneo de líneas	Ajuste geométrico	Punto de fuga
TUM Phoenix	Escala de grises Mapeo perspectivo inverso	Escaneo de líneas	Ajuste geométrico Filtros de Kalman	Punto objetivo

Tabla 1.1: Técnicas de detección y seguimiento de carril usados por los equipos participantes en la Copa Carolo 2016 (Karouach and Ivanov, 2016).

1.2. Planteamiento del problema

En la conducción de vehículos autónomos contar con información del entorno, es fundamental para la navegación y localización. La mejor forma de presentar esta información es por medio de un mapa a nivel de carreteras.

Tanto los sistemas de conducción asistida como la planeación de trayectorias necesitan conocer el entorno para generar una ruta. Un mapa proporciona información de la geometría del camino necesaria para que un algoritmo de navegación siga cualquier ruta.

En las competencias a escala de vehículos autónomos, conocer el entorno es base para realizar recorridos sin salirse del carril. En específico, para la navegación de la plataforma AutoNOMOS mini V2 en una pista de competencias, como la mostrada en la figura 1.7, el no tener conocimiento previo de la forma y tamaño de ella dificulta la planeación de trayectorias. Para competir en cualquier categoría que involucre navegación y localización es necesario reconocer los carriles y seguirlos adecuadamente. Así mismo si se tiene un mapa se pueden tomar mejores decisiones en cuanto a velocidad y dirección a seguir al trasladarse en la pista. Atendiendo a esta necesidad se plantea diseñar un algoritmo para construir un mapa usando información visual del sensor Real Sense y la orientación del vehículo del sensor IMU en la plataforma AutoNOMOS.

Figura 1.7: Pista de competencias.

1.3. Justificación

El mapeo forma parte de las áreas fundamentales de la navegación de un robot móvil, contar con conocimiento del entorno donde se mueve es primordial para la realización de cualquier tarea.

La detección de carriles en carreteras forma parte de las tecnologías de asistencia para conducción de automóviles, que ha tenido un gran desarrollo durante los últimos 15 años (Arce et al., 2017). Actualmente estos sistemas alertan a los conductores sobre errores en el seguimiento de carriles, aunque pueden ser aplicados en el seguimiento de trayectorias.

La participación en competencias a pequeña escala, cuyas pistas simulan ser carreteras, permite el desarrollo de tecnología en las áreas fundamentales de la navegación de coches autónomos que puede ser aplicado a dimensiones reales.

Para el modelado de cualquier entorno pueden usarse sensores como láseres, GPS y cámaras. El uso de estas últimas proporciona mayor información, comparado con otros tipos de sensores. Generar un mapa a partir de la detección de carriles usando cámaras, permite reducir errores en seguimiento de trayectorias ya que provee una referencia, así mismo permite recorridos más eficientes.

La plataforma AutoNOMOS Mini v2 cuenta con la computadora a bordo (Odroid XU4) y el sensor Real Sense 300 que hacen factible la implementación y puesta en marcha de los algoritmos que se desarrollen. Esto proporcionará una base para trabajos futuros relacionados con localización y seguimiento de trayectorias.

El desarrollo de este trabajo de investigación contribuye al desarrollo de tecnología mexicana en el campo de los automóviles y robots autónomos.

1.4. Hipótesis

Es posible la elaboración de un mapa métrico de una pista de competencias, de manera automática, usando la información de los sensores del robot AutoNOMOS adquirida durante un recorrido teleoperado de dicha pista.

1.5. Objetivos

1.5.1. General

Implementar una metodología para la generación de un mapa métrico de una pista de competencias en la plataforma AutoNOMOS mini V2, usando visión artificial.

1.5.2. Específicos

1. Desarrollar e implementar un algoritmo para el pre-procesamiento de las imágenes obtenidas del sensor Real Sense.

2. Desarrollar e implementar algoritmos para identificación de líneas de los carriles de forma automática.

3. Desarrollar un algoritmo de asociación de datos para encontrar las transformaciones geométricas entre imágenes sucesivas.

4. Desarrollar una metodología para obtener un mapa de la pista de competencias.

1.6. Metodología

En los trabajos revisados del área de mapeo, los autores dividen, generalmente, sus procesos en pre-procesamiento, extracción de características y reconstrucción del mapa. Con base en esta observación se propone la metodología para el desarrollo de este trabajo tal como se muestra en la figura 1.8. Cada etapa es descrita a continuación:

- **Fase inicial:** Revisar y comprender el funcionamiento de la plataforma y de los elementos de la misma que serán utilizados.

- **Adquisición de imágenes:** Esta etapa comprende conocer las características de las imágenes y como realizar la captura cuando la plataforma esté en movimiento.

- **Pre-procesamiento de imágenes:** Dar tratamiento a las imágenes obtenidas de la cámara Intel Real Sense 300 para resaltar la información de interés, con técnicas como escala de grises, binarización, aplicación de filtros, entre otras. Esta etapa también incluye la corrección de perspectiva de la imagen.

- **Segmentación:** En esta fase del trabajo se detecta la información de importancia en la imagen.

- **Detección de carriles:** En cada imagen se identifican los carriles presentes por medio de un detector de bordes y/o contornos, así como la clasificación de pixeles según la línea que le corresponda.

- **Asociación de datos:** Esta etapa comprende el diseño del algoritmo para determinar la correspondencia de puntos en distintas imágenes tomadas de forma consecutiva.

- **Generación del mapa:** En esta penúltima etapa se reconstruye una imagen con la descripción de la pista.

- **Pruebas básicas:** En esta etapa se realizar pruebas básicas que se refieren a validaciones visuales, de tipo cualitativas, del mapa obtenido. Dichas pruebas consistirán en comparación del mapa obtenido con el modelo real.

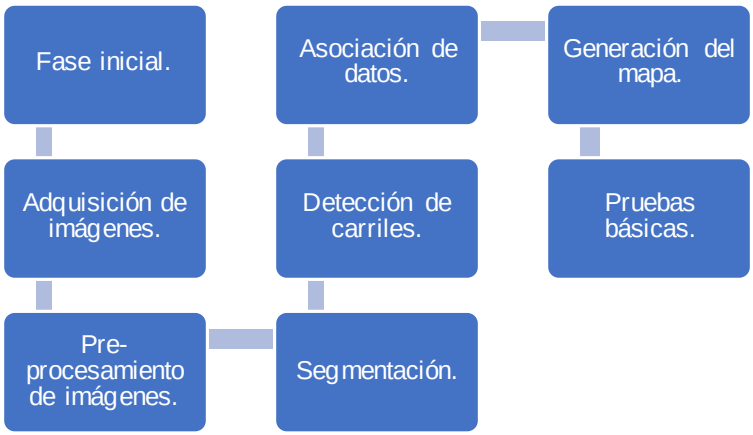


Figura 1.8: Metodología a desarrollarse

1.7. Delimitaciones

Para la realización de este proyecto se tomaron las siguientes consideraciones:

- Se implementó en la plataforma AutoNOMOS mini V2, en la cual se usará el sensor Real Sense y la unidad de medición inercial para adquirir información que servirá para el reconocimiento del ambiente.
- La pista de competencias sobre la que se trabajó está en interiores, sin cambios de iluminación y se considerará como un entorno estático. Dicha pista tiene un fondo negro, líneas blancas y dos carriles.
- La reconstrucción se realiza mientras el robot es teleoperado por medio de un nodo en ROS.
- La reconstrucción del mapa desde el sensado hasta el resultado final se realiza en tiempo real.
- Para hacer la reconstrucción del mapa se necesita que el robot realice, como mínimo, una vuelta completa a la pista.
- Se utilizaran imágenes a color RGB obtenidas de la cámara Intel Real Sense 300.

1.8. Estructura de la tesis

Este trabajo de tesis está organizado de la siguiente forma: en el Capítulo 1 se presenta la introducción y los antecedentes de esta investigación, se describe el planteamiento del problema y su justificación, se especifican los objetivos a cumplir para demostrar la hipótesis planteada y se describe la metodología seguida en el desarrollo. En el Capítulo 2 se describen los conceptos y bases teóricas necesarios para este trabajo de tesis. En el Capítulo 3 se describe el desarrollo de la metodología por etapas llevada a cabo para la reconstrucción del mapa. En el Capítulo 4 se muestran los resultados obtenidos de la reconstrucción de un mapa en tres ambientes virtuales y una pista real. En el Capítulo 6 se dan a conocer las conclusiones generales obtenidas del trabajo desarrollado y se mencionan posibles trabajos futuros.

Capítulo 2

Marco teórico

Los coches autónomos han tenido un gran desarrollo en los últimos años y su crecimiento hacia el futuro puede vislumbrarse fácilmente. Los Sistemas Avanzados de Asistencia a la Conducción (ADAS), presentes en modelos de automóviles con cierto nivel de autonomía, dependen del conocimiento del entorno que los sensores puedan proporcionarle. El reconocimiento del entorno usando imágenes es parte importante de ADAS, por lo que las cámaras están presentes en estos sistemas.

El desarrollo de técnicas para construcción de mapas es parte importante del desarrollo de coches autónomos. Los prototipos a escala equipados con sensores como es el caso del AutoNOMOS Mini V2 permiten la investigación y desarrollo de estas técnicas. Y esta idea es base para esta propuesta, por lo que a continuación se describen los conceptos necesarios para su realización.

2.1. Visión artificial y procesamiento de imágenes

El objetivo de la visión artificial es obtener información del entorno utilizando el procesamiento de imágenes para identificar objetos de interés y su posición en un ambiente. La visión artificial está estrechamente relacionada con el procesamiento de imágenes, que a pesar de compartir información en común sus objetivos son diferentes.

El procesamiento de imágenes se encarga del mejoramiento de la calidad de las imágenes para su posterior interpretación, por ejemplo (Sucar and Gómez, 2011):

- remover defectos,
- remover problemas por movimiento o desenfoque,
- mejorar ciertas propiedades como color, contraste, estructura, etc.
- agregar “colores falsos” a imágenes monocromáticas.

Por el contrario, la visión computacional se encarga de extraer características de una imagen para su descripción e interpretación.

El procesamiento de imágenes es un campo de estudio considerablemente extenso el cual comprende una gran cantidad de funciones como son:

- Filtrado.
- Correcciones de color.
- Binarización por umbral.
- Operaciones morfológicas.
- Transformaciones geométricas euclidianas.

2.2. Transformaciones proyectivas 2D

La geometría proyectiva 2D (Hartley and Zisserman, 2003) es el estudio de las propiedades del plano proyectivo $\mathbb{P}^2$ que son invariantes bajo un grupo de transformaciones conocidas como proyectividades.

Una proyectividad es una transformación invertible dada por $H = \mathbb{P}^2 \rightarrow \mathbb{P}^2$ de manera tal que una linea recta es transformada a una linea recta. La ecuación 2.1 representa una transformación proyectiva planar donde H es una matriz 3×3 no singular. La ecuación 2.1 esta definida de forma explícita en la ecuación 2.2.

$$h(x) = x' = Hx \tag{2.1}$$

$$\begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{2.2}$$

A partir de la ecuación 2.1 se dice que x' es la transformación lineal H de x. Esta transformación tiene una correspondencia uno a uno, es decir, a cada punto en un primer plano le corresponde solo un punto de un segundo plano, y viceversa. Dada la forma de la ecuación 2.2 si H es multiplicada por un valor constante $k \neq 0$, lo resultados de la transformación no varían, lo cual indica que H es una matriz homogénea. Por tanto si se divide toda la matriz entre el elemento h_{33} la transformación H se define solo por 8 parámetros.

A continuación se describe la categorización de las transformaciones proyectivas según sus propiedades geométricas.

Isometrías

Son transformaciones en el plano $\mathbb{R}^2$ que conservan la distancia euclidiana. Una transformación de tipo isométrica se representa con la ecuación 2.3, donde $\epsilon = \pm 1$, θ el ángulo de rotación y (t_x, t_y) el desplazamiento del origen.

$$\begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} = \begin{bmatrix} \epsilon cos(\theta) & -\epsilon sen(\theta) & t_x \\ \epsilon sen(\theta) & \epsilon cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \\ x_3 \end{bmatrix} \tag{2.3}$$

Si $\epsilon = 1$ la isometría conserva la orientación y es una transformación euclidiana (compuesta de una traslación y rotación). Si $\epsilon = -1$ entonces la isometría invierte la orientación, por ejemplo en una reflexión.

Una transformación euclidiana modela el movimiento de un objeto rígido. Esta puede ser reescrita en forma de bloques con la ecuación 2.4 donde R es una matriz de rotación orto normal de 2×2 y t un vector de traslación de 2 dimensiones. Esta transformación tiene 3 grados de libertad, uno de rotación y dos de traslación.

$$x' = H_E x = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} x \tag{2.4}$$

Las isometrías mantienen invariantes las propiedades de longitud (distancia entre dos puntos), angulo (entre dos líneas) y área.

Transformaciones de similitud

Estas transformaciones son una isometría con un escalamiento. La ecuación 2.5 representa esta transformación donde s representa el escalamiento.

$$x' = H_s x = \begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix} x \tag{2.5}$$

La transformación de similitud posee 4 grados de libertad, los mismos de la isometría más el escalamiento. Esta transformación preserva los ángulos entre lineas, las líneas paralelas, la proporción entre distancias y entre áreas.

Transformación afín

La transformación de este tipo esta compuesta de una transformación lineal no singular seguida de una traslación. La ecuación 2.6 muestra su representación, donde A es una matriz no singular de 2×2.

$$x' = H_A x = \begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix} x \tag{2.6}$$

Esta transformación tiene 6 grados de libertad, 4 pertenecen a la matriz A y 2 más al vector de traslación. Se mantienen invariantes las líneas paralelas, la razón entre áreas y la razón entre la longitud de líneas paralelas.

Transformación proyectiva general

Esta transformación esta definida en la ecuación 2.2. Se define como una transformación no lineal de coordenadas homogéneas.

En forma de bloques la transformación proyectiva tiene la forma de la ecuación 2.7, donde $v = (v_1, v_2)^T$.

$$x' = H_P x = \begin{bmatrix} A & t \\ v^T & v \end{bmatrix} x \tag{2.7}$$

Si bien la matriz H_p tiene 9 elementos, estos pueden reducirse a 8 siempre que v no sea 0.

La invariancia más fundamental de esta transformación es la relación cruzada de cuatro puntos colineales.

2.3. Calibración de cámaras

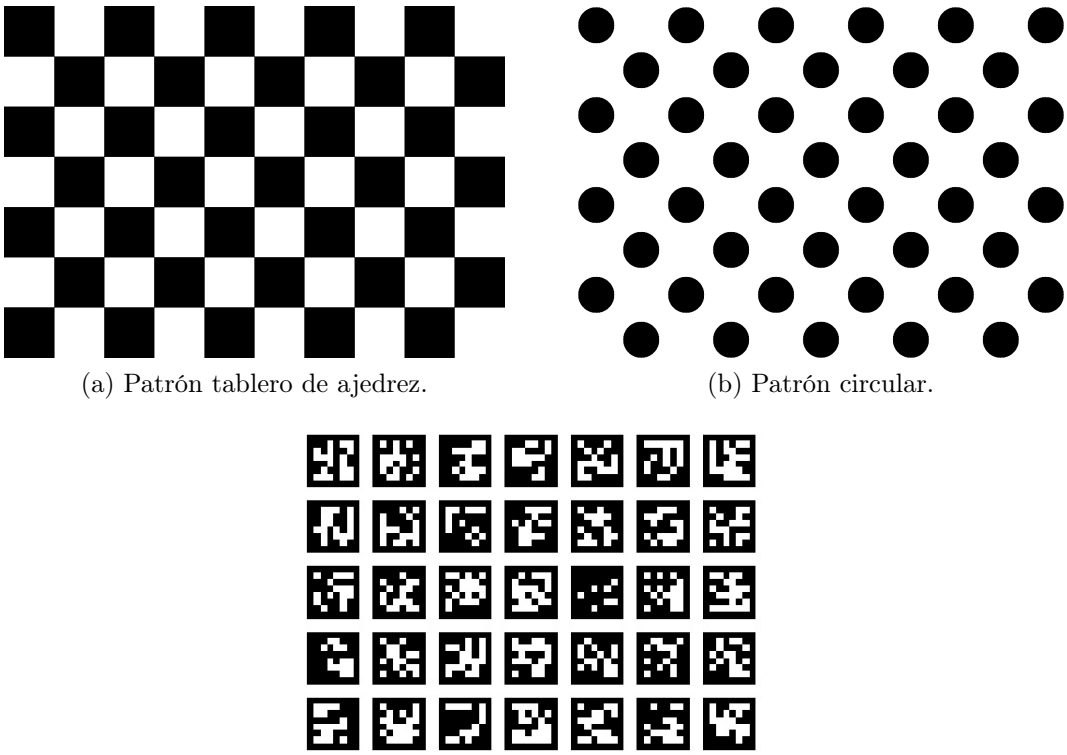
La calibración de una cámara (Isern González, 2003) es el proceso que permite obtener los parámetros que definen las condiciones de formación de una imagen, incluyendo la geometría interna y la óptica de la cámara (parámetros intrínsecos), así como su posición y orientación respecto a un patrón de calibración (parámetros extrínsecos). Dichos parámetros se definen como sigue (de la Escalera et al., 2010):

- Parámetros Intrínsecos: Son los que relacionan el sistema de coordenadas de la cámara con el sistema de coordenadas de la imagen. Estos parámetros se refieren a las distancias focales horizontal, f_u, y vertical, f_v, las coordenadas del centro óptico (c_u,c_v), y las distorsiones de la lente: radial, k, y longitudinal, p.

- Parámetros Extrínsecos: Se refieren a la ubicación física de la cámara definida por la posición y orientación de ésta respecto al sistema global de coordenadas del mundo. Esta posición está definida por una traslación T seguida de una rotación, R.

Existen herramientas y algoritmos utilizados en la calibración de cámaras. OpenCV cuenta con funciones que realizan esta operación de forma automática. Estas funciones están basadas en la utilización de patrones planos con características definidas, tales como esquinas y líneas, que son detectadas y por medio de ellas es posible el cálculo de los parámetros propios de cada cámara. La figura 2.1 muestra algunos de los patrones planos más utilizados en la calibración de cámaras.

El procedimiento para realizar el calibrado consiste en colocar el patrón frente a la cámara y moverlo en distintas direcciones, rotarlo, acercarlo y alejarlo tal como se muestra en la figura 2.2.



(a) Patrón tablero de ajedrez.

(b) Patrón circular.

(c) Patrón ArUco.

Figura 2.1: Distintos patrones planos usados para calibración de cámaras en OpenCV (Kaehler and Bradski, 2016) .

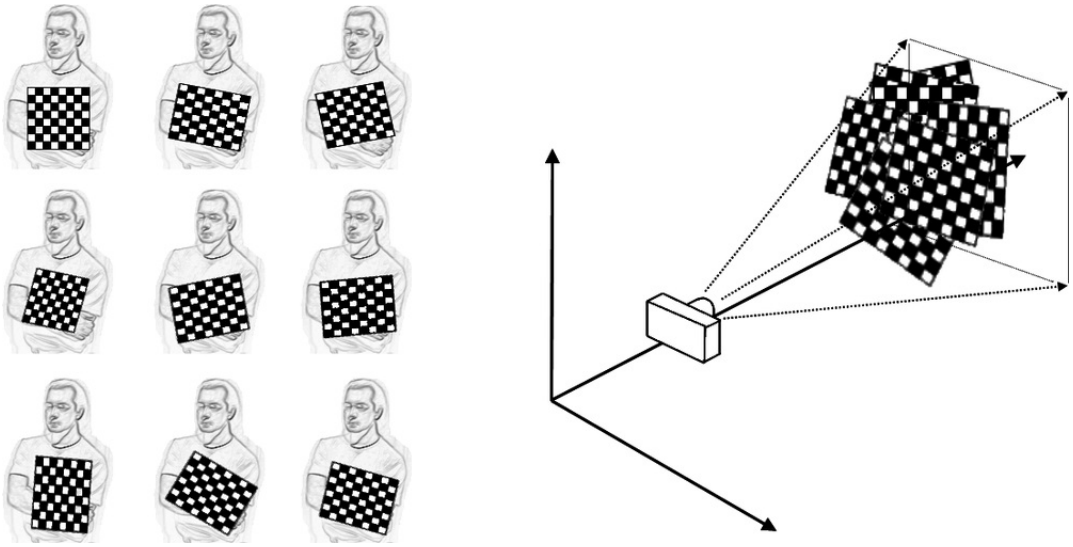


Figura 2.2: Procedimiento de calibración de cámaras en OpenCV (Kaehler and Bradski, 2016).

2.4. Homografía planar

Las imágenes captadas por una cámara están sujetas a una serie de efectos que hacen que la realidad no esté fielmente representada (González, 1999). Estos efectos tienen, principalmente, tres contribuciones:

- Distorsión de la lente: Debido a que los centros de las lentes no están perfectamente alineados y a que existen pequeñas aberraciones en cada una de ellas, la imagen captada distorsiona la realidad. Típicamente esta distorsión se descompone en radial y tangencial, siendo la primera la más importante y la que tienda a curvar las líneas rectas.
- Efecto de la perspectiva: debido a la conicidad de la perspectiva, líneas paralelas en la realidad convergen a un punto de fuga en la imagen.
- Efectos del relieve del terreno: la existencia de ondulaciones en la escena a plasmar, produce que existan unos aparentes desplazamientos en la imagen de forma que los puntos con mayor elevación aparecen más próximos al centro.

La homografía planar es una forma de rectificar las imágenes corrigendo el efecto de la perspectiva. Puede definirse homografía planar como el procedimiento de proyección de un punto de un plano a otro, y puede aplicarse si el terreno no tiene irregularidades y el plano al que se proyecta es paralelo a la superficie (Benítez, 2007). La homografía o proyectividad, como también suele llamarse, puede ser usada como una representación formal del movimiento entre imágenes consecutivas cuando la escena capturada en la cámara se puede suponer que se encuentra en un plano.

2.5. Detector de bordes de Canny

El algoritmo de Canny (Rebaza, 2007) es usado para detectar todos los bordes existentes en una imagen. Este algoritmo, que utiliza máscaras de convolución y se basa en la primera derivada, se considera como uno de los mejores métodos de detección de contornos.

El métodos relacionado con la detección de bordes usando la primera derivada, tiene como fundamento que toma el valor de cero en todas las regiones donde no varía la intensidad y tiene un valor constante en toda la transición de intensidad. Por tanto un cambio de intensidad se manifiesta como un cambio brusco en la primera derivada, característica que es usada para detectar un borde, y en la que se basa el algoritmo de Canny.

En 1986, Canny propuso un método para la detección de bordes, el cual se basa en tres criterios, estos son:

- Un criterio de detección expresa el hecho de evitar la eliminación de bordes importantes y no suministrar falsos bordes.
- El criterio de localización establece que la distancia entre la posición real y la localizada del borde se debe minimizar.

- El criterio de una respuesta que integre las respuestas múltiples correspondientes a un único borde.

El algoritmo de Canny consta de tres pasos:

- Obtención del gradiente: en este paso se calcula la magnitud y orientación del vector gradiente en cada píxel.
- Supresión no máxima: en este paso se logra el adelgazamiento del ancho de los bordes, obtenidos con el gradiente, hasta lograr bordes de un píxel de ancho.
- Histéresis de umbral: en este paso se aplica una función de histéresis basada en dos umbrales; con este proceso se pretende reducir la posibilidad de aparición de contornos falsos.

2.6. Mapas en robótica

Un robot móvil autónomo posee áreas fundamentales, que en conjunto, permiten que el robot navegue en un ambiente. Estas son:

- Percepción: Provee al robot de información de su entorno.
- Localización: Permite al robot ubicarse en el entorno.
- Mapeo y exploración: A partir de la percepción, el robot realiza un modelo de su entorno, el cual sirve de base para que pueda localizarse y navegar
- Planeación.
- Navegación y control.

La unión y dependencia de las áreas anteriores puede verse en la figura 2.3.

Una de las tareas importantes de un robot móvil autónomo es adquirir conocimiento sobre su entorno. Para esta tarea hace uso de sensores. Los sensores pueden ser propioceptivos o extroceptivos, siendo los primeros los que otorgan información del estado interno del robot y los segundos, los que adquieren información el ambiente. A su vez se clasifican en activos y pasivos. Los primeros emiten señales al ambiente para funcionar, mientras que los pasivos miden la energía ya existente. La desventaja de los activos es que al generar energía esta puede afectar al ambiente mismo al emitir energía.

Las cámaras son sensores de tipo pasivo, que funcionan debido a los cambios de la luz visible, además de ser de bajo costo y que proporcionan abundante información del entorno. Estas características hacen que las cámaras sean un sensor muy usado en la construcción de mapas

Figura 2.3: Interrelación de las áreas de un robot móvil autónomo (Pulido Fentanes et al., 2012).

Un modelo o mapa del entorno es una abstracción con la que se representan únicamente aquellas características del entorno que se consideran útiles para la navegación o la localización del robot (Gallardo López, 2000). La utilidad principal de un mapa es proporcionar el elemento fundamental para la localización del robot.

El mapeado robótico tiene importantes desafíos (Thrun, 2002) listados a continuación :

- Ruido en la medición: esta complicación viene del hecho de que el ruido en la medición es estadísticamente dependiente debido a que el error es acumulativo a través del tiempo.
- Dimensionalidad: la descripción de los ambientes tiene una alta dimensionalidad, dependiendo de las entidades presentes en el entorno a ser reconstruido.
- Problema de correspondencia o asociación de datos: este problema corresponde a la determinación de la relación entre distintas mediciones a través del tiempo.
- Entornos cambiantes a través del tiempo: generalmente los entornos donde se mueve un robot no se mantienen estáticos lo que representa un reto para la actualización constante del mapa.
- Navegación durante el mapeo: este último desafío se refiere a la exploración, es decir, el robot debe construir el mapa mientras decide la ruta que seguirá.

2.6.1. Tipos de mapas

Los mapas pueden clasificarse de diferentes maneras, por ejemplo según el ámbito en local o global. La forma más común es en función de su nivel de representación: topológico, métrico y semántico (Aguilar, 2014). Los mapas topológicos, modelan de forma cualitativa,

Mapas métricos	**Mapas topológicos**
Ventajas	
+Fáciles de representar, construir y mantener. +El reconocimiento de lugares (basado en la geometría) no es ambiguo y es independiente del punto de vista. + Facilita el cálculo de rutas más cortas.	+Permite una planificación eficiente, baja complejidad del espacio. +No requiere la conocer la posición exacta del robot.
Desventajas	
-Planificación ineficiente y que consume mucho espacio. -Requiere conocer la posición exacta del robot. -Interfaz deficiente para la mayoría de los solucionadores de problemas simbólicos.	-Difíciles de construir y mantener en entornos a gran escala -Dependen del punto de vista. - Puede producir rutas subóptimas

Tabla 2.1: Ventajas y desventajas de los mapas métricos y topológicos (Thrun, 1998).

considerando "lugares" y las conexiones entre ellos. Estos mapas tienen forma de grafos, donde los nodos representan distintas posiciones alcanzables por el robot y las aristas, la posibilidad de acceder a uno desde el otro.

Los mapas semánticos descartan toda información geométrica, por lo que su representación del entorno esta basado en etiquetas con atributos lingüísticos. Este nivel es similar a a la comunicación humana.

Las representaciones métricas consideran el espacio bidimensional en el que se producirá el desplazamiento y ubica en este los objetos. El tipo de mapa métrico se subdivide en mapas de rejilla y mapas basados en objetos. En la tabla 2.1 se listan las ventajas y desventajas que diferencian a los mapas métricos y topológicos.

Mapas basados en rejillas

Los mapas basados en rejillas dividen el espacio en celdas y consisten en determinar la ocupación o no de dichas celdas. El algoritmo de este tipo de mapas fue desarrollado a mediados de los años ochenta por Elfes y Moravec (Thrun et al., 2002). Estos mapas tienen principalmente tres ventajas:

- Son robustos, fáciles de implementar y de actualizar.
- El reconocimiento de lugares se basa en geometría y no es ambigua.
- Facilita el cómputo de trayectorias más cortas.

Los mapas basados en rejilla tienen la desventaja de requerir mayor capacidad de almacenamiento y cómputo cuanto mayor sea la complejidad del espacio.

Mapas basados en objetos

Los mapas basados en objetos están compuestos por formas geométricas básicas u objetos, tales como líneas, paredes, entre otros. En la robótica móvil, la idea de representar mapas por formas geométricas simples se remonta a un documento de Chatila y Laumond, que propuso representar mapas en 2D utilizando líneas en lugar de cuadrículas. Estos mapas tiene cuatro principales ventajas:

- Son más compactos que los basados en rejillas.
- Pueden ser más precisos, siempre que haya correspondencia entre objetos reales y del enfoque.
- Son más útiles para describir entornos dinámicos donde los objetos pueden cambiar su ubicación a lo largo del tiempo.
- Su percepción es más parecida a la de las personas, lo que puede facilitar la interacción entre humanos y robots.

Su principal desventaja es que solo pueden representar ambientes con objetos o formas simples.

2.7. Interpolación polinomial

El problema matemático de la interpolación es el siguiente:

Dado un conjunto de n pares de valores (xk, yk), encontrar una función$f(x)$ que cumpla $f(xk) = yk$, $k = 1, \cdots, n$.

Existen diversos métodos para encontrar dicha función. Los más conocidos son los métodos que interpolan $f(x)$ por medio de un polinomio o una función racional (cociente de dos polinomios).

En el caso de la interpolación polinómica dados n puntos, el grado más bajo del polinomio que pasa por los n puntos es $n-1$, a menos que dos o más puntos pertenezcan a un polinomio de grado más bajo. Un método de interpolación polinómica es del diferencias divididas de Newton.

El método de diferencias divididas de Newton se basa en calcular una tabla de diferencias divididas una vez, y estás son utilizadas para cada dato que se vaya a interpolar (Michavila and Gavete, 1992). El polinomio se define en la ecuación 2.8.

$$P_n(x) = a_0 + (x - x_0)\, a_1 + (x - x_0)(x - x_1)\, a_2 + \cdots + (x - x_0)(x - x_1)\cdots(x - x_{n-1})\, a_n \quad (2.8)$$

El cálculo de coeficientes se realiza por medio del conjunto de ecuaciones 2.9.

$$\begin{aligned} a_1 &\equiv f[x_0, x_1] = \tfrac{f(x_1) - f(x_0)}{x_1 - x_0} \\ a_2 &\equiv f[x_0, x_1, x_2] = \tfrac{f[x_1, x_2] - f[x_0, x_1]}{(x_2 - x_0)} \\ an &\equiv f[x_0, x_1, \cdots, x_n] = \tfrac{f[x_1, \cdots, x_n] - f[x_0, \cdots, x_{n-1}]}{(x_n - x_0)} \end{aligned} \quad (2.9)$$

2.8. AutoNOMOS Mini V2

"AutoNOMOS Mini v2" (AutoNOMOS, 2017) es un modelo de vehículo (escala 1:10) desarrollado para fines educativos por la Freie Universität Berlin. Puede ser controlado usando un teléfono móvil o puede ser programado para conducción en modo completamente autónomo. La computadora principal es una placa Odroid (XU4 64GB) que ejecuta Ubuntu Linux y el Sistema Operativo Robótico (ROS). El vehículo, mostrado en la figura 2.4 , se mueve con un motor con escobillas RC540, y la dirección se realiza a través de un servomotor HS-645MG. La plataforma cuenta con distintos sensores:

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

Figura 2.4: AutoNOMOS mini V2

2.8.1. Intel Real Sense 300

El sensor Intel Real sense 300, mostrado en la figura 2.5, es una cámara RGB-D. Este tipo de cámaras nos permiten adquirir una imagen plana en color y una imagen de profundidad,

es decir, nos permiten saber a qué distancia (o profundidad) de la cámara se encuentran cada uno de los píxeles de la imagen.

Figura 2.5: Intel Real Sense 300

La cámara Real sense tiene las siguientes características:

- Captura de profundidad de 0.2 a 1.5 m
- Sistema de proyector de láser infrarrojo (IR)
- Profundidad sincronizada, color, video infrarrojo
- Mapeo de textura de profundidad, a color
- Desproyección de profundidad a coordenadas globales
- Hasta 60FPS de profundidad a 640x480 (VGA)
- Color de 30FPS a 1920x1080 (FHD)
- Infrarrojos de 200FPS a 640x480 (VGA)

2.8.2. Unidad de medición inercial

La Unidad de Medición inercial (IMU, por sus siglas en inglés) es un conjunto de sensores conformado por acelerómetros, giroscopios, magnetómetro, termómetro y barómetro. Su principal función es brindar información que ayude a determinar la posición de un agente.

La plataforma AutoNOMOS mini V2 incluye al MPU6050 que contiene un giroscopio de tres ejes con el que se puede medir velocidad angular y un acelerómetro también de tres ejes con se mide la aceleración en los componentes X, Y y Z. La adquisición y procesamiento de las mediciones del MPU6050 se realizan en un Arduino Nano, presente también en la plataforma.

2.9. Odometría

La palabra odometría se compone de las palabras griegas odos (que significa "camino") y metron (que significa "medida"). Se denomina odometría al uso de los datos de los sensores para estimar los cambios en la posición de un objeto en el tiempo.

En robótica la odometría es un método muy utilizado para la localización de robots móviles de ruedas. La odometría (Sánchez, 2013) consiste en colocar sensores propioceotivos, generalmente encoders ópticos, para obtener una aproximación de la posición real en un determinado instante respecto a un sistema de referencia inicial. La odometría es un método de fácil implementación en tiempo real, además de que no es costoso si utiliza encoders. La desventaja de este método consiste en una acumulación de errores cuando el recorrido es muy largo, mismos que también causan incertidumbre en la orientación del móvil.

Otro método de odometría es la de tipo visual. Esta se encarga de obtener la posición de un robot móvil por medio del procesamiento de imágenes. Este tipo de odometría esta basado en que el ambiente es estático y que es posible la detección y asociación de características entre imágenes sucesivas.

2.10. OpenCV

OpenCV (Kaehler and Bradski, 2016) es una biblioteca de visión artificial de código abierto. Fue lanzada en 1999 por Gary Bradski con la esperanza de acelerar la visión artificial y la inteligencia artificial proporcionando una infraestructura sólida para todos los que trabajan en el campo. La biblioteca está escrita en C y C++ y se ejecuta en Linux, Windows y Mac OS X. Así mismo esta diseñada para funcionar en distintas interfaces: Python, Java, MATLAB y otros lenguajes, incluida la portabilidad de la biblioteca a Android e iOS para aplicaciones móviles.

OpenCV fue diseñada en función de eficiencia computacional y con un enfoque fuerte en aplicaciones en tiempo real. Uno de sus objetivos es facilitar una herramienta simple de usar para el rápido desarrollo de sofisticadas aplicaciones de visión. Esta librería contiene cerca de 500 funciones aplicables en muchas áreas de visión.

2.11. Robot Operating System (ROS)

ROS (Robot Operating System) provee librerías y herramientas para ayudar a los desarrolladores de software a crear aplicaciones para robots. Fue desarrollado en Willow Garage, Laboratorio de Inteligencia Artificial de Stanford. ROS (ROS, 2018) provee abstracción de hardware, controladores de dispositivos, librerías, herramientas de visualización, comunicación por mensajes, administración de paquetes y más.

No es un sistema operativo en el sentido tradicional de administración y planificación de procesos, sino que provee una capa de comunicaciones estructuradas por encima del sistema

operativo anfitrión. Provee la funcionalidad básica de cualquier sistema operativo tradicional como: abstracción del hardware, acceso a dispositivos de bajo nivel, implementación de funcionalidad de uso común, comunicación entre procesos, mantenimiento de paquetes, etc., además proporciona librerías, y todo tipo de herramientas (visualizadores, GUI's, etc.) para el desarrollo de robots y sus aplicaciones.

ROS organiza el software según una arquitectura de grafos en la que en el más bajo nivel se tienen los nodos (porción de código ejecutable) y éstos se comunican entre sí a través de tópicos (interfaz de ROS para el paso de mensajes entre nodos). A su vez los nodos se agrupan en paquetes (colecciones mínimas de código reusable), y éstos en stacks que permiten compartir el código fácilmente. La ventaja de ROS es que es de código abierto y tiene una comunidad creciente que lo apoya y da mantenimiento, así como gran actividad respecto de módulos disponibles para manejadores de dispositivos y algoritmos de robótica móvil.

A nivel de software los paquetes se organizan con los elementos listados a continuación:

- **CMakeFile:** Es el archivo que se encarga de compilar el paquete. En él están los códigos que el paquete utiliza, librerías y las dependencias del mismo.
- **Package.xml:** Contiene las dependencias del paquete, y se define como serán construidos y ejecutados.
- **Src:** es una carpeta que contiene el código que se convierte en ejecutables o nodos.
- **Launch:** contiene los lanzadores .launch que permiten iniciar y configurar varios nodos a la vez.

ROS provee herramientas para visualización de información en tiempo real como es el caso de Rviz. Por medio de ella pueden visualizarse imágenes, mediciones de odometría, nubes de puntos, entre otras salidas de sensores.

Una de las característica importantes de este sistema operativo para robots es que permite la simulación de los robots tomando en cuenta los parámetros físicos del robot y sus restricciones. Para esta función se utiliza la herramienta Gazebo.

2.12. Simulador AutoNOMOS mini en Gazebo

Dentro de las prestaciones importantes de ROS está la simulación de robots y entornos de forma gráfica utilizando Gazebo. Gran cantidad de estos modelos generados se ponen a disposición del público en repositorios en línea. Para el desarrollo de este trabajo se utilizó el simulador diseñado por ITAM (2019), el cual esta disponible en línea. El robot que incluye este modelo cuenta con una cámara y un sensor Lidar, así como con movimiento y medición de orientación y odometría.

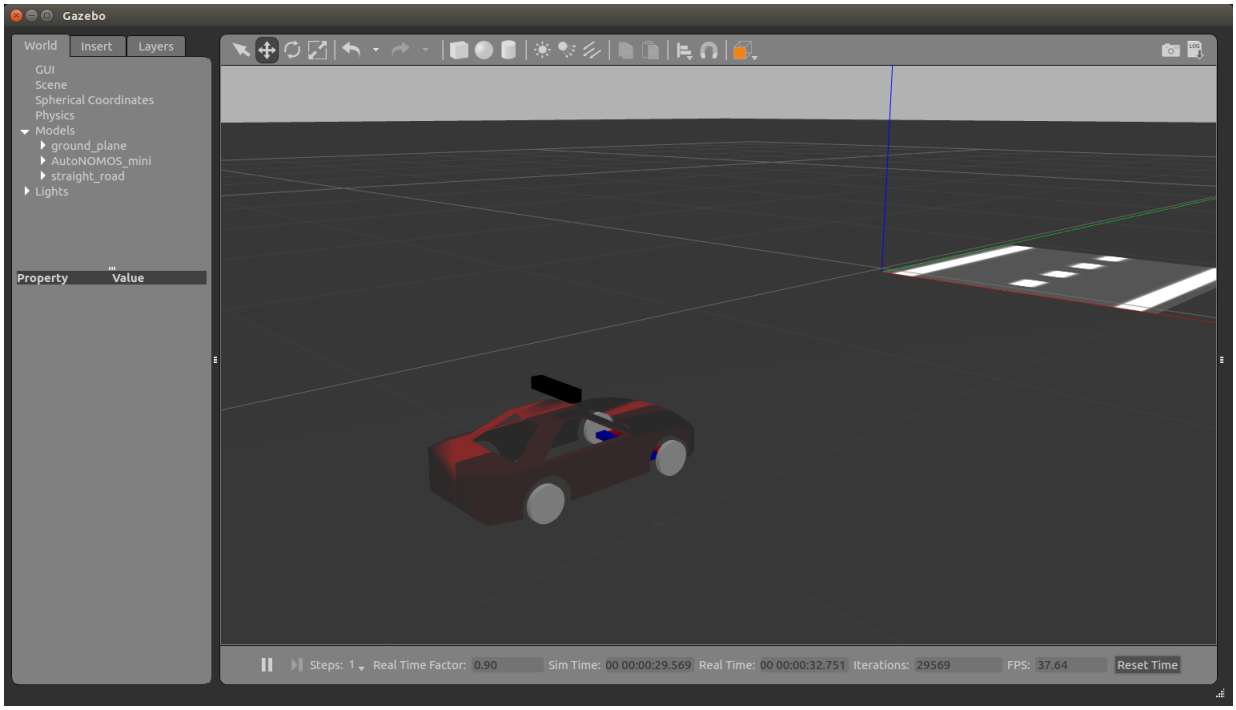


Figura 2.6: Simulador AutoNOMOS mini (ITAM, 2019)

Capítulo 3

Desarrollo

En este capítulo se describe la metodología seguida para realizar la reconstrucción del mapa. En la primera sección se incluye una descripción breve del funcionamiento de la plataforma AutoNOMOS mini, así como su organización y nodos principales en ROS.

La figura 3.1 muestra de forma extendida la metodología propuesta para la reconstrucción del mapa. Las etapas contenidas en dicho diagrama son desarrolladas en este capítulo a partir de la Sección 3.2. Las etapas 3 y 4 del diagrama 3.1 son descritas en la Sección 3.4 dentro de cada método de detección de carriles.

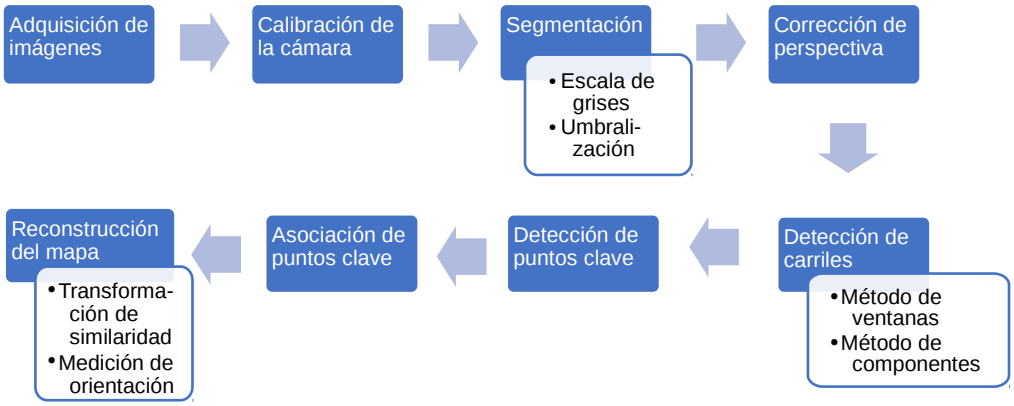


Figura 3.1: Metodología propuesta.

3.1. AutoNOMOS mini y su funcionamiento en ROS

El vehículo AutoNOMOS mini es un robot cuyo funcionamiento esta integrado en ROS. Se cuentan con distintos paquetes diseñados para sensado y movimiento de la plataforma. La figura 3.2 muestra esta organización. Los nodos controlan cámaras, sensor Lidar, odometría, orientación, motores y leds de iluminación.

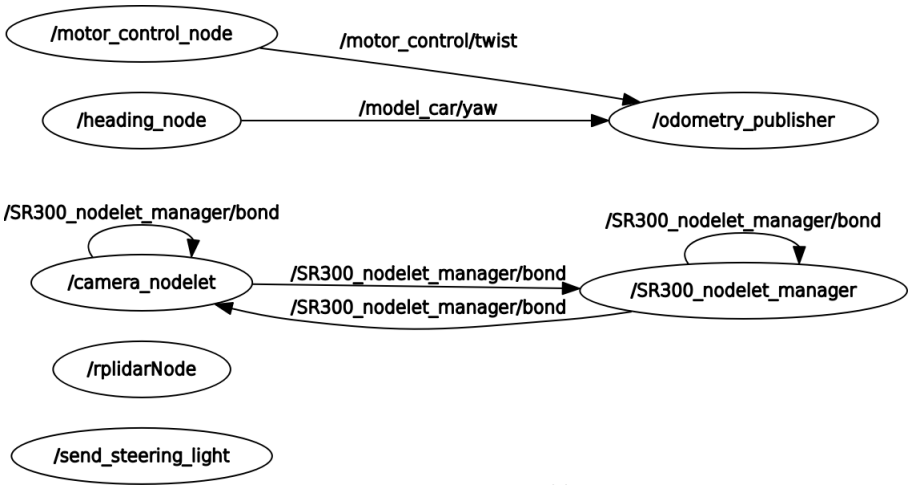


Figura 3.2: Organización en nodos de la plataforma AutoNOMOS mini.

Los nodos se comunican entre si utilizando tópicos. Su organización conjunta se muestra en la figura 3.3. Al suscribirse a los tópicos cualquier nodo puede adquirir información o enviar indicaciones para hacer alguna acción como el aumento de velocidad o cambio de dirección de giro de los motores.

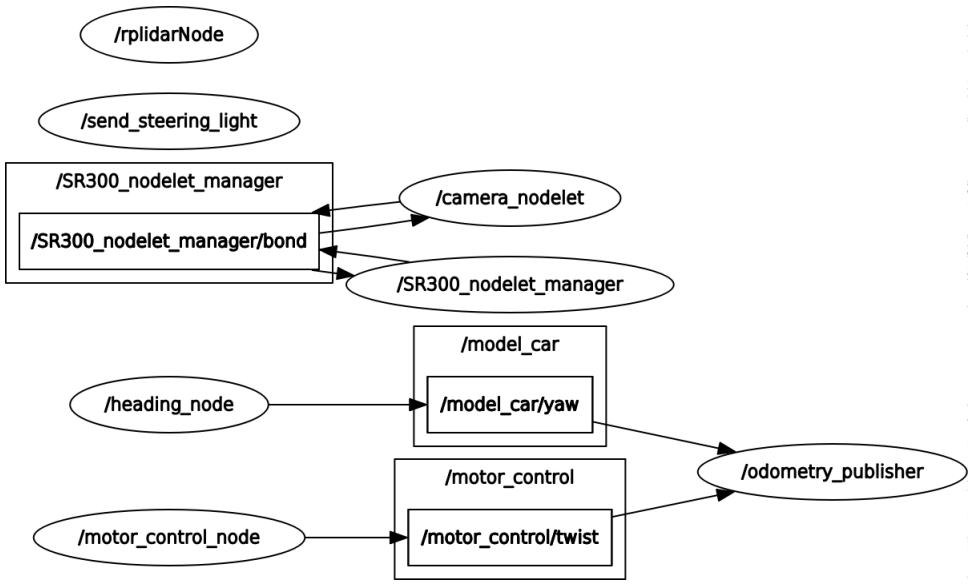


Figura 3.3: Organización de tópicos de la plataforma AutoNOMOS mini.

3.2. Adquisición de imágenes

En la plataforma se cuenta con 2 cámaras (Real Sense y ojo de pescado), así como un sensor láser Lidar. Esto permite que el vehículo pueda conocer su entorno por medio de

imágenes y nubes de puntos. La cámara Real Sense proporciona información, por el lugar donde esta montada, de lo que se encuentra frente al vehículo. De esta forma, en un recorrido de un circuito de competencia, se convierte en la principal fuente de información para el seguimiento de carriles. Las imágenes capturadas por este sensor son como la mostrada en la figura 3.4.

Figura 3.4: Imagen inicial.

3.3. Calibración de la cámara Intel Real Sense

Uno de las etapas iniciales importantes en el desarrollo de este trabajo es la calibración de la cámara Intel Real Sense. Los parámetros intrínsecos que se obtienen en dicho proceso, descritos en la sección 2.3 del capítulo 2, son utilizados posteriormente en la corrección de perspectiva. En ROS existe un paquete propio para la calibración de cámaras, tanto monoculares como estéreo, que utiliza las funciones de *OpenCV*. Dicho paquete se llama *camera_ calibration* e incluye distintos nodos para funciones distintas. Para comenzar la calibración de una cámara de tipo monocular la instrucción en consola es la siguiente:

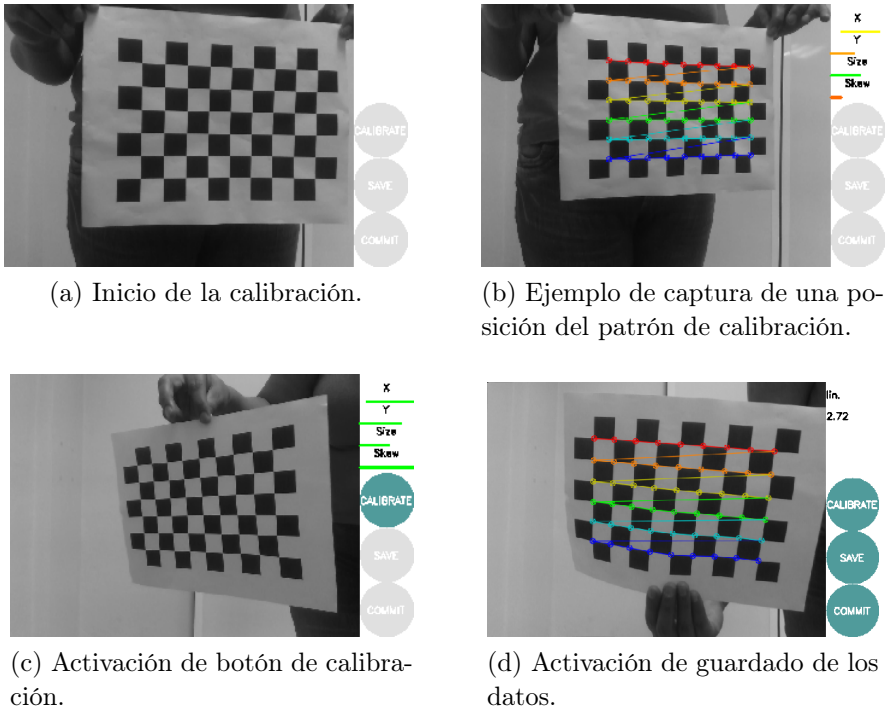
rosrun camera_ calibration cameracalibrator.py –size 10x7 –square 0.254 image:=/app/camera/ir/image_ raw

Como se observa, los parámetros se refieren al nodo ejecutable, el número de cuadros del patrón de calibración, el tipo de patrón, el tamaño de los cuadros y el nombre del tópico en el cual se pública la imagen de la cámara a calibrar. El patrón usado en este procedimiento fue de tipo tablero de ajedrez de 10×7, de 1 pulgada cada uno, tal como se muestra en las capturas de la figura 3.5.

Dado que la cámara a utilizar es la Intel Real Sense, se realizó su calibración tal como se muestra en la figura 3.5. El procedimiento de calibración es relativamente simple y consiste

en lo siguiente:

- Se lanza el nodo con la instrucción descrita anteriormente.
- Se coloca el patrón frente a la cámara y se realizan distintos movimientos. Estos consisten principalmente en acercar, alejar, girar, subir y bajar el patrón de calibración dentro del campo de visión de la cámara.
- La aplicación toma captura de todas las distintas posiciones del patrón obtenidas y una vez que considera que tiene las suficientes se activa el botón de calibración, tal como se muestra en la figura 3.5 c).
- Una vez calculados los parámetros se activa la opción de respaldar los datos, esto se ve en la figura 3.5 d).



(a) Inicio de la calibración.

(b) Ejemplo de captura de una posición del patrón de calibración.

(c) Activación de botón de calibración.

(d) Activación de guardado de los datos.

Figura 3.5: Proceso de calibración.

La salida de esta aplicación corresponde a la muestra de los datos en consola y en un archivo de texto. Estos datos se refieren a los parámetros intrínsecos. La forma de presentación de estos es la mostrada en la figura 3.6.

3.4. Detección de carriles

Se presentan dos algoritmos diferentes para la detección de carriles, el primero basado en ventanas y el segundo en la detección de componentes o contornos. Si bien las reglas de

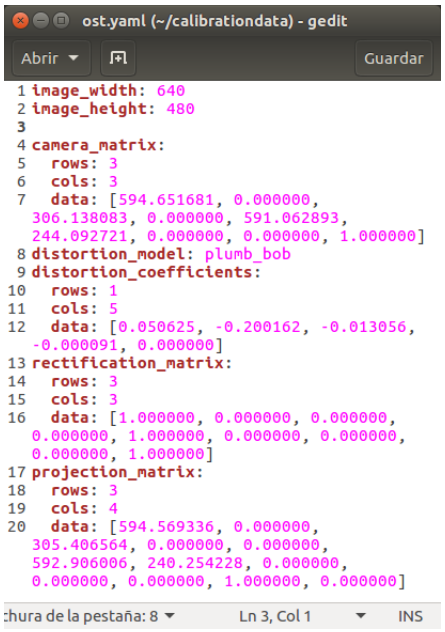


Figura 3.6: Archivo de texto con parámetros obtenidos de la calibración.

las competencias de coches autónomos a escala, como la Copa Carolo y Talent Land, indican que la pista debe tener la línea central de forma discontinua también se encuentran pistas que no cumplen esta característica, como es el caso de Torneo Mexicano de Robótica. Esta es la razón de tener dos métodos diferentes, uno basado en el recorrido por ventanas y el otro en las componentes de la línea central.

3.4.1. Detección de carriles usando ventanas

Este primer método para detección de carriles está basado en la búsqueda de concentración de píxeles blancos en el eje x de la imagen y en un recorrido por ventanas tomando la idea de (Ballew, 2018). En este contexto se define una ventana como una porción de la imagen, delimitada por un rectángulo y un punto de inicio.

Para este trabajo se modificó el algoritmo de forma que se adaptara a las imágenes tomadas por la cámara Intel Real Sense. Además esta técnica fue implementada con C++ construyendo cada una de las funciones para detección y conteo de píxeles blancos, cálculo de corrección de perspectiva, generación de ventanas, entre otras. De forma general se muestra en el diagrama de la figura 3.7 la simplificación por etapas del desarrollo de este procedimiento.

El primer paso es corregir la distorsión radial de la imagen. Dicho procedimiento utiliza los coeficientes $k = [0.158445 - 0.345884 0.017252 - 0.001923;]$, obtenidos durante la calibración de la cámara. En la figura 3.8 puede verse el resultado de esta corrección, cuya modificación es mínima debido a las magnitudes de los coeficientes de distorsión.

La siguiente etapa es el recorte de la sección de interés, con la cual se reduce el área de búsqueda a solo la mitad inferior que, como se observa en la figura 3.9, es la porción de la

Figura 3.7: Etapas de la detección de carriles utilizando ventanas.

(a) Original. (b) Corrección de distorsión.

Figura 3.8: Corrección de distorsión.

imagen que contiene la información más relevante en cuanto a las líneas de los carriles. El nuevo tamaño de la imagen es de 640×240.

Figura 3.9: Recorte inferior: sección de interés.

Con esta imagen más pequeña lo siguiente a realizar es la corrección de perspectiva. Esta corrección es llevada a cabo utilizando una aplicación desarrollada en C++ y OpenCV. En dicha

aplicación se estiman los parámetros de rotación, traslación y escalamiento ajustándolo de forma interactiva. El cálculo de la matriz de corrección de perspectiva H se realiza utilizando la ecuación 3.1, donde A_1 y A_2 son matrices definidas por el tamaño de la imagen y las distancias focales(f_u, f_v), R_x, R_y y R_z corresponden a las matrices de rotación, T es la matriz de traslación y S la matriz de escalamiento.

$$H = A_1 * (T * (S * (R_x R_y R_z) * A_2)) \tag{3.1}$$

$$A_1 = \begin{bmatrix} f_u & 0 & 320 & 0 \\ 0 & f_v & 240 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.2}$$

$$A_2 = \begin{bmatrix} 1 & 0 & -320 \\ 0 & 1 & -240 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

La figura 3.10a) muestra el ajuste de los valores usando barras de deslizamiento, mientras que en consola (figura 3.10b)) se puede ver el cálculo de la matriz H correspondiente. La figura 3.10c) es el resultado de la corrección de perspectiva de la figura 3.9. La imagen corregida es del mismo tamaño que la imagen original, es decir 640×240. Los pixeles de la imagen corregida que no contienen información después de la corrección se rellenan con ceros.

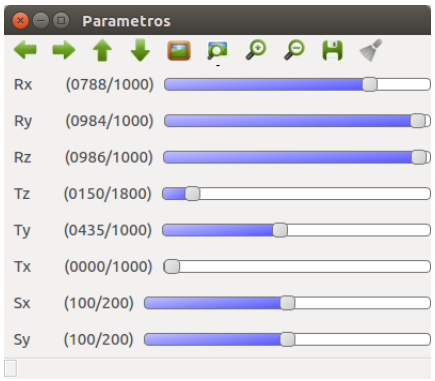
Una vez obtenida la imagen con información relevante para la detección de carriles, el siguiente paso es resaltar los píxeles de interés por medio de una operación de umbralización. El resultado puede verse en la figura 3.11(a). Con este paso se culmina la etapa de pre-procesamiento de la imagen. En el algoritmo 1 se presenta la secuencia de pasos correspondiente.

Algoritmo 1: Pre-procesamiento

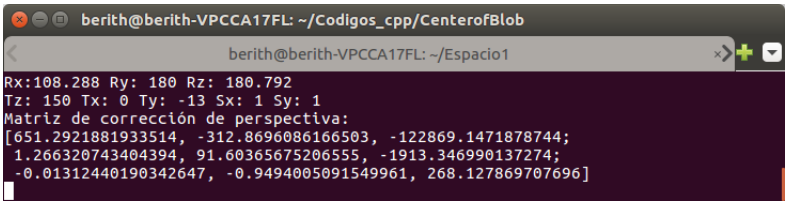
Entrada: imagen original (640x480)
Salida : imagen binarizada(200x200), histograma de pixeles blancos(1x200)

1 Corrección de distorsión.
2 Recorte de imagen(640*240)
3 Homografia(imagen original).
4 Recorte sección de interés(200x200)
5 Conversión a escala de grises.
6 Umbralizado(umbral=180)

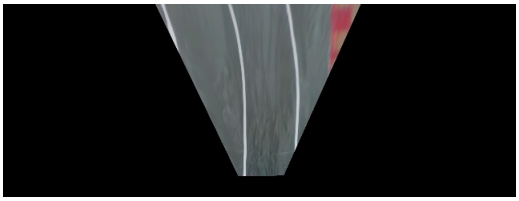
Identificados los píxeles correspondientes a líneas presentes en la imagen, se continua con el recorrido por ventanas, cuyo descripción se encuentra en el algoritmo 2. En este, la primera etapa es encontrar los puntos de inicio de cada carril. Esto se realiza por medio de la búsqueda



(a) Ajuste de parámetros.



(b) Parámetros y matriz de corrección estimados.



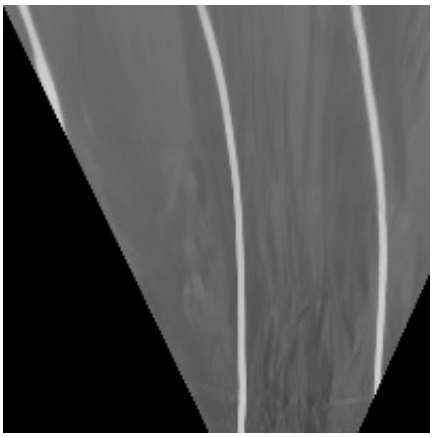
(c) Corrección de perspectiva.

Figura 3.10: Homografía.

de concentración de pixeles blancos de forma horizontal utilizando el histograma. El resultado de este paso se muestra en la figura 3.11(c). Estos puntos corresponden a las coordenadas donde empieza la primera ventana de búsqueda. Cada ventana tiene de un tamaño de 30×10 píxeles, el cual fue elegido después de realizar recorridos completos y ver que con este tamaño no se pierde información aun en las curvas.

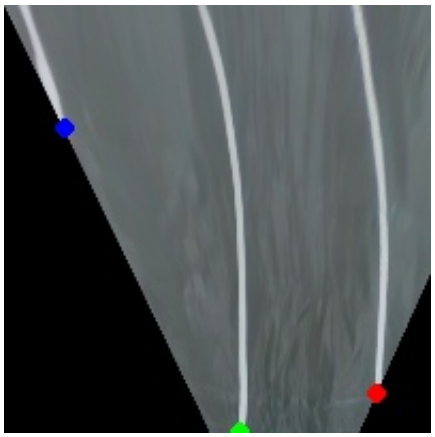
Los pixeles encontrados dentro de cada ventana se concatenan en un solo vector. De esta forma se tienen máximo 3 vectores en caso de que las tres líneas estén presentes. La ubicación de cada ventana de búsqueda puede verse en la figura 3.12. Si bien la primer ventana de inicio se toma a partir del punto inicial de cada línea, cada ventana siguiente se define con base en la ubicación del píxel con la menor coordenada en y de la ventana anterior.

Finalmente con cada conjunto de puntos se realiza una aproximación a un polinomio de segundo orden utilizando el método de mínimos cuadrados. En la figura 3.13 se observan los puntos reales en color negro y los aproximados por los polinomios de cada línea en colores azul, verde y rojo.



(a) Conversión a escala de grises.

(b) Umbralización con valor de gris 180.



(c) Puntos de inicio de cada línea.

Figura 3.11: Búsqueda de puntos de inicio de cada línea.

Algoritmo 2: Ventanas

Entrada: Histograma, imagen binaria

Salida : Vector puntos de líneas existentes, Coeficientes de polinomios

1 Conteo de pixeles blancos en el eje horizontal (Histograma).
2 Cálculo de máximos del histograma.
3 Búsqueda de coordenadas y iniciales.
4 Recorrido de ventanas(imagen binaria, punto inicial)
5 Agrupación de puntos por líneas.
6 Cálculo de coeficientes de polinomio de 2do orden.

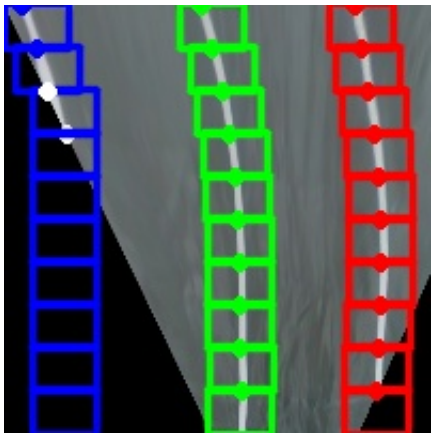


Figura 3.12: Ventanas de búsqueda de puntos de cada línea.

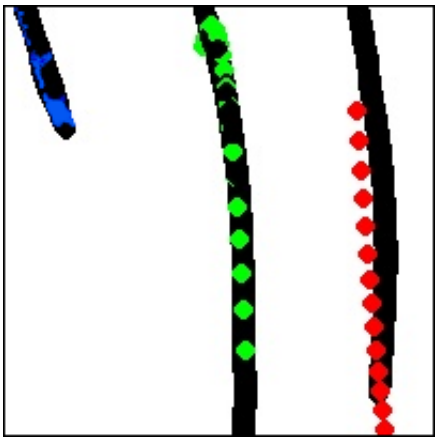


Figura 3.13: Aproximación polinomial de los puntos.

3.4.2. Identificación de líneas de carril utilizando detección de componentes

Este método diseñado para la detección de líneas de carril consiste en detectar las componentes presentes en la imagen y a partir de las áreas y perímetros determinar a que línea pertenecen. El diagrama 3.14 muestra los pasos seguidos en esta metodología.

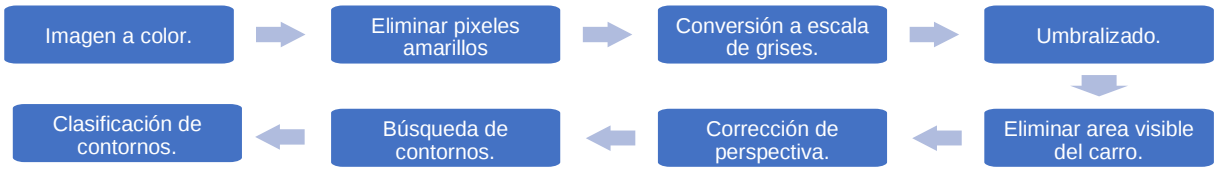


Figura 3.14: Etapas para el procesamiento de las imágenes y detección de los carriles usando detección de componentes.

Figura 3.15: Imagen de inicio.

La imagen de partida es la obtenida directamente de la cámara, esta se muestra en la figura 3.15.

Procesamiento y segmentación

La imagen inicial tiene algunos elementos de tonalidades de amarillo que afectan al momento de realizar el umbralado por lo que se eliminan utilizando una mascara. Dicha mascara es obtenida pasando la imagen al esquema de color CIE L*A*B*. Este esquema de color fue elegido por ser el que eliminaba la mayor parte de los elementos que causaban ruido en las siguientes etapas. El resultado de la multiplicación de la imagen original con la mascara obtenida da como resultado la imagen 3.16.

Figura 3.16: Eliminación de tonalidades de amarillo.

Para facilitar la manipulación de la imagen, se pasa a escala de grises, tal como se muestra en la figura 3.17.

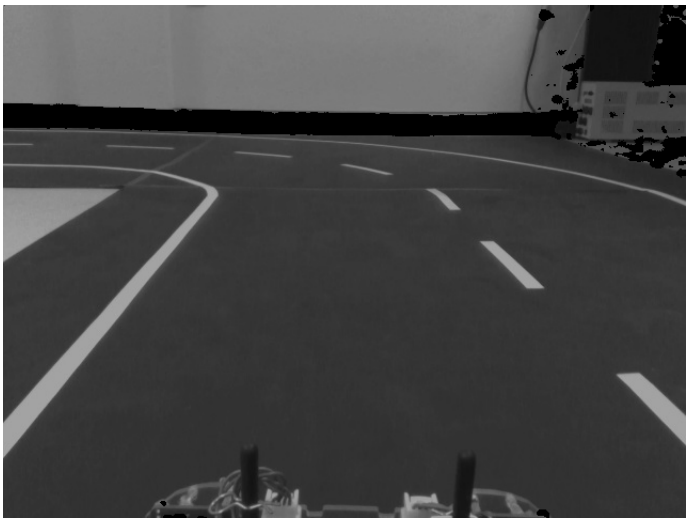


Figura 3.17: Escala de grises.

Figura 3.18: Umbralización.

Dado que es de interés eliminar lo más posible el fondo de la imagen para conservar solo los píxeles que corresponden al nivel de gris de las líneas de los carriles, se realiza la umbralización de la imagen. En la figura 3.18 se muestra el resultado de esta operación. Posteriormente se elimina el área que es visible del chasis del carro, ayudándose de una matriz de tamaño 385×50. El resultado de esta operación se observa en la imagen 3.19.

Figura 3.19: Eliminación de área visible del chasis del carro.

Corrección de perspectiva.

La posición de la cámara respecto al plano provoca que halla una distorsión de las líneas y curvas de forma tal que se pierde la distancia entre las líneas paralelas. La corrección de este problema de perspectiva se encuentra mediante la transformación de la imagen usando una matriz homogénea. Como se describe en la ecuación 3.4, H corresponde a la matriz de transformación de puntos de un plano a otro.

$$H = A_1(TRA_2) \tag{3.4}$$

Donde:

- A_1 y A_2 son definidas con respecto a los parámetros intrínsecos obtenidos de la calibración de la cámara. Las matrices que representan a A_1 y A_2 están indicadas por las ecuaciones 3.5 y 3.6 respectivamente, donde f_u y f_v son las distancias focales y c_u y c_v corresponden a las coordenadas del centro óptico.

$$A_1 = \begin{bmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & c_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.5}$$

$$A_2 = \begin{bmatrix} 1 & 0 & -c_u \\ 0 & 1 & -c_v \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

- R corresponde a la matriz de rotación(XYZ) definida por la ubicación de la cámara respecto a un eje de referencia global. En la plataforma la rotación corresponde solo a $-90°$ sobre el eje X.

- T a la matriz de traslación. Esta matriz esta definida solo por una traslación en el eje Z del plano, correspondiente a la distancia en milímetros del plano al eje focal de la cámara.

Las consideraciones anteriores en cuanto a rotación y desplazamiento en la corrección de perspectiva se toman de esa forma debido a que la cámara esta fija al vehículo. Esto implica que no se desplaza sobre la vertical y que su rotación está restringida al giro del carro. Si bien lo ideal es que la cámara este ubicada a $-90°$ del plano, en la aplicación real es necesario ajustar este ángulo. Lo mismo sucede con la traslación. Para esto se utiliza una aplicación en la cual se ajustan estos dos parámetros, la rotación en x y la traslación en y. La figura 3.20 muestra el resultado real de la corrección de perspectiva.

Figura 3.20: Corrección de perspectiva.

Detección de contornos y clasificación de componentes.

El siguiente paso después de la corrección de perspectiva es la aplicación de un detector de contornos, la figura 3.21 indica en color azul los contornos detectados en la imagen. La imagen aún conserva un área triangular grande que corresponde al suelo donde esta montada la pista. Los píxeles correspondientes a esta área pueden llegar a confundirse con los que corresponden a los carriles. Observando un recorrido completo se determinó que el área del suelo es mayor a las correspondientes a las componentes de los carriles y cuando llega a ser menor su ubicación se encuentra en los extremos de la imagen. Esto permite discriminar de forma correcta las líneas de los carriles.

Según se observo durante los recorridos completos de la pista las componentes del centro se ubican en una sección de la imagen, que corresponde al área blanca mostrada en la figura 3.22. Esta es la primera verificación de que la componente delimitada por un contorno puede pertenecer a la línea central.

Posteriormente, ya delimitada la posible ubicación de la línea central, se notó que las porciones visibles de las líneas laterales en algunas partes del recorrido tienen un área parecida

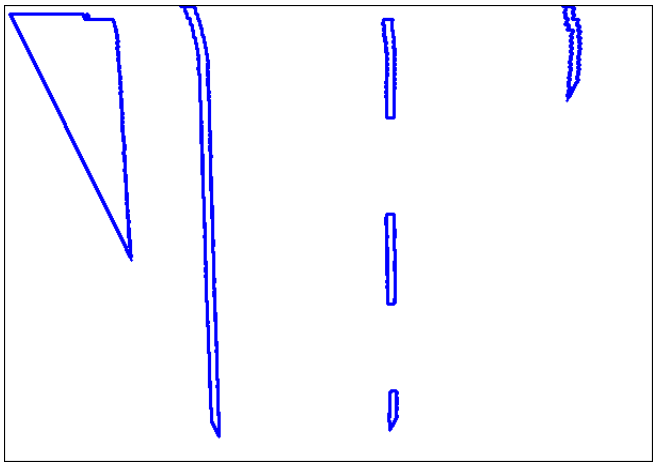


Figura 3.21: Detección de contornos.

a las componentes del centro por lo que tienden a confundirse entre si. Esta confusión no puede resolverse usando la ubicación de sus píxeles ya que en presencia de curvas las líneas se distribuyen por toda la imagen, incluyendo el área blanca de la figura 3.22. Por esta razón se introduce la aproximación polinomial como forma de predecir la ubicación de las componentes centrales a partir de la información de la imagen anterior.

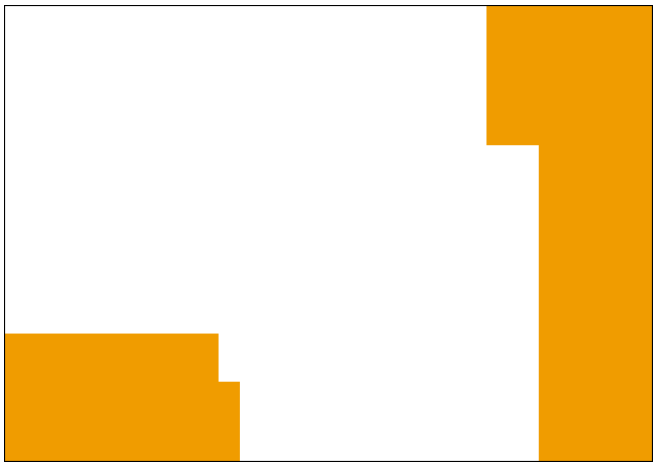


Figura 3.22: Ubicación de contornos centrales.

Se usan los puntos almacenados para la aproximación de un polinomio, el cual puede ser de tres tipos según la información disponible:

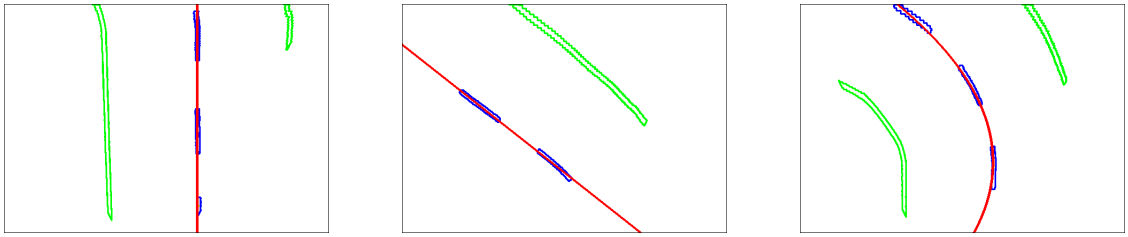
1. Polinomio de grado 0: Cuando el centroide de las componentes de la línea central se ubican muy cerca unos de otros en su coordenada x.

2. Polinomio de grado 1: Se aproxima a una línea recta cuando solo hay presentes dos componentes de la línea central y están distribuidos en el eje x.

3. Polinomio de grado 2: Aproximación a una función de segundo orden por medio del método de mínimos cuadrados cuando hay más de 2 componentes y están alejadas entre si.

En las imágenes a), b) y c) de la figura 3.23 se pueden observar los ajustes de los puntos a los polinomios obtenidos con los datos de la imagen anterior. La aproximación polinomial se realizó sobre el eje x, es decir, se considera como variable dependiente x y como variable independiente y.

La clasificación de los contornos que son componentes centrales esta determinada por la ubicación dentro del área blanca de la figura 3.22 y la cercanía de su centroide, dentro de un umbral, al polinomio de la imagen anterior.



(a) Polinomio de grado 0. (b) Polinomio de grado 1. (c) Polinomio de grado 2.

Figura 3.23: Aproximación polinomial.

De forma general se presenta el método que clasifica cuales contornos detectados pertenecen a la línea central en el algoritmo 3.

En la figura 3.24 se indican las líneas de los carriles, marcando en color verde las líneas laterales y en color azul las componentes de la línea central.

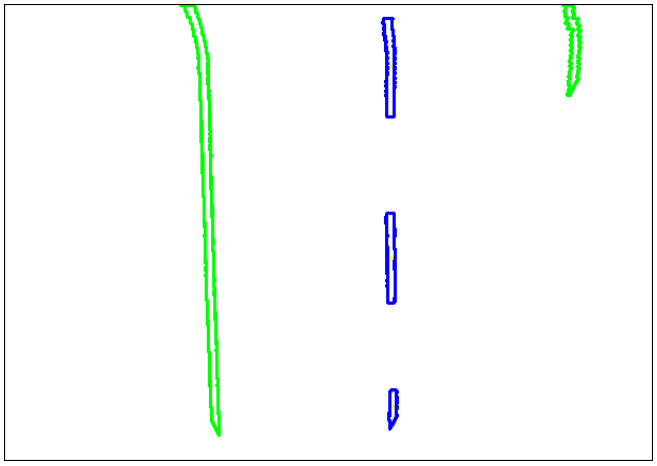


Figura 3.24: Clasificación de contornos.

Algoritmo 3: Clasificación de componentes centrales.

Entrada: Vector de componentes
Salida : Clasificación de componentes

```
1  para i=0;i<tamaño de vector;i++ hacer
2      Cálculo de área por contorno.
3      Cálculo de centroide por contorno.
4      si área(contorno_i)<área máxima y área(contorno_i)>área mínima entonces
5          si centroide(contorno_i) esta en área blanca(figura 3.22) entonces
6              si ¿Es primera imagen? entonces
7                  Guarda centroide en vectorCentro.
8              en otro caso
9                  resul=Evalúa polinomio_anterior(coordenada y del centroide)
10                 si absoluto(resul-coordenada x del centroide)<distancia máxima
                    entonces
11                     Es componente central
12                     Guarda centroide en vectorCentro.
13                 fin
14             fin
15         fin
16     fin
17 fin
18 si Tamaño de VectorCentro!=0 entonces
19     si Los puntos se apoximan a una constante entonces
20         Polinomio de grado 0(constante)
21     en otro caso
22         si Tamaño de VectorCentro==2 entonces
23             Polinomio de grado 1(línea recta)
24         en otro caso
25             Polinomio de grado 2(Curva cuadrática)
26         fin
27     fin
28 fin
```

3.5. Asociación de imágenes

Cuando se realiza la reconstrucción de entornos una etapa clave, sin importar el sensor utilizado, es la asociación de las mediciones. Cuando la información viene de las imágenes, al ser estas sucesivas es importante destacar que puntos pertenecen a la anterior. Para resolver este problema se suelen usar descriptores para encontrar las características importantes de cada imagen.

Las imágenes obtenidas de la Real Sense y que han sido modificadas por las etapas anteriores tienen en común la presencia de componentes centrales, las cuales no cambian en gran medida entre frames sucesivos. Esto hace posible que por medio de estos puedan relacionarse una imagen con su siguiente consecutiva.

3.5.1. Detección de puntos clave (esquinas)

En esta etapa de la reconstrucción del mapa se busca utilizar las características invariantes en las imágenes consecutivas y que permiten encontrar una relación entre las mismas. En este caso se optó por las esquinas las cuales, dado el movimiento, no tienen un cambio brusco entre los frames sucesivos.

De manera general el algoritmo 4 describe el procedimiento de búsqueda de esquinas. Dicho procedimiento consiste en encontrar los coordenadas (x, y) de la imagen que encierran todos los puntos de un contorno en el rectángulo más pequeño posible. Para esto se realiza la búsqueda de las coordenadas x y y mayores y menores de todo el conjunto de contornos.

Posteriormente se buscan los puntos más cercanos a la combinación de las coordenadas x y y mayores y menores para definir las esquinas de la siguiente forma:

- Esquina 0: es el punto más cercano a x_{menor} y y_{menor}.
- Esquina 1: es el punto más cercano a x_{mayor} y y_{menor}.
- Esquina 2: es el punto más cercano a x_{menor} y y_{mayor}.
- Esquina 3: es el punto más cercano a x_{mayor} y y_{mayor}.

La descripción de la búsqueda de esquinas está enlistada en el algoritmo 4. De igual forma la detección se muestra en la figura 3.25.

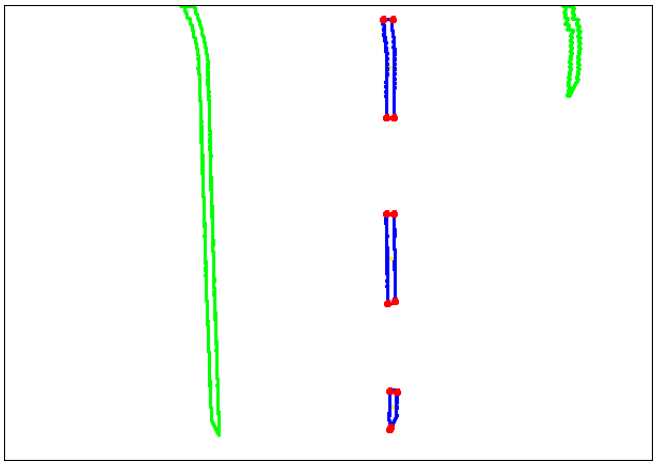


Figura 3.25: Detección de esquinas.

Algoritmo 4: Encontrar esquinas.

Entrada: V=Vector de componentes centrales
Salida : V_2=Vector de esquinas.

```
1  para i=0;i<tamaño V;i++ hacer
2      Encontrar x_mayor, x_menor, y_mayor y y_menor
3      para j=0;j<tamaño de V[i];j++ hacer
4          si (punto_ij más cercano a x_menor, y_menor) entonces
5              esquina_0 = punto_ij
6          fin
7          si (punto_ij más cercano a x_mayor, y_menor) entonces
8              esquina_1 = punto_ij
9          fin
10         si (punto_ij más cercano a x_menor, y_mayor) entonces
11             esquina_2 = punto_ij
12         fin
13         si (punto_ij más cercano a x_mayor, y_mayor) entonces
14             esquina_3 = punto_ij
15         fin
16     fin
17     V_2=Concatena(esquina_0, esquina_1, esquina_2 y esquina_3)
18 fin
```

3.5.2. Asociación de puntos

Una vez detectadas las esquinas de cada contorno central, lo siguiente es relacionarlas con las esquinas de la imagen anterior. Para este paso, se tiene como consideración el hecho de que entre frame y frame el desplazamiento y rotación no aumentan, o disminuyen según el caso, de manera considerable.

La primera comprobación que se realiza es que el componente central este completo. Dado que estos pueden ser rectos o curvos, se optó por usar la distancia entre la Esquina 0 y la Esquina 3. Si esta distancia es aproximadamente igual a la distancia de un componente completo se utilizan las cuatro esquinas. Si este no es el caso solo se guardan las dos superiores o las dos inferiores, según la ubicación de la componente. Se verifica la distancia entre cada una de las componentes del frame anterior para relacionarlo con el nuevo.

El algoritmo 5 enlista las condiciones y etapas para la asociación de esquinas. De manera gráfica se muestra el resultado de está relación entre frames en la figura 3.26. Cada elipse azul encierra dos puntos, los cuales corresponden a la esquina actual y la anterior correspondiente.

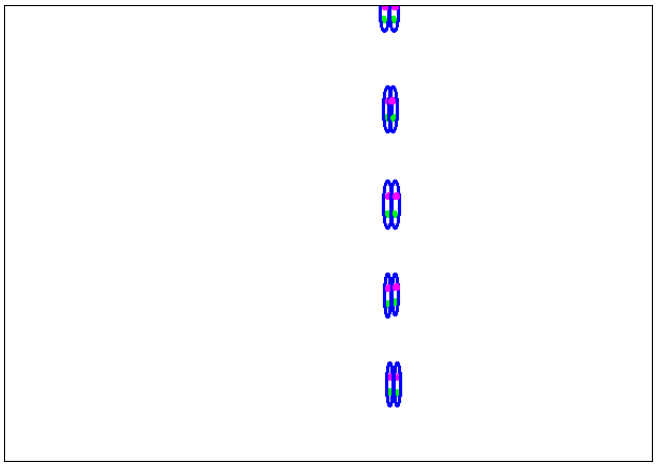


Figura 3.26: Asociación de esquinas.

Algoritmo 5: Asociación de esquinas.

Entrada: V_1=Vector de esquinas anteriores,V_2=Vector de esquinas nuevo
Salida : P_1=Puntos asociados de V_1, P_2=Puntos asociados de V_2

```
1  para i=0;i<tamaño V1;i++ hacer
2      para j=0;j<tamaño de V2;j++ hacer
3          si Distancia(V1[i], V2[j])<Distancia máxima entonces
4              si (Distancia(Esquina 0 y Esquina 3 de V1[i])>Distancia mínima y
                 Distancia(Esquina 0 y Esquina 3 de V2[j])>Distancia mínima) entonces
5                  Agrega en P1(esquinas de V1[i])
6                  Agrega en P2(esquinas de V2[j])
7              en otro caso
8                  si (Ubicación de V1[i] y V2[j] cercano a y_imagen = 0) entonces
9                      Agrega en P1(esquina_0 y esquina_1 de V1[i])
10                     Agrega en P2(esquina_0 y esquina_1 de V2[j])
11                 en otro caso
12                     Agrega en P1(esquina_2 y esquina_3 de V1[i])
13                     Agrega en P2(esquina_3 y esquina_3 de V2[j])
14                 fin
15             fin
16         fin
17     fin
18 fin
```

3.6. Reconstrucción del mapa

Encontradas las esquinas presentes en dos frames consecutivos, así como su relación, el paso siguiente es encontrar la matriz de transformación entre ellas. Para esta etapa se utilizó una variación del método descrito en (Arun et al., 1987), donde realizan el ajuste de dos colecciones de puntos tridimensionales por medio de una transformación de similaridad. Este algoritmo tiene como primer paso centrar los datos, es decir calcular la media de cada conjunto y restarla a cada punto. Posteriormente, usando la descomposición en valores singulares se obtiene la matriz de rotación que relaciona ambas colecciones. Por último se realiza el cálculo de la traslación usando la matriz de rotación y las medias de las componentes.

Después de varias pruebas se decidió por utilizar la medición de la orientación proporcionada por el sensor IMU. De esta forma cada vez que se captura una imagen nueva se estima la nueva rotación a partir de la resta de la orientación nueva menos la anterior. Esta información es obtenida subscribiéndose al tópico */model_car/yaw*.

Obtener el incremento en la orientación por cada frame capturado requiere de ciertas consideraciones. Esto es debido a la forma en que el sensor IMU otorga la medición del angulo. Los ángulos medidos están dados en grados y van de 0° a −180° y de 180° a 0°. Esta es la razón por la que hay que hacer una conversión entre el valor otorgado por el sensor y la medición real, además de que la posición del vehículo al iniciar el recorrido no es cero. Todas estas consideraciones son resueltas por el algoritmo 7.

Algoritmo 6: Cálculo de matrices de transformación entre imágenes sucesivas.

Entrada: Esquinas(puntos) asociados P_1(tamaño n) y P_2(tamaño m)
Salida : H: Matriz de transformación

1 P_{m1}=punto medio (x,y)de P_1.
2 P_{m2}= punto medio (x,y)de P_2. A=Matriz($2\times n$)
3 B=Matriz($2\times m$)
4 Acomodo de puntos de P_1 en A
5 Acomodo de puntos de P_2 en B
6 $A_m = A - P_{m1}$
7 $B_m = B - P_{m2}$
8 $D = A_m * B_m^T$
9 $U, S, v_t = SVD(D)$
10 $R = v_t^T * U^T$
11 $t = -R * P_{m1} + P_{m2}$
12 $H = \begin{bmatrix} R & t \\ 0^{1\times 2} & 1 \end{bmatrix}$

Algoritmo 7: Cálculo de incremento de la orientación.

Entrada: θ_{nuevo}, $\theta_{anterior}$
Salida : $\nabla\theta$

global bandera=1 **si** *bandera==1* **entonces**
    bandera=0
    $\theta_{inicial}$=$\theta_{nuevo} * (\pi/180)$
    $\theta_{anterior}$=$\theta_{inicial}$
    $\nabla\theta = 0$
en otro caso
    $aux = \theta_{nuevo} * (\pi/180)$;
    si $aux <= \theta_{anterior}$ **entonces**
        $\theta_{nuevo} = \theta_{inicial} - aux$;
    en otro caso
        $\theta_{nuevo} = 2 * \pi + \theta_{inicial} - aux$
    fin
    $\nabla\theta = \theta_{nuevo} - \theta_{anterior}$
fin

Capítulo 4

Resultados experimentales

En este capítulo se describen los resultados de la metodología propuesta para la reconstrucción de mapas, mostrada en la figura 3.1. Dicha metodología fue puesta a prueba en escenarios virtuales y reales.

Las etapas de la metodología propuesta son: la adquisición de imágenes, la detección de carriles, la asociación de datos y, principalmente, la reconstrucción del mapa. La primera etapa corresponde al conocimiento y calibración de cámaras, cuyas operaciones fueron descritas en el capítulo anterior.

Dentro de las operaciones de la segunda etapa, se encuentra la corrección de perspectiva, una transformación de imágenes entre planos base para las dos técnicas de detección de carriles.

En los resultados de la última etapa de la metodología, correspondiente a la reconstrucción del mapa, se presentan, primero, las reconstrucciones de escenarios virtuales correspondientes a 3 pistas distintas simuladas en Gazebo. Posteriormente, se describe el resultado de la reconstrucción del mapa de una pista a bordo de la plataforma AutoNOMOS mini. Esta pista, mostrada en la figura 4.1 fue diseñada y construida en el Laboratorio de Robótica de la Universidad Tecnológica de la Mixteca, siguiendo medidas según reglamentos de las competencias Torneo Mexicano de Robótica y Talent Land.

4.1. Condiciones de implementación para la reconstrucción de la pista real

Los algoritmos descritos en el capítulo 3 fueron implementados en el vehículo AutoNOMOS mini, usando para ello las imágenes capturadas por la cámara Intel Real Sense RGB de tamaño 640×480 a 60 fps. El vehículo fue teleoperado con una velocidad máxima de 300 rpm, equivalente a 0.177 m/s. Las mediciones de la orientación fueron obtenidas del sensor IMU 6050 a bordo del robot.

Figura 4.1: Pista construida en el Laboratorio de Robótica Inteligente.

4.2. Organización del algoritmo en ROS

Los algoritmos diseñados para la reconstrucción del mapa y descritos en el capítulo anterior se agruparon en un solo nodo llamado *mapa_nodo* el cual puede personalizarse para simulación y vehículo real en un archivo *launch* por medio de los siguientes parámetros.

- Ángulo de rotación en X
- Ángulo de rotación en Y
- Ángulo de rotación en Z
- Distancia en Z
- Distancia en Y
- Distancia focal horizontal f_u
- Distancia focal vertical f_v
- Coordenada del centro óptico horizontal c_u
- Coordenada del centro óptico vertical c_v

La reconstrucción se lleva a cabo mientras el vehículo es teleoperado usando el nodo *teleopera_teclado* con el que se controla la dirección y avance del robot por medio del teclado.

Antes de empezar cualquier recorrido se coloca la cámara en posición, esto es se mueve para que coincida con la cuadrícula de la corrección de perspectiva. De igual forma que con el nodo *mapa_ nodo*, se personaliza para simulación o real con los parámetros de calibración (f_u, f_v, c_u y c_v)

4.3. Corrección de perspectiva

Para la corrección de perspectiva se realizó de manera distinta en simulación y con la plataforma física. Si bien en los dos casos se aplicó una cuadrícula sobre la imagen corregida, los parámetros obtenidos no fueron los mismos.

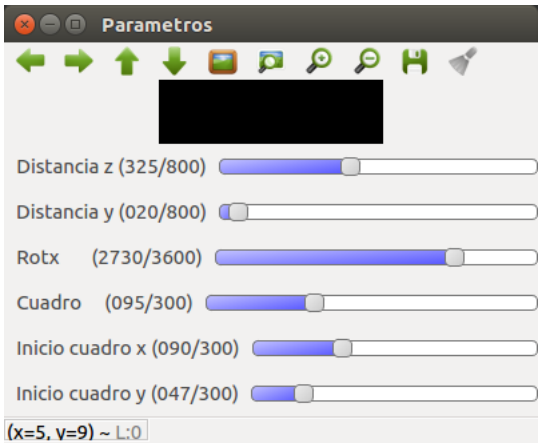
Como se mencionó anteriormente se implementaron dos métodos para la detección de carriles en la plataforma. De igual forma también se obtuvo la corrección de perspectiva de dos maneras. En la primera, se calcularon 8 parámetros correspondientes a 3 ángulos de rotación y 3 traslaciones en x, y, y z, así como 2 valores correspondientes a los escalamientos en los ejes X y Y de las imágenes.

En la segunda opción de corrección de perspectiva solo eran dos parámetros para el caso real y tres para la simulación. Esto es debido a que en simulación el ángulo de visión es mucho mayor que en el espacio real, además de que no es posible calibrar la cámara virtual. En el caso real, los valores a calcular eran solo la rotación en el eje X y la traslación en el eje Z cuyo valor representa aproximadamente la distancia en mm del plano al centro focal de la cámara.

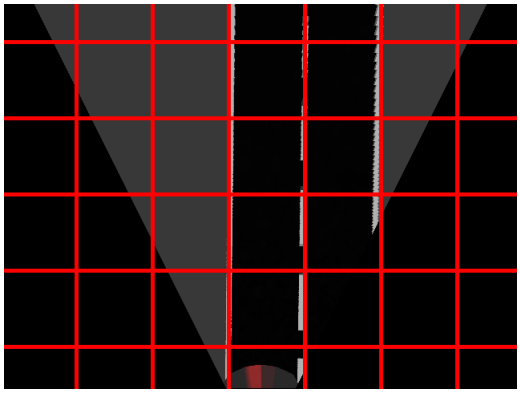
En el caso real los parámetros usados en la corrección de perspectiva corresponden a una rotación de 286° y una traslación de 298 mm, como puede verse en la figura 4.3. En el robot simulado los valores se ajustaron a una rotación en X de 273°, una traslación en Z de 325 y en Y de 20, tal como se muestra en la figura 3.10.

$$H_{real} = \begin{bmatrix} 594.6516800000001 & -294.2788133371886 & -18985.060353804 \\ 0 & -71.71797363104531 & 90245.4661862081 \\ 0 & -0.9612616975104943 & 532.6369833384155 \end{bmatrix} \tag{4.1}$$

$$H_{sim} = \begin{bmatrix} 640 & -319.5614512126457 & -24105.25170896499 \\ 0 & -206.1760798937173 & 140282.2591744922 \\ 0 & -0.998629535039518 & 564.6710884094844 \end{bmatrix} \tag{4.2}$$



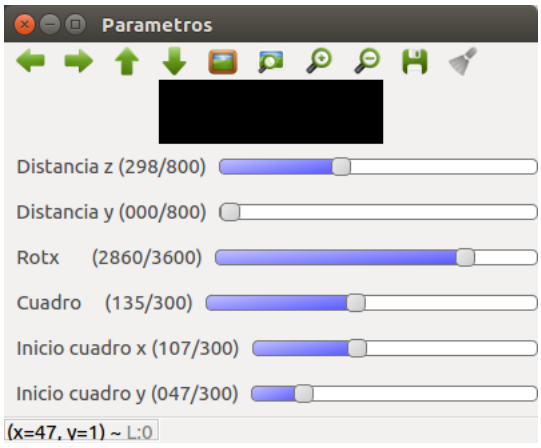
(a) Ajuste de parámetros.



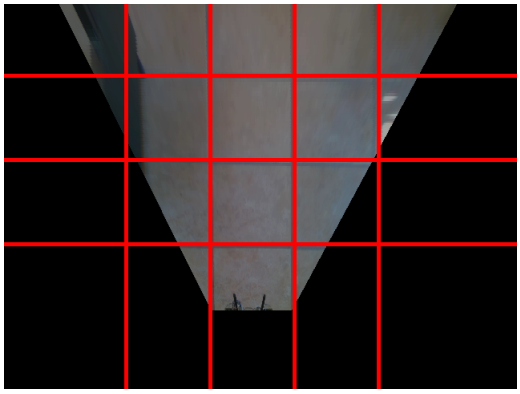
(b) Cuadrícula

(c) Resultado.

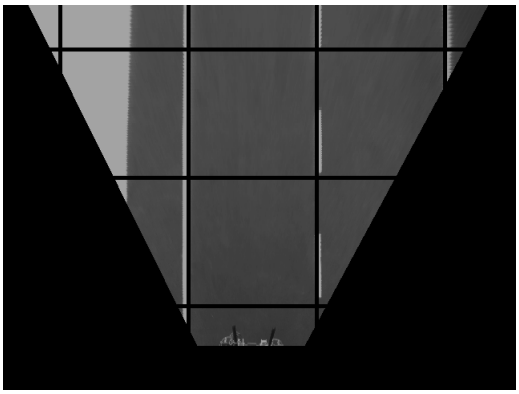
Figura 4.2: Cálculo de corrección de perspectiva en el ambiente virtual.



(a) Ajuste de parámetros .



(b) Calibración con una cuadrícula.

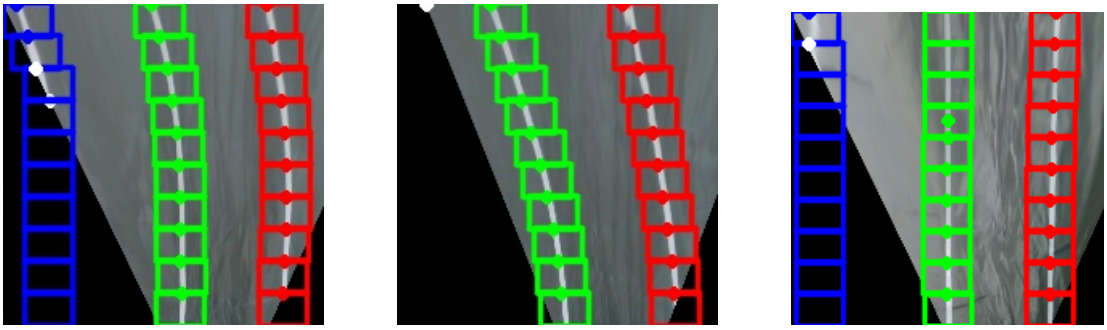


(c) Calibración en la pista.

Figura 4.3: Calculo de corrección de perspectiva en la plataforma real.

4.4. Detección de carriles

Se implementaron dos métodos distintos para este fin: el de ventanas y el de detección de componentes. En la figura 4.4 se muestra la detección del primer método en la pista real en tres situaciones distintas dependiendo del ángulo de visión de la cámara. Del mismo modo en la figura 4.5 se puede ver la misma situación con el método de detección de componentes.

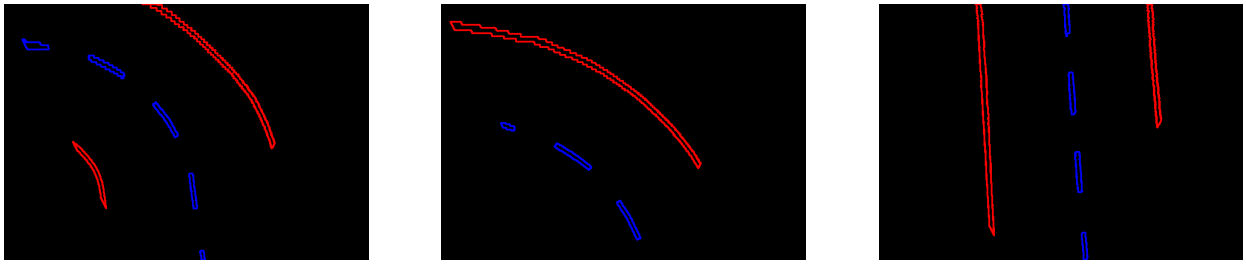


(a) Detección curva (3 líneas).

(b) Detección curva (2 líneas).

(c) Detección recta (3 líneas)

Figura 4.4: Detección de carriles usando ventanas.



(a) Detección curva (3 líneas).

(b) Detección curva (2 líneas).

(c) Detección recta (3 líneas)

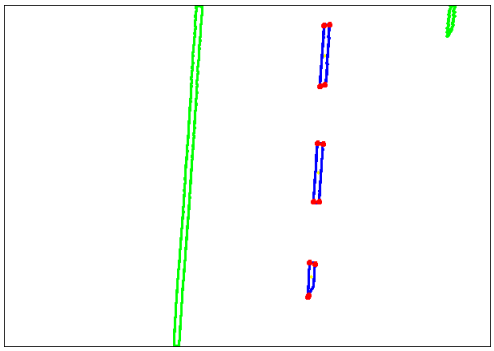
Figura 4.5: Detección de carriles usando componentes.

4.5. Detección de esquinas

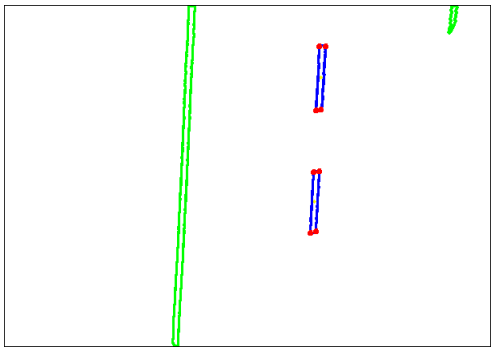
Las esquinas de cada componente central se consideran puntos claves para el cálculo de la relación entre imágenes. La figura 4.6 muestra el resultado de este procedimiento. Como se puede observar en la figura 4.6 a) cuando en la imagen es visible solo una parte de un componente central, la ubicación de dos esquinas coincide como en el caso del componente inferior. Sin embargo, esto es discriminado en la siguiente etapa.

En la figura 4.6 b) se ve el caso donde la detección solo se centra en dos componentes presentes y es realizada de forma correcta.

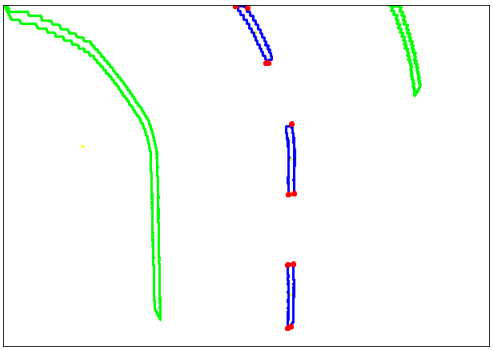
Cuando el vehículo esta en una curva, la detección de esquinas es más susceptible de presentar errores, como es el caso de colocar dos esquinas en el mismo punto, como es el caso de la figura 4.6 c). Estos errores tienen influencia en el cálculo de la transformación entre imágenes.



(a) Esquinas detectadas cuando hay 3 componentes.



(b) Esquinas detectadas en presencia de 2 componentes.



(c) Esquinas detectadas en una curva.

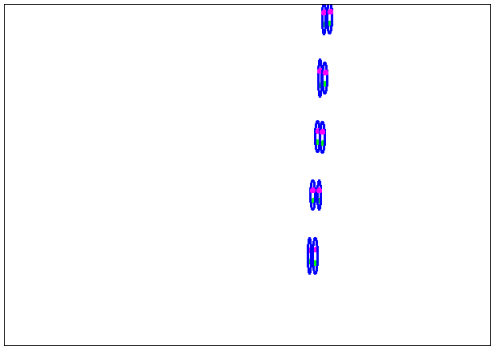
Figura 4.6: Detección de esquinas.

4.6. Asociación de imágenes por medio de las esquinas

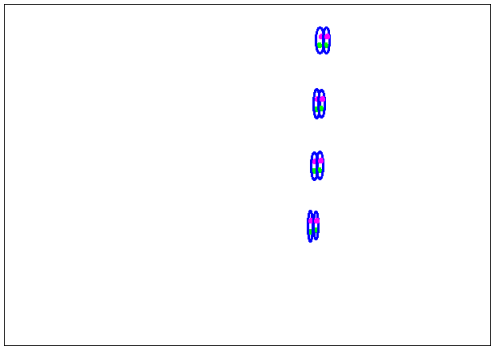
Al relacionar las esquinas detectadas en dos frames consecutivos se presentaron distintas situaciones dependiendo de la zona donde se ubicaba el vehículo. Cuando el robot se traslada en la parte recta de la pista la correspondencia entre puntos es fácilmente detectable, mientras que en las curvas el error en la relación correcta crece.

En la figura 4.7 se puede observar la asociación de esquinas entre dos frames sucesivos. Las elipses encierran dos puntos que corresponden a la ubicación de la misma esquina en un frame anterior y uno nuevo. La figura 4.7 a) corresponde a la asociación de la esquinas detectadas en el frame correspondiente a la figura 4.6 a) con su consecutivo. Aunque se detectaron 11 esquinas, la asociación solo toma 10, dado que el componente inferior esta incompleto.

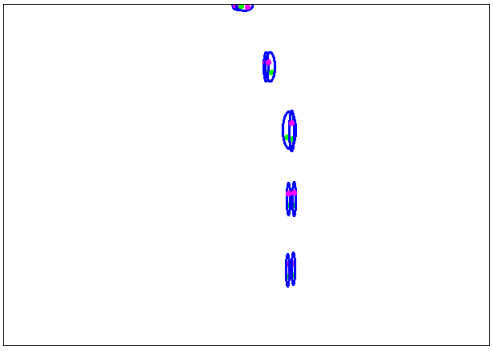
Los aciertos y errores de la detección de esquinas tienen un efecto en esta etapa. Si el resultado es correcto la relación encontrada también lo es, como el caso de las figuras 4.7(a) y (b). En caso contrario el efecto es notorio, como el caso de la figura 4.7(c).



(a) Asociación de esquinas en rectas con tres componentes.



(b) Asociación de esquinas en rectas con dos componentes.



(c) Asociación de esquinas en curvas.

Figura 4.7: Asociación de esquinas.

4.7. Reconstrucción del mapa

Como etapa final y clave de este trabajo se tiene la reconstrucción del mapa. Al principio se había propuesto realizarla unicamente con información de las imágenes pero como puede verse en la figura 4.8 la acumulación del error en el cálculo de la rotación era notoria. El cálculo de la orientación aumentaba su error despues de un ángulo aproximado de 180°. Debido a esto se reemplazó el ángulo calculado con la medida de la orientación obtenida del IMU. En las siguientes secciones se describen los resultados de esta modificación.

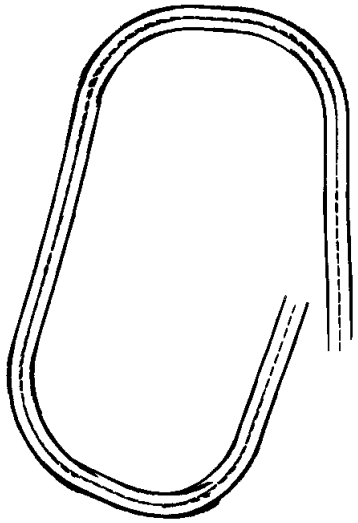


Figura 4.8: Mapa incorrecto basado solo en información de imágenes.

4.7.1. Resultados en escenarios virtuales

En la herramienta gráfica Gazebo se construyeron 3 modelos de pista en los cuales se teleoperó el vehículo, un ejemplo de estos modelos virtuales se muestra en la figura 4.9 .

Las etapas descritas en el desarrollo se implementaron en un solo nodo con distintas características en simulación y real. Para describir el proceso de construcción del mapa en simulación se presentan una serie de imágenes de distintas partes del circuito. Se tomó como referencia la pista 3, mostrada en la figura 4.15, que contiene cuatro curvas con los mismos radios y cuatro partes rectas. Se muestran las etapas principales: Adquisición de imágenes, segmentación, corrección de perspectiva, detección de carriles, aproximación polinomial y clasificación de línea central, detección de esquinas y reconstrucción del mapa.

La primera colección de imágenes, mostrada en la figura 4.10, muestra dichas etapas hasta el frame 200. La reconstrucción corresponde a una sección de la parte recta de la pista 3, cuyo

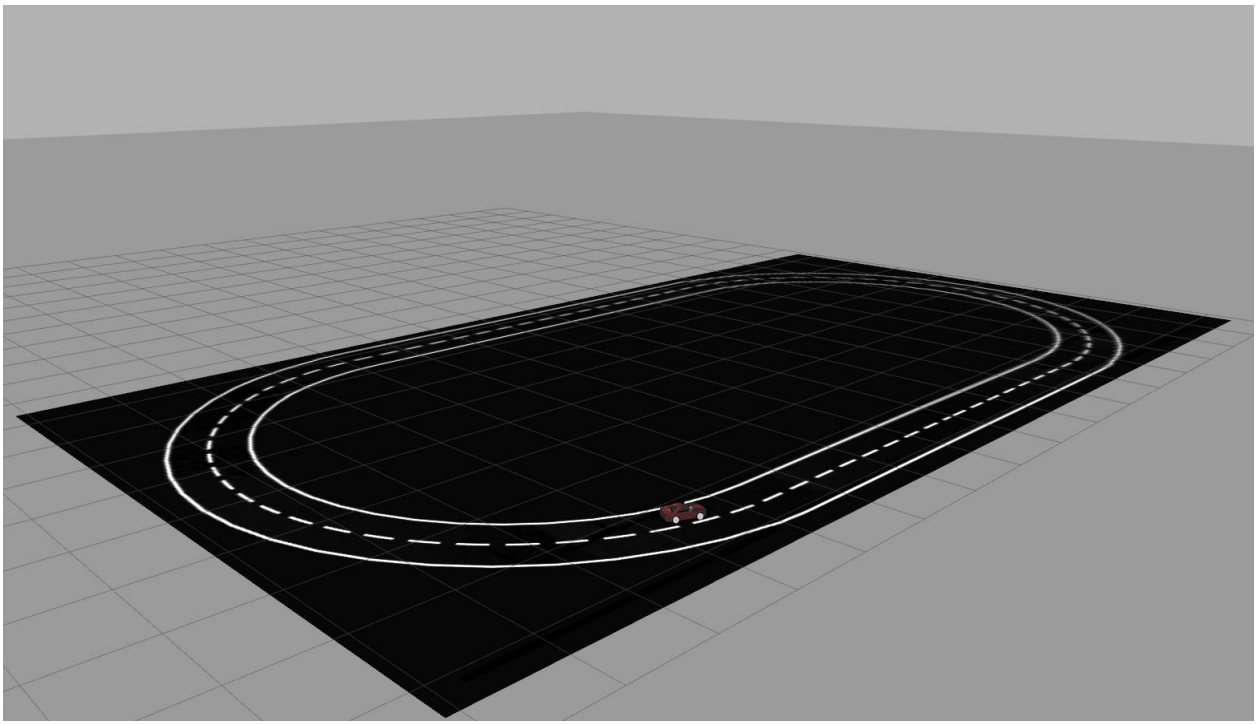
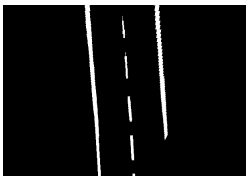


Figura 4.9: Ambiente virtual en Gazebo.

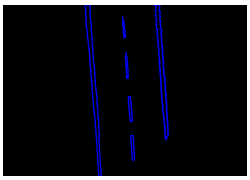
punto de inicio esta marcado con un recuadro rojo en la figura 4.16 a).

(a) Imagen original.

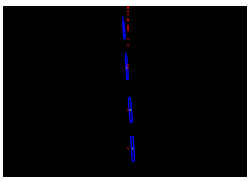
(b) Umbralado.



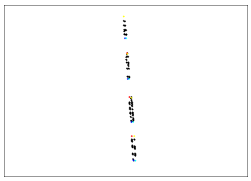
(c) Corrección de perspectiva.



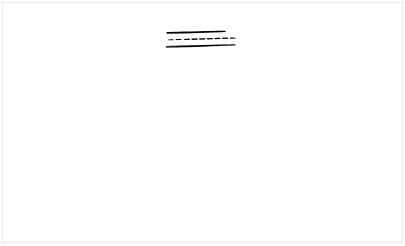
(d) Detección de carriles.



(e) Aproximación polinomial.



(f) Detección de esquinas.



(g) Reconstrucción del mapa.

Figura 4.10: Proceso de construcción del mapa, frame 200.

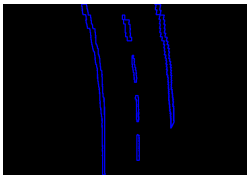
La siguiente etapa corresponde a 600 frames, y corresponde al inicio de la primer curva

a a partir del punto de salida. La reconstrucción de esta etapa es una línea recta, como puede verse en 4.11.

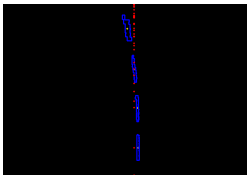
(a) Imagen original.

(b) Umbralado.

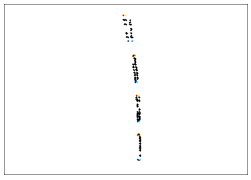
(c) Corrección de perspectiva.



(d) Detección de carriles.



(e) Aproximación polinomial.



(f) Detección de esquinas.

(g) Reconstrucción del mapa.

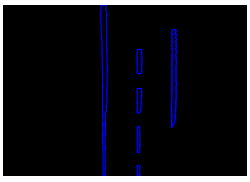
Figura 4.11: Proceso de construcción del mapa, frame 600.

Una vez pasados los 2000 frames se tiene más de la mitad del mapa, este avance esta mostrado en la figura 4.12.

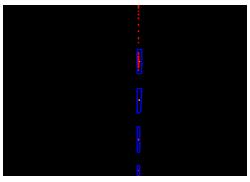
(a) Imagen original.

(b) Umbralado.

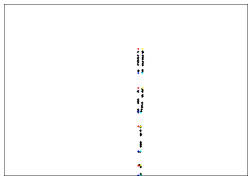
(c) Corrección de perspectiva.



(d) Detección de carriles.



(e) Aproximación polinomial.



(f) Detección de esquinas.

(g) Reconstrucción del mapa.

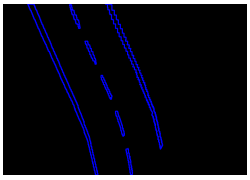
Figura 4.12: Proceso de construcción del mapa, frame 2000.

La reconstrucción total es alcanzada cuando el número de frames llega a 3000. En la figura 4.13 se encuentran las imágenes correspondientes al procesamiento en ese frame, así como la reconstrucción total en la figura 4.13 g).

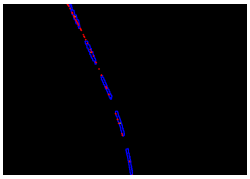
(a) Imagen original.

(b) Umbralado.

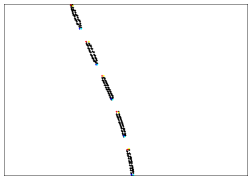
(c) Corrección de perspectiva.



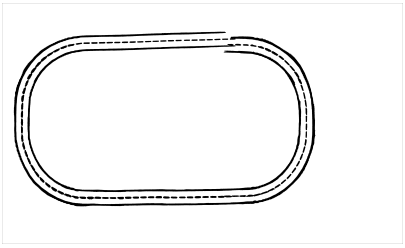
(d) Detección de carriles.



(e) Aproximación polinomial.



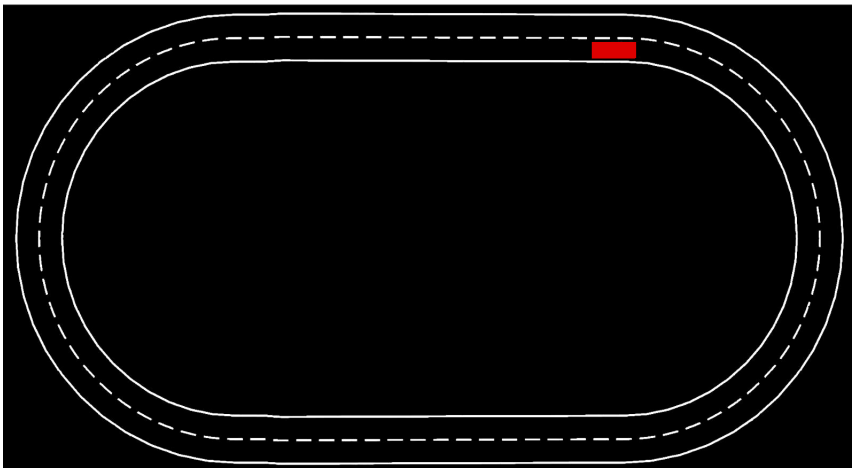
(f) Detección de esquinas.



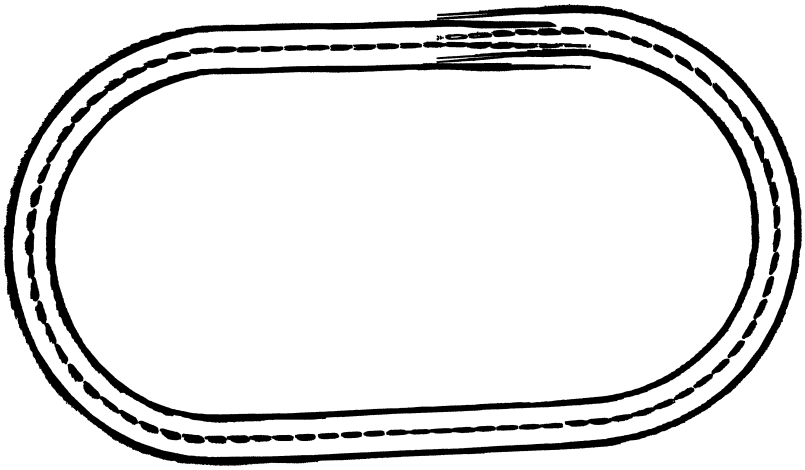
(g) Reconstrucción del mapa.

Figura 4.13: Proceso de construcción del mapa, frame 3000.

En las figuras 4.14, 4.15 y 4.16 se muestra la reconstrucción de tres pistas distintas. El inicio del recorrido se indica con un recuadro rojo. De estos tres modelos el mayor error obtenido, en cuanto a la orientación final del vehículo, corresponde a la pista 1, cuyo final e inicio del recorrido encaja con una diferencia visible.

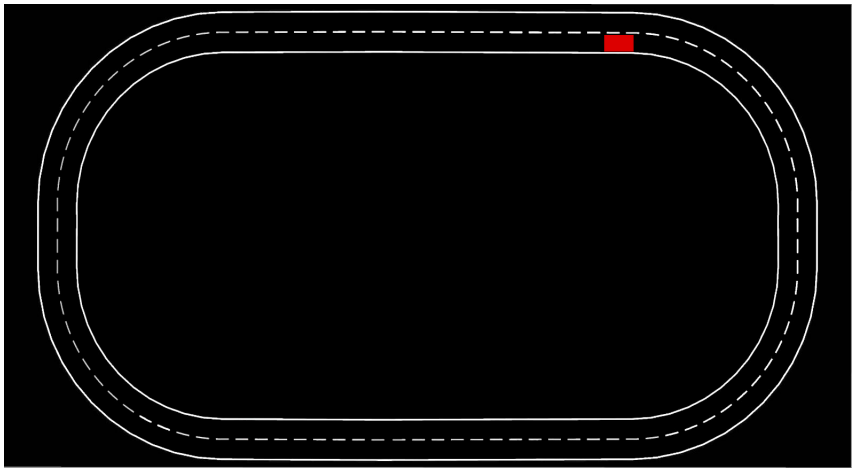


(a) Pista virtual 1.

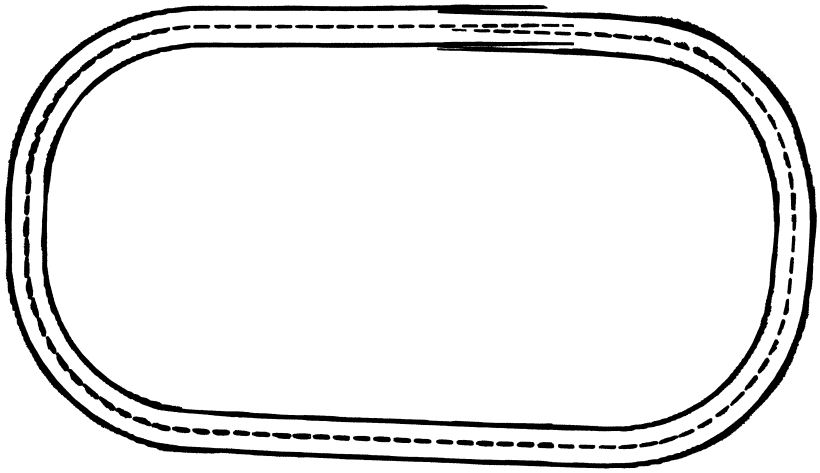


(b) Mapa 1.

Figura 4.14: Reconstrucción del mapa 1.

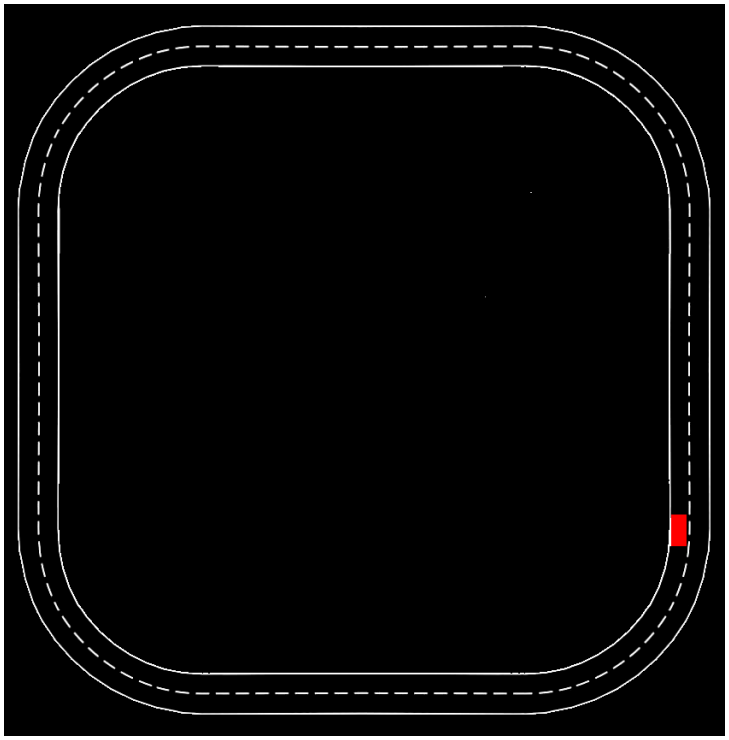


(a) Pista virtual 2.

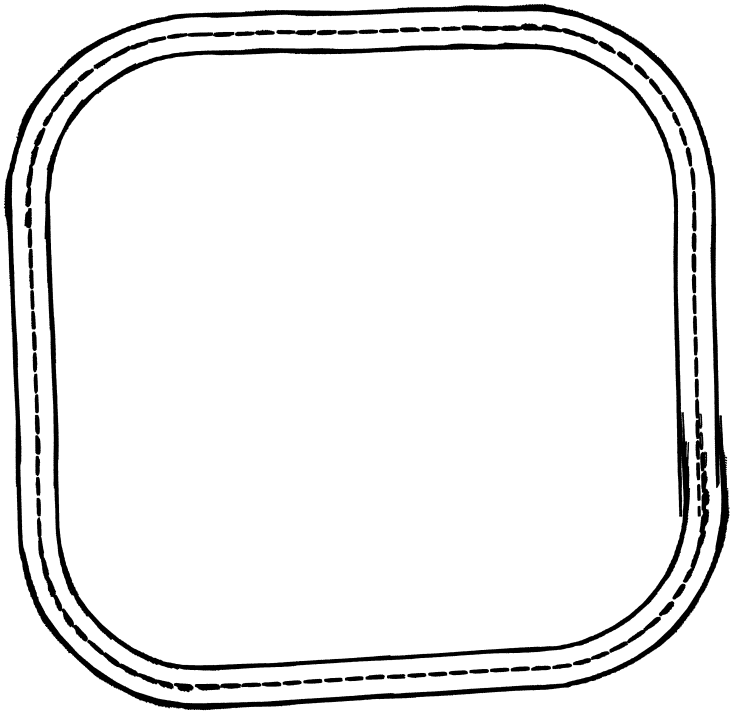


(b) Mapa 2.

Figura 4.15: Reconstrucción del mapa 2.



(a) Pista virtual 3.

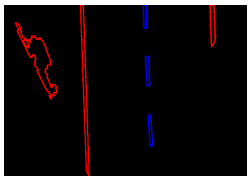


(b) Mapa 3.

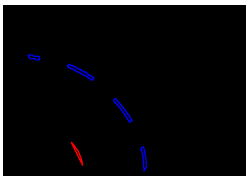
Figura 4.16: Reconstrucción del mapa 3.

4.7.2. Resultados en la pista real

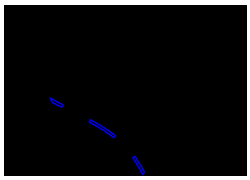
De igual forma que en las pistas del mundo virtual, la pista real se reconstruye con base en las etapas de la figura 3.1. También se incluyen una serie de imágenes para mostrar el proceso de construcción del mapa. A diferencia de las pistas virtuales, en la pista real se presenta un mayor número de errores en la detección de carriles. Estos van de la confusión de áreas del fondo con carriles a la falta o perdida de información debido al ángulo de visión del carro. En la figura 4.17 se muestran estas equivocaciones, siendo la imagen a) y d) ejemplos de errores en clasificación de carriles, mientras que b) y c) muestran la falta de información.



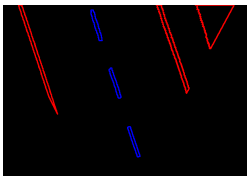
(a) Contornos en el frame 5.



(b) Contornos en el frame 240.



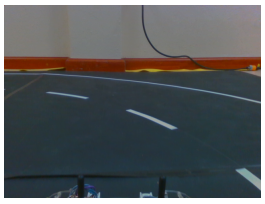
(c) Contornos en el frame 305.



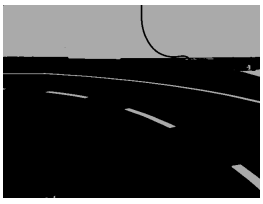
(d) Contornos en el frame 920.

Figura 4.17: Errores en la detección de carriles.

La colección de imágenes de la figura 4.18 corresponden a la reconstrucción de los primeros 250 frames capturados por la cámara. Se observan las etapas realizadas sobre la imágen 250, así como los efectos del error en la clasificación de carriles en la reconstrucción del mapa.

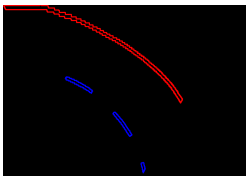


(a) Imagen original.

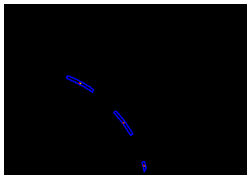


(b) Umbralado.

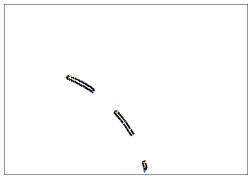
(c) Corrección de perspectiva.



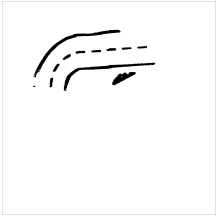
(d) Detección de carriles.



(e) Aproximación polinomial.



(f) Detección de esquinas.



(g) Reconstrucción del mapa.

Figura 4.18: Proceso de construcción del mapa real, frame 250.

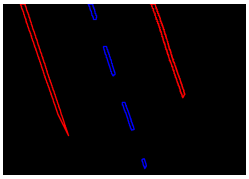
Dado que la pista construida es pequeña el número de frames capturados en su recorrido

completo apenas si supera las 1000 imágenes. Cuando el número de frames a llegado a 500 se esta casi a la mitad del circuito, tal como se muestra en las imágenes de la figura 4.19.

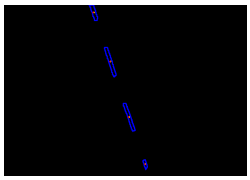
(a) Imagen original.

(b) Umbralado.

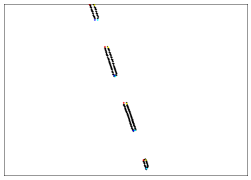
(c) Corrección de perspectiva.



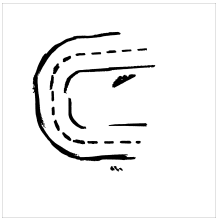
(d) Detección de carriles.



(e) Aproximación polinomial.



(f) Detección de esquinas.



(g) Reconstrucción del mapa.

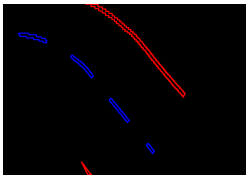
Figura 4.19: Proceso de construcción del mapa real, frame 500.

En las imágenes de la figura 4.20 se muestra el avance cuando el número de frames ha alcanzado 750. Puede verse la influencia de la falta de información en la discontinuidad de la línea interna.

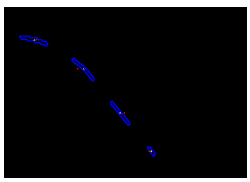
(a) Imagen original.

(b) Umbralado.

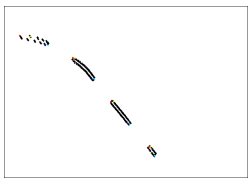
(c) Corrección de perspectiva.



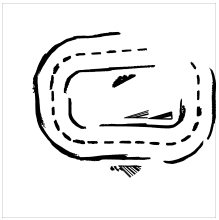
(d) Detección de carriles.



(e) Aproximación polinomial.



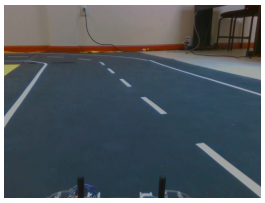
(f) Detección de esquinas.



(g) Reconstrucción del mapa.

Figura 4.20: Proceso de construcción del mapa real, frame 750.

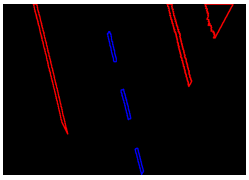
El mapa terminando finalmente puede verse en la figura 4.21 g), junto con las distintas etapas aplicadas al frame 1000.



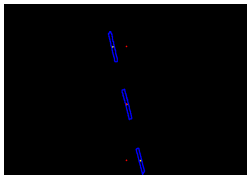
(a) Imagen original.

(b) Umbralado.

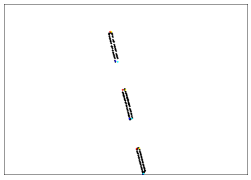
(c) Corrección de perspectiva.



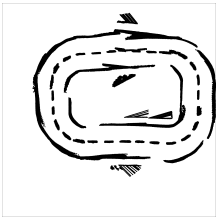
(d) Detección de carriles.



(e) Aproximación polinomial.



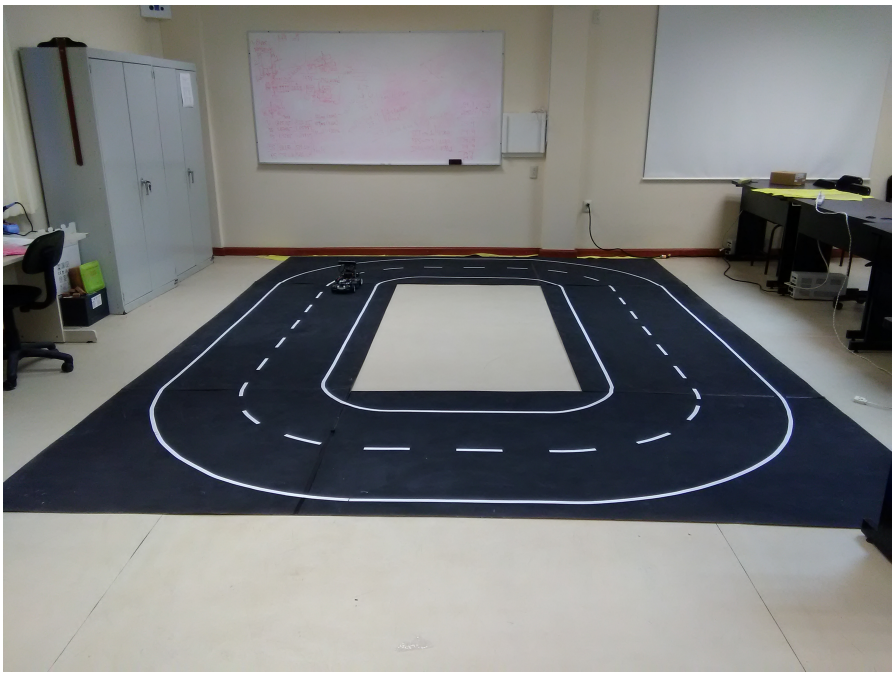
(f) Detección de esquinas.



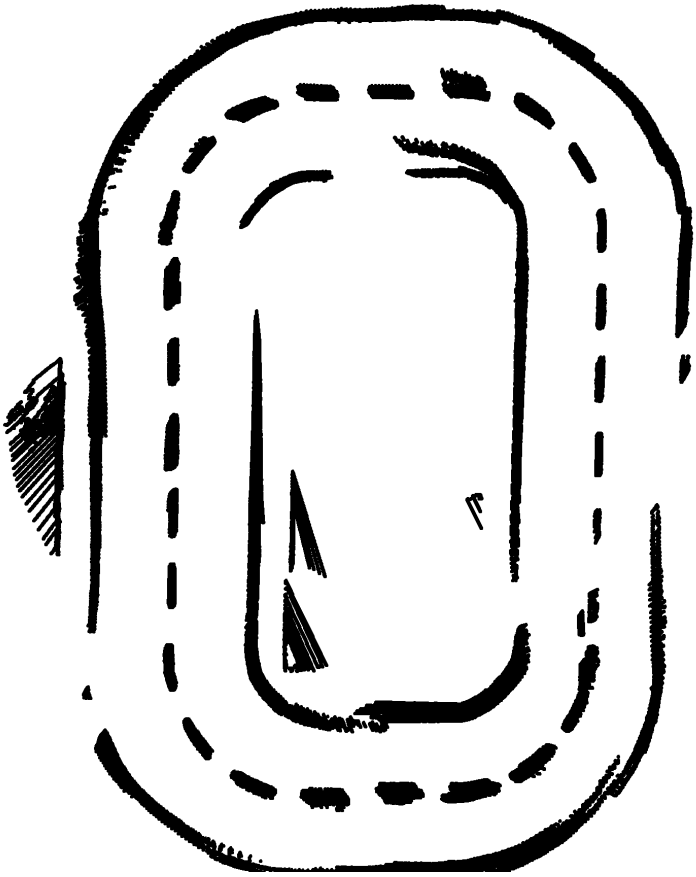
(g) Reconstrucción del mapa.

Figura 4.21: Proceso de construcción del mapa real, frame 1000.

La pista real sobre la que se condujo el vehículo se muestra en la figura 4.22(a), así como una reconstrucción de la misma en la figura 4.22(b). En el mapa final se observan puntos que no corresponden a las líneas de carril, que son un efecto de la confusión en la clasificación de componentes. También es visible el efecto de la conducción errónea al salir de la primera curva a partir del punto de salida. La discontinuidad de las líneas se debe, principalmente, a la pérdida de las líneas en el campo de visión de la cámara de la plataforma.



(a) Pista real.



(b) Mapa real.

Figura 4.22: Reconstrucción del mapa de la pista real.

Capítulo 5

Conclusiones y trabajos a futuro

A continuación se describen las conclusiones obtenidas y algunos trabajos futuros a implementar con los resultados y para mejorar el sistema.

5.1. Conclusiones

Durante el presente trabajo se implementaron dos formas de obtener la corrección de perspectiva, la primera realizando el ajuste de los parámetros de rotación, traslación y escalamiento. El segundo solamente ajustando dos parámetros, la traslación en Z y la rotación en X en el caso real, mientras que en la simulación se agregó una traslación en y. Esto se debe a que se supone es una cámara ideal y los parámetros de calibración no se calcularon.

En la detección de carriles, si bien se implementaron dos métodos, solo se realizó la reconstrucción con los resultados del método de detección de componentes. El primero, basado en el ventanas, se diseñó para ser usado en líneas continuas totalmente, mientras que el segundo esta basado en la discontinuidad de la línea central. Al ser líneas continuas, en el primer caso, cuando estás son rectas, visualmente no hay variación en cuanto a la estimación de la traslación entre frame y frame.

En el segundo método la variación es visible entre cada componente central de forma que se pudo tomar como puntos clave las esquinas de cada rectángulo correspondiente a la línea del centro. Sin embargo el cálculo de la matriz de rotación generaba muchos errores cuando el ángulo de orientación superaba 180°, por lo que el error acumulado era bastante notorio. Debido a esto se decidió tomar la medición de la orientación del vehículo dada por el IMU para la estimación del ángulo de rotación.

En las etapas de detección y asociación de esquinas los mejores resultados se obtienen en líneas rectas y con componentes completos, mientras que en presencia de curvas los errores aumentan. Estos errores tienen influencia cuando se calcula el punto medio para encontrar la traslación entre dos frames sucesivos.

Tal como se muestra en la figuras 4.14, 4.15, 4.16 y 4.22, se logró la reconstrucción completa de las pistas en escenarios virtuales y real, respectivamente. Se había propuesto

inicialmente una o más vueltas en la reconstrucción del mapa, sin embargo en la práctica solo fue necesaria una vuelta completa.

Al realizar la reconstrucción del mapa en los escenarios virtuales, aunque no se midió el ángulo real, a partir de la estimación aproximada de manera visual y con el ángulo de salida final el error mayor en orientación observado fue menor a 7°. Esto se debe a que en la segmentación de carriles en el caso virtual no tiene pérdida de información en líneas ni presencia de pixeles que no corresponden.

Cuando las componentes de las líneas centrales son detectadas correctamente la asociación de puntos funciona aún cuando el vehículo se cambie de carril, siempre y cuando el número de componentes sea mínimo 2. De igual forma en el cálculo de la orientación los cambios bruscos generan errores en la estimación del giro del vehículo, lo cual afecta a las transformaciones.

En la pista real se tienen errores de segmentación debido al fondo en el que esta montada la pista, cuyo color es muy cercano al blanco. Estos errores agregan pixeles erróneos a la reconstrucción, sin embargo son pocos en comparación a los correctos. Como se puede observar en la 4.22(b) en algunas partes de las líneas derecha e izquierda hay discontinuidades. Estas se deben a errores de clasificación en líneas rectas y a perdida de información por el campo de visión de la cámara. En general el mapa obtenido tiene una buena aproximación con la pista real.

5.2. Trabajos a futuro

A partir de los resultados obtenidos y en busca de mejorar la metodología propuesta en la reconstrucción del mapa se proponen algunos trabajos futuros.

- Implementar el detector de carriles basado en ventanas estimando la traslación con odometría y la rotación con la medición de la orientación por medio del IMU.
- Agregar pistas más complicadas, es decir, con cruces y obstáculos.
- Implementar algoritmos de localización y navegación a partir de los mapas.
- Realizar la construcción del mapa mientras se conduce de manera autonoma la plataforma AutoNOMOS mini.
- Hacer fusión de mediciones de odometría y orientación obtenidas de las imágenes con las de los sensores de la plataforma.

Glosario

Componentes: Contorno que representa parte de una línea correspondiente a un carril.

Gazebo: Herramienta presente en ROS que permite virtualizar ambientes y robots, tomando en cuenta parámetros físicos y restricciones.

Homografía: rectificación de imágenes corrigiendo el efecto de la perspectiva.

IMU: Unidad de Medición Inercial,conjunto de sensores conformado por acelerómetros, giroscopios, magnetómetro, termómetro y barómetro cuya función es brindar información para determinar la posición de un agente.

Launch: archivo lanzador en ROS que permite iniciar uno o más nodos a la vez, así como personalizarlos con parámetros en cada ejecución.

Mapas métricos: representación del ambiente mediante formas y distancias geométricas.

Mapas semánticos: representación de entornos basada en etiquetas con atributos lingüísticos.

Mapas topológicos: mapas que modelan entornos de forma cualitativa considerando lugares y sus conexiones entre ellos.

Nodo: Código ejecutable desarrollado en Python o C++, que constituye el nivel más bajo en ROS.

Odometría: método estima los cambios de posición de objetos en el tiempo a partir de datos obtenidos de sensores como encoders.

Paquete: contiene los programas ejecutables y lanzadores que permiten la inicialización de nodos.

ROS: sistema operativo para robots que provee librerías y herramientas para la creación de aplicaciones para robots.

Tópico: enlace de comunicación entre nodos.

Ventana: porción de una imagen, definida por un rectángulo y un punto de inicio.

Referencias

Aguilar, D. (2014). Construcción de mapas probabilísticos mediante técnicas de slam en entorno ros. Master's thesis, Universidad de Alcalá.

Aly, M. (2008). Real time detection of lane markers in urban streets. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 7–12. IEEE.

Arce, F., Zamora, E., Hernández, G., and Sossa, H. (2017). Efficient lane detection based on artificial neural networks. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 4.

Arun, K. S., Huang, T. S., and Blostein, S. D. (1987). Least-squares fitting of two 3-d point sets. *IEEE Transactions on pattern analysis and machine intelligence*, (5):698–700.

AutoNOMOS (2017). https://github.com/automodelcar/automodelcarwiki/wiki/hardware-(autonomos-model-v2). Último acceso: 10 enero de 2018.

Bae, J.-H. and Song, J.-B. (2011). Monocular vision-based lane detection using segmented regions from edge information. In *Ubiquitous Robots and Ambient Intelligence (URAI), 2011 8th International Conference on*, pages 499–502. IEEE.

Ballew, G. (2018). Sdc lane and vehicle detection tracking. `https://github.com/galenballew/SDC-Lane-and-Vehicle-Detection-Tracking`.

Bender, P., Ziegler, J., and Stiller, C. (2014). Lanelets: Efficient map representation for autonomous driving. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pages 420–425. IEEE.

Benítez, F. C. (2007). *Homography based techniques for unmanned aerial vehicle self-localization using monocular imagery*. PhD thesis, Universidad de Sevilla.

Bravo Conejo, C. G. (2018). Navegación autónoma de un robot tipo automóvil en pista de carreras con obstáculos. Master's thesis, Centro de Investigación en Computación IPN.

de la Escalera, A., Armingol, J. M., Pech, J. L., and Gómez, J. J. (2010). Detección automática de un patrón para la calibración de cámaras. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, 7(4):83–94.

Gallardo López, D. (2000). *Aplicación del muestreo bayesiano en robots móviles: estrategias para localización y estimación de mapas del entorno*.

González, J. (1999). Visión por computador. *Paraninfo, Madrid*.

Guo, C., Kidono, K., Meguro, J., Kojima, Y., Ogawa, M., and Naito, T. (2016). A low-cost solution for automatic lane-level map generation using conventional in-car sensors. *IEEE Transactions on Intelligent Transportation Systems*, 17(8):2355–2366.

Guo, C., Mita, S., and McAllester, D. (2010). Lane detection and tracking in challenging environments based on a weighted graph and integrated cues. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5543–5550. IEEE.

Gwon, G.-P., Hur, W.-S., Kim, S.-W., and Seo, S.-W. (2017). Generation of a precise and efficient lane-level road map for intelligent vehicle systems. *IEEE Transactions on Vehicular Technology*, 66(6):4517–4533.

Hartley, R. and Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

Isern González, J. (2003). *Estudio experimental de métodos de calibración y autocalibración de cámaras*.

ITAM, R. (2019). Ek_autonomos_sim. `https://github.com/ITAM-Robotica/EK_AutoNOMOS_Sim`.

Jeong, J., Cho, Y., and Kim, A. (2017). Road-slam: Road marking based slam with lane-level accuracy. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1736–1473. IEEE.

Jo, K. and Sunwoo, M. (2014). Generation of a precise roadway map for autonomous cars. *IEEE Transactions on Intelligent Transportation Systems*, 15(3):925–937.

Joshi, A. and James, M. R. (2015). Generation of accurate lane-level maps from coarse prior maps and lidar. *IEEE Intelligent Transportation Systems Magazine*, 7(1):19–29.

Kaehler, A. and Bradski, G. (2016). *Learning OpenCV 3: computer vision in C++ with the OpenCV library*. .°'Reilly Media, Inc.".

Karouach, I. and Ivanov, S. (2016). Lane detection and following approach in self-driving miniature vehicles.

Krüger, T.-J., Rojas, R., and Göhring, D. (2015). Fahrspurmodellierung mit punktvalidierung bei autonomen modellfahrzeugen.

Küçükyildiz, G. and Ocak, H. (2014). Development and optimization of a dsp-based real-time lane detection algorithm on a mobile platform. *Turkish Journal of Electrical Engineering & Computer Sciences*, 22(6):1484–1500.

López, A., Serrat, J., Canero, C., Lumbreras, F., and Graf, T. (2010). Robust lane markings detection and road geometry computation. *International Journal of Automotive Technology*, 11(3):395–407.

Michavila, F. and Gavete, L. (1992). *Programación y cálculo numérico*. Reverté.

Pulido Fentanes, J. et al. (2012). Exploración y reconstrucción tridimensional de entornos mediante robots móviles.

Rebaza, J. V. (2007). Detección de bordes mediante el algoritmo de canny. *Escuela Académico Profesional di Informática. Universidad Nacional de Trujillo*.

Rehder, E. and Albrecht, A. (2015). Submap-based slam for road markings. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1393–1398. IEEE.

ROS (2018). http://www.ros.org/about-ros/. Último acceso:10 enero de 2018.

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

Sánchez, G. E. A. (2013). *Localización y mapeo simultáneos para robot de búsqueda en entornos de desastre*. PhD thesis, Universidad Nacional Autónoma de México.

Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71.

Thrun, S. (2002). Probabilistic robotics. *Communications of the ACM*, 45(3):52–57.

Thrun, S. et al. (2002). Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, 1:1–35.

Vanegas Sánchez, T. D. (2018). Navegación autónoma de un vehículo a escala por medio de redes neuronales profundas. Master's thesis, Centro de Investigación en Computación IPN.

Vivacqua, R. P. D., Bertozzi, M., Cerri, P., Martins, F. N., and Vassallo, R. F. (2018). Self-localization based on visual lane marking maps: An accurate low-cost approach for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 19(2):582–597.