ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

TEMA: "PERCEPCIÓN 3D Y PLANIFICACIÓN DE TRAYECTORIAS BASADO EN UNA COMBINACIÓN DE LIDAR 2D Y VISIÓN PARA VEHÍCULOS TERRESTRES NO TRIPULADOS"

AUTORES:

LIMAICO ORTEGA, ALEX DARÍO

SANDOVAL CÁRDENAS, DAVID SEBASTIÁN

DIRECTOR: DR. AGUILAR CASTILLO, WILBERT GEOVANNY

SANGOLQUÍ

2018

DEPARTAMENTO DE ELECTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

CERTIFICACIÓN

Certifico que el trabajo de titulación, ***"PERCEPCIÓN 3D Y PLANIFICACIÓN DE TRAYECTORIAS BASADO EN UNA COMBINACIÓN DE LIDAR 2D Y VISIÓN PARA VEHÍCULOS TERRESTRES NO TRIPULADOS"*** fue realizado por los señores ***LIMAICO ORTEGA, ALEX DARÍO*** y ***SANDOVAL CÁRDENAS, DAVID SEBASTIÁN***, el mismo que ha sido revisado en su totalidad, analizado por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustenten públicamente.

Sangolquí, 16 de agosto del 2018

Ing. Aguilar Castillo Wilbert Geovanny PhD.

CC: 0703844696

DEPARTAMENTO DE ELECTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORÍA DE RESPONSABILIDAD

Nosotros, ***LIMAICO ORTEGA, ALEX DARÍO*** y ***SANDOVAL CÁRDENAS, DAVID SEBASTIÁN***, declaramos que el contenido, idea y criterios del trabajo de titulación: ***"PERCEPCIÓN 3D Y PLANIFICACIÓN DE TRAYECTORIAS BASADO EN UNA COMBINACIÓN DE LIDAR 2D Y VISIÓN PARA VEHÍCULOS TERRESTRES NO TRIPULADOS"*** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Consecuentemente el contenido de la investigación mencionada es veraz.

Sangolquí, 16 de agosto del 2018

Alex Darío Limaico Ortega
CC. 1721413613

David Sebastián Sandoval Cárdenas
CC. 1716890445

DEPARTAMENTO DE ELECTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

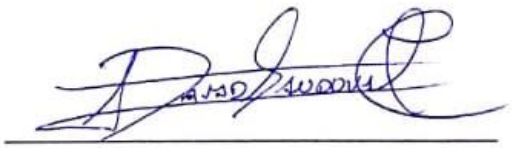
AUTORIZACIÓN

Nosotros, ***LIMAICO ORTEGA, ALEX DARÍO*** y ***SANDOVAL CÁRDENAS, DAVID SEBASTIÁN***, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: ***"PERCEPCIÓN 3D Y PLANIFICACIÓN DE TRAYECTORIAS BASADO EN UNA COMBINACIÓN DE LIDAR 2D Y VISIÓN PARA VEHÍCULOS TERRESTRES NO TRIPULADOS"*** en el repositorio institucional cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Sangolquí, 16 de agosto del 2018

Alex Darío Limaico Ortega

CC. 1721413613



David Sebastián Sandoval Cárdenas

CC. 1716890445

DEDICATORIA 1

Dedico este trabajo a mis queridos padres Consuelo y Fabián. Por haberme guiado con sus enseñanzas, por haberme educado con valores, por haberme inspirado el deseo de superación constante, pero sobre todo por ser un buen ejemplo. Me han mostrado el camino a seguir para convertirme en un hombre de bien, que mediante el esfuerzo y sacrificio aporte positivamente a la sociedad.

A mi hermana Micaela, por ser la fuente más pura de inspiración. Con su corta edad me recuerda las virtudes de soñar como un niño y de sonreírle a la vida como si de un juego se tratase. Te dedico todo mi trabajo y esfuerzo que me llevó hasta aquí. Siempre seré tu apoyo y espero que cada vez que leas estas palabras te motive a ser igual o mejor que yo.

A mis familiares quienes me acompañaron en mi proceso de formación personal y académica, porque siempre estuvieron disponibles y gustosos de brindarme un consejo para que pueda ser mejor cada día.

Finalmente, te lo dedico a ti, la persona que tuvo el interés de leer este trabajo. Espero que uses el conocimiento aquí plasmado para un bien constructivo que aporte al desarrollo del país y del mundo.

Alex Darío Limaico Ortega

DEDICATORIA 2

La gloria es de Dios porque no existe ecuación que describa mi existencia y que describa como a cada paso de mi carrera universitaria cayeron bendiciones; vida, salud, sabiduría, logros, talentos y personas únicas. A él le dedico este gran paso en mi vida.

A mi madre, por ser esa persona que confía en mí, que ama cada paso que doy y que me brinda ejemplo de trabajo, profesionalidad y constancia. Por fortalecer mi espíritu y guiarme por el camino correcto, llenándome de buenos valores y principios.

A mi abuela, por su consejo y visión de que mi sabiduría podría hacer que alcance logros personales, por creer en que puedo avanzar y superarme constantemente.

A mi familia por su apoyo incondicional, porque cada uno tiene una función especial en mi vida, formaron con cariño mi carácter y son mi mejor orgullo.

A novia, por ser mi soporte, porque me enseño que un camino posee caídas, pero se debe levantar y seguir, por ser un ejemplo de estudio, de organización y crecimiento constante, y por fortalecer mi espíritu, mi ánimo y mis deseos de alcanzar las metas.

Finalmente, dedico este escrito a cada persona que lo lea, el conocimiento es libre.

David Sebastián Sandoval Cárdenas

AGRADECIMIENTO 1

Agradezco primeramente a mis padres por todo el tiempo brindado y su preocupación en cada paso que doy, agradezco que hayan estado siempre a mi lado, que me hayan dado una excelente educación, que hayan creído en mi y en mis capacidades. Gracias a todo ello son quien soy hoy.

Al Doctor Wilbert Aguilar, quien me ha apoyado durante el desarrollo de este proyecto y también en mi desarrollo profesional. Por ser un excelente docente, ético y responsable. Agradezco que haya compartido conmigo su conocimiento y experiencia que me han permitido crecer como investigador y como persona.

Al Centro de Investigación Científica y Tecnológica del Ejército (CICTE), por la oportunidad de mostrar mis habilidades y trabajar en sus proyectos, incluyendo el presente trabajo.

A mi compañero de tesis, Sebastián Sandoval, por su apoyo en las aulas y en este trabajo. Por haberme demostrado ser un gran compañero de trabajo, pero sobre todo por ser un amigo leal. Agradezco también su entrega, siempre da lo mejor de sí mismo, conocerlo me permitió ganar experiencia y superarme cada día.

A mis compañeros y amigos, quienes siempre me regalaron una sonrisa cuando la necesitaba con quienes crecí y compartí varias de las mejores experiencias como estudiante y como persona.

Alex Darío Limaico Ortega

AGRADECIMIENTO 2

La incapacidad del ser humano es la capacidad de Dios y le agradezco porque a pesar de que mi carrera universitaria no ha sido fácil me ha puesto en el lugar indicado, con las personas correctas y en el tiempo adecuado. Permitiendo en mí alcanzar este logro académico.

La fe de alcanzar lo inimaginable llevo a mis padres sembrar en mí los recursos necesarios para ser un profesional. Les agradezco por estar pendientes, por velar que siempre este en el camino correcto, anímame a desarrollar mí potencial y porque con orgullo me brindan amor en cada paso de mi vida.

A mi compañero Alex Limaico, por su entrega en el desarrollo del presente trabajo y porque a lo largo de la carrera universitaria ha sido un ejemplo de capacidad, generosidad y empeño. Agradezco su lealtad e ingenio con lo cual hemos superado cualquier reto que tomemos, es un gran compañero con el cual puedo confiar.

Agradezco, a nuestro mentor, Dr. Wilbert Aguilar por visualizar en mí, logros, que no pensé realizarlos, ha sido un gran ejemplo profesional que motiva a que un estudiante alcance el reconocimiento que se merece, fomentando la amistad y el desarrollo personal.

A mi novia, por participar de mis alegrías y mis dificultades, apoyándome, dándome ánimo y generando paz en mí, permitiendo que las circunstancias sean más agradables. Agradezco por compartir la meta de ser profesionales con una persona que reinventa mis ideas.

A mis amigos y compañeros con los que compartí decepciones, alegrías, logros y experiencias durante la vida universitaria, grabando momentos inolvidables e irrepetibles.

Finalmente agradezco a cada persona que hizo posible este logro; a mi familia, a la familia de mi compañero, a los profesores y mentores universitarios, a los funcionarios de la institución, al Centro de Investigaciones del Ejercito y a todo aquel que velo por nosotros.

David Sebastián Sandoval Cárdenas

INDICE DE CONTENIDOS

INDICE DE TABLAS

INDICE DE FIGURAS

RESUMEN

En la robótica móvil se ha explorado muy poco la utilización de vehículos de uso cotidiano en la navegación autónoma. Además, la percepción del entorno para la navegación de vehículos no tripulados generalmente se lo realiza por medio de sensores 3D los cuales encarecen el desarrollo de estas plataformas. Una alternativa poco investigada es la combinación de un sensor laser de percepción horizontal como el LIDAR y un sensor vertical como las cámaras de estéreo visión, de esta manera tener una percepción tridimensional. En esta investigación se desarrolla una plataforma móvil que planifica trayectorias libres de colisión en un entorno desconocido. Inicialmente se caracterizan los sensores y el vehículo, obteniendo parámetros definidos. El desarrollo de la plataforma se realiza en un vehículo de configuración Ackerman por lo que se genera su modelo cinemático. La planificación de trayectorias se la realiza por medio del algoritmo de planificación denominado Non-Holonómic Risk RRT. Este algoritmo combina el uso de posicionamiento otorgado por la cámara y el mapeo otorgado por el sensor laser. Adicional se almacena la percepción tridimensional para convertir los entornos en conocidos. Se evalúa diferentes trayectorias de un punto inicial a un punto meta. Se generan entornos internos, externos, fijos y dinámicos. Se concluye con un sistema íntegro que responde adecuadamente en entornos controlados, el noventa por ciento de los casos llega a la meta, la percepción es de gran calidad y el sistema servirá para diversificar futuras aplicaciones de navegación autónoma.

PALABRAS CLAVE:

- **NAVEGACIÓN AUTÓNOMA**
- **VEHÍCULOS TERRESTRES NO TRIPULADOS**
- **PERCEPCIÓN TRIDIMENCIONAL**

ABSTRACT

In mobile robotics, the use of common vehicles in autonomous navigation has been explore less. In addition, the perception of the environment for navigation of unmanned vehicles usually has done by 3D sensors, which make the development of these platforms more expensive. A little researched alternative is the combination of a horizontal perception laser sensor such as the LIDAR and a vertical sensor such as the stereo vision cameras, in this way having a three-dimensional perception. This research develops a mobile platform that plans collision free trajectories in an unknown environment. Initially we characterize the sensors and the vehicle, defining the parameters. We develop the platform by an Ackerman configuration vehicle, and then we generate its kinematic model. The trajectory planning is carried out by means of a planning algorithm called Non-Holonomic Risk RRT. This algorithm combines the use of positioning granted by the camera and the mapping provided by the laser sensor. Additional three-dimensional perception is stored with the aim of converting environments into known. We evaluate different trajectories from a starting point to a goal. Internal, external, fixed and dynamic environments are generated. It concludes with an integral system that responds well in controlled environments, ninety percent of the cases reach the goal, the perception is of high quality and the system will serve to diversify future applications of autonomous navigation.

KEYWORDS:

- **AUTONOMOUS NAVIGATION**
- **UNMANNED GROUND VEHICLES**
- **THREE-DIMENSIONAL PERCEPTION**

CAPÍTULO I

1. INTRODUCCIÓN

1.1. Antecedentes

La percepción (Aguilar, y otros, Visual SLAM with a RGB-D Camera on a Quadrotor UAV Using on-Board Processing, 2017), (Aguilar, y otros, On-Board Visual SLAM on a UGV Using a RGB-D Camera, 2017), o identificación del entono 3D (Aguilar, y otros, Real-Time 3D Modeling with a RGB-D Camera and On-Board Processing, 2017) es una herramienta requerida en diversas aplicaciones como; sistemas de visión autónoma (Meers & Ward, 2004), mapeo en ambientes internos o externos (Wulf, Arras, Christensen, & Wagner, 2004), sistemas servo visual (Pino, 2009), reconocimiento de objetos (Klimentjew, Hendrich, & Zhang, 2010), navegación en robótica móvil (Gutmann, Fukuchi, & Fujita, 2012), ente otras (Andrea, Byron, Jorge, Inti, & Aguilar, 2018), (Jara-Olmedo, y otros, 2018), (Pardo, Aguilar, & Toulkeridis, 2017). En muchas de estas la percepción es paso previo que permite el desarrollo del mapeo instantáneo, y si de robótica móvil se trata es necesario el mapeo y localización simultanea SLAM con lo cual se llega a aplicaciones en tiempo real (Engelhard & Endres, 2011) o estimación 3D (Stryk, 2011).

Aplicaciones reales de la robótica móvil como la búsqueda y rescate requieren de la capacidad de mapear entornos desconocidos y de forma complementaria la planificación de trayectorias eficientes (Kohlbrecher & Stryk, 2011). En la literatura existen múltiples algoritmos para planificación de movimientos que trabajan con métodos diferentes cada uno enfocado en mejorar la calidad de la trayectoria. Por ejemplo estudios muestran comparativas entre algoritmos genéticos y algoritmos de optimización de enjambre de partículas en aplicaciones de tiempo real (Roberge V., 2012), con lo cual se concluye que es necesario considerar las propiedades

dinámicas del vehículo y la complejidad del entorno 3D. Otros algoritmos para planificación de movimientos en vehículos autónomos consideran; la variación angular entre vértices (Kim, 2014), áreas potencialmente peligrosas (Purcaru, 2013), trayectorias dinámicas (Duchoň, 2014) o redes neuronales (Verma, 2014), entre otras.

Los algoritmos que más destacan son Probabilistic Roadmap o PRM y Rapidly Exploring Random Tree o RRT. Este último permite la planificación de movimientos en un espacio conocido X, de un patrón continuo o trayectoria que describe la conexión entre un estado X_inicial y un X_meta, pertenecientes al espacio X (Valle, 1998). Existen diversas variaciones de esta herramienta como RRT* la cual toma en cuenta el tamaño de paso y la separación mínima entre vértices vecinos lo cual optimiza la generación del árbol (Islam, 2012). La variación depende de la dinámica propia de la plataforma móvil o robot en el cual se implementará, ya que si esta no es capaz de modificar su dirección instantáneamente será necesaria una modificación llamada SRRT la cual considera trayectorias curvas en el árbol (Moon, 2013).

Por otro lado, el PRM es un algoritmo que resuelve el mismo problema por medio de mapa probabilístico aleatorio que explora caminos factibles alrededor de una serie de obstáculos poligonales. La idea básica es tomar muestras aleatorias del espacio del vehículo, probar si están en el espacio libre, y utilizar un planificador local para conectar otras configuraciones cercanas. (James & Kuffner, 2000).

Los vehículos autónomos o robots autónomos son dispositivos que pueden operar con un elevado grado de autonomía, en el caso del campo de exploración y la planificación de trayectorias deben considerar entornos dinámicos y ofensivos (Trahanias, 2002). Zonas pobladas o llenas de obstáculos dificultan la navegación de robots autónomos considerando que el vehículo debe ser multiplataforma y debe adaptarse a las condiciones ambientales (Kümmerle, 2014). Al

colocar el vehículo en entornos heterogéneos es necesario una respuesta inmediata refrescando la ruta y redireccionando al vehículo, lo cual se logra realizando simulaciones en tiempo-real (Jaklin, 2013).

Una limitante en los robots móviles es el campo de visión y la optimización de recursos. Adicionalmente se debe contemplar una solución que conjugue eficiencia con resultados. Es posible experimentar la fusión de un sensor laser de alto rango, Laser Rangefinder (Unnikrishnan, 2005) y cámaras de estéreo visón (Kumar, 2010) para incrementar; rango de visión, detección de obstáculos (KUEN-HAN, CHUN-HUA, & ANDREAS, 2012) y estimación de profundidad (Saurav, 2010). Incentivando la generación de una trayectoria espacial que optimice el desempeño del panificador de movimientos (Baltzakis, Argyros, & Trahanias, 2003). La fusión de datos en múltiples sensores debe contemplar criterios tales como el filtro de Kalman (Sun & Deng, 2004), el cual provee precisión, tolerancia a fallos y robustez sin discriminar la naturaleza de los datos. También se deben considerar clasificadores que permitan identificar la correspondencia de puntos de interés y relaciones de imágenes (Hwang, Cho, & Ryu, 2007), (Zlot, 2009).

La aplicación de estas herramientas puede realizarse sobre un vehículo terrestre no tripulado o UGV (Baldwin, 2012), esto contempla que el vehículo tenga la capacidad de auto localizarse o edometría, tanto en aplicaciones de localización urbana (Chong & Qin, 2013) como escenarios mixtos que poseen situaciones controladas (Baig & Aycard, 2011).

Una de las técnicas de detección más comunes es el uso de la visión por ordenador. Esta puede proporcionar una gran cantidad de información sobre el entorno. Sin embargo, sufre variaciones de intensidad, textura, campos estrechos de la vista, e información de profundidad de baja precisión (M. Y. Kim, 2004). La detección del alcance del láser (LIDAR) es otra tecnología

atractiva debido a su alta precisión en el rango medición, su visión de área amplia y la baja requisitos (Ho, 1999) LIDAR estima distancias entre el sensor a bordo y una variedad de objetos detectados dependiendo del tiempo de vuelo de la luz láser. Sus mediciones dependen del tamaño y la reflectividad del objetivo, por lo que la probabilidad de detección disminuye con la distancia. Dado que sus características se complementan entre sí, es lógico utilizar tanto la visión, mejor aún la estéreo visión, en conjunto con el LIDAR para detectar objetos diferentes. Los sistemas de fusión de sensores se utilizan para combinar los datos sensoriales de fuentes dispares, de modo que el resultado será más preciso y completo en comparación con la salida de un sensor (Huang, 2009).

1.2. Justificación e Importancia

El desarrollo de la investigación científica y de la tecnología es pieza fundamental en aplicaciones de interés social. La robótica cognitiva coloca herramientas que simplifican tareas en la sociedad, y contribuyen en la accesibilidad y seguridad (Merrick, 2017). Diversas aplicaciones enfatizan el uso de la navegación autónoma en vehículos terrestres utilizados en entornos con presencia estocástica de personas (Liu, 2015). Por otro lado, en el campo militar los sistemas de navegación autónoma tienen una fuerte acogida en el desarrollo de investigación en líneas tales como: planeación táctica (Langley, 2005), defensa, seguridad y rescate (Bhat, 2015). La navegación autónoma da pie al futuro desarrollo de la robótica móvil a nivel social y comercial en el país.

El desarrollo del proyecto de investigación busca solventar la percepción del entorno 3D por medio del uso de sensores de bajo costo como son LIDAR 2D y cámara de estéreo visión para vehículos terrestres, así como la planificación de trayectorias de forma que vehículos de uso

cotidiano asciendan a vehículos de navegación autónoma. Adicionalmente se busca implementar el sistema de percepción y navegación en un UGV siendo esta una contribución directa al proyecto; “Sistemas de guiado para la navegación autónoma de vehículos terrestres en entornos GPS-denegados (VisualNavCar)”, y abrirá futuras líneas de investigación dentro de este campo. El estado del arte ubica a la percepción 3D como una opción de bajo coste computacional, y la combinación de sensores como un sistema integro que permite optimizar la planificación de trayectorias en tiempo real.

Resultado de varias investigaciones en el campo de la robótica móvil enfocadas en vehículos autónomos UGV han demostrado lo complejo que resulta la integración de la planificación de trayectorias en entonos 3D (Langley, 2005), por lo cual el trabajo planteado resulta en un aporte importante al estado del arte. El desarrollo de un sistema de percepción y navegación para UGV requiere de una implementación ordenada y sistemática, considerando todas las propiedades dinámicas del vehículo. El paso inicial recae sobre la correlación de los grafos resultantes de cada sensor, para su posterior interpretación, caracterización y planificación de trayectorias. Consecuentemente, se busca generar una base sólida en navegación de vehículos autónomos para su implementación y optimización.

1.3. Alcance del Proyecto

Mediante el presente proyecto, se plantea el desarrollo y optimización de algoritmos de percepción 3D y planificación de trayectorias usando un sensor LIDAR 2D combinado con una cámara de estéreo visión, implementado sobre un vehículo terrestre pequeño a batería.

Inicialmente se realiza la caracterización de los sistemas, partiendo por la calibración del sensor LIDAR, esto debido a que usualmente un sensor proporciona mediciones en su propio

sistema de coordenadas. La calibración permite la transformación de un punto del sistema de coordenadas del escáner láser al plano de la imagen aplicando matrices de rotación y traslación (Dietmayer, 2008). Luego se realiza la calibración de la cámara para que pueda trabajar en función del sensor, con la mayor precisión y para evitar errores de alineación.

Partiendo del antecedente de la odometría definida por el sensor se realiza la construcción de un mapa 2D para ello se usará localización y mapeo simultáneo (SLAM). Este mapa sirve como base para facilitar la odometría o pose actual del robot, eso gracias al uso del posicionamiento de la cámara de estéreo visión a cada paso de tiempo.

Se realizan aproximaciones geométricas de las escenas en las que se incluye la superficie 2D. Para utilizar imágenes y datos de rango simultáneamente, es necesario transformar los datos en un sistema de coordenadas común. (Yunsu Bok, 2007). Por lo tanto, se debe calcular la posición relativa entre los sensores.

Posterior se identifica los puntos de profundidad en las imágenes para correlacionar la profundidad percibida con el mapa 2D. Todo esto se conjuga en un algoritmo de percepción 3D que represente el entorno de forma similar a Google Street View (Anguelov, 2010). Esta percepción 3D se genera conforme el vehículo desarrolle la trayectoria.

Parte de las pruebas inician al acondicionar el vehículo terrestre, analizando y definiendo las restricciones dinámicas y físicas. La posición correcta de los sensores debe permitir verificar la correlación geométrica y que la percepción se construya uniforme y sin solapamientos. Esta prueba se llevará a cabo por teleoperación del vehículo en un entorno controlado de prueba. Debido a restricciones en los dispositivos y para evitar efectos externos, inicialmente el entorno será un entorno interno cerrado tipo coliseo, con obstáculos fijos y dinámicos.

Consecuentemente se desarrollará un algoritmo de planificación de trayectorias el cual parte del desempeño y mejoras realizadas con anterioridad al algoritmo RRT. Debido a que se pretende la navegación en entornos desconocidos con obstáculos fijos y dinámicos, es necesario analizar las particularidades probabilísticas de PRM y realizar pruebas para un algoritmo de planificación dinámica. Este algoritmo deberá refrescar la trayectoria conforme los objetos externos. Las simulaciones desarrolladas en Ubuntu permiten tener una base de funcionamiento previo a la implementación.

Luego se implementa el planificador. Esto implica varias consideraciones y modificaciones, lo primero es que la planificación se realice en tiempo real en función a las restricciones dinámicas del vehículo. Además, se debe contemplar una correlación directa entre el mapa generado y la planificación, así las áreas inexploradas se consideran parcialmente en la planificación. Por último, de la percepción 3D relacionar la planificación de trayectorias con los relieves restrictivos definidos como cambios de relieve o entornos que comprometan la integridad física del vehículo.

Finalmente se realizan pruebas de navegación autónoma en el entorno controlado con el objetivo de alcanzar una meta asignada. Para ello se analizará el desempeño en cuatro escenas; Obstáculos Fijos - Relieves Uniformes, Obstáculos Dinámicos - Relieves Uniformes, Obstáculos Fijos - Relieves Restrictivos y Obstáculos Dinámicos - Relieves Restrictivos. Para con ello presentar los resultados, concusiones y recomendaciones para trabajos futuros.

1.4. Objetivos

1.4.1. Objetivo General

- Integrar los sensores LIDAR 2D y cámara de estéreo visión para la percepción 3D y la navegación autónoma de un vehículo terrestre no tripulado.

1.4.2. Objetivos Específicos

- Generar un mapa 2D a partir del sensor laser LIDAR que permita tener los datos del entorno usando la técnica de SLAM.
- Obtener las aproximaciones geométricas y los puntos de profundidad del entorno por medio de imágenes estereoscópicas para generar el algoritmo de percepción 3D.
- Acondicionar hardware y software del vehículo terrestre, y verificar la correlación geométrica y que la percepción se construya uniforme por medio de tele operación.
- Desarrollar el algoritmo de planificación de trayectorias basados en RRT para un vehículo terrestre ubicado en un entorno interno desconocido.
- Implementar el algoritmo de planificación de trayectorias tomando en consideración tiempo real, áreas inexploradas y relieves restrictivos.
- Evaluar los resultados obtenidos de la implementación de los algoritmos para la navegación autónoma en entornos fijos y dinámicos.

CAPÍTULO II

2. MARCO CONCEPTUAL

2.1. Introducción

Este capítulo describe los conceptos necesarios para la presente investigación, así como también el estado del arte de las principales temáticas tratadas. Se define un marco de conceptos, ecuaciones y algoritmos necesarios para el desarrollo del trabajo. Se considera a este capítulo como la base que permitirá desarrollar un sistema de percepción 3D y planificación de trayectorias para vehículos terrestres no tripulados (UGV's).

De forma inicial se describe a los UGV, definiendo sus principios básicos, características, morfología, clasificación, modelamiento y aplicaciones. Se presenta los sistemas de percepción y mapeo en dos y tres dimensiones. Por lo tanto, se conceptualiza el estado del arte referente a los localizadores laser, con su funcionamiento, clasificación, estructura, ventajas y desventajas, en función de la aplicación y el uso que se les otorgue. En consecuencia, se describen los métodos y algoritmos para la percepción de profundidad y su posterior generación de mapas en dos dimensiones.

Posteriormente se describen los sistemas de percepción en tres dimensiones, enfocados específicamente en detallar los sistemas basados en visión estereoscópica. Se describen los algoritmos para percepción de profundidad, reconstrucción e identificación de entornos. Posterior se presentan los avances más recientes y trabajos en desarrollo que involucren a los sistemas basados en LIDAR y visión estereoscópica, para con ello la propuesta de la fusión de ambos sensores.

El capítulo finalizará con conceptualización de la planificación de trayectorias para un UGV, tanto como para planificadores locales como planificadores globales. En este último caso se realiza un resumen de los principales algoritmos para la planificación de trayectorias enfocados en algoritmos de planificación de ruta basados en muestreo, como Probabilistic Road Maps (PRM) y Rapidly-exploring Random Trees (RRT).

2.2. Vehículo Terrestre no Tripulado (UGV)

En el campo de investigación de la robótica móvil se encuentran los vehículos terrestres no tripulados o por sus siglas en ingles UGV. De forma general un UGV es todo vehículo que se desplaza a través de la superficie del suelo y no transporta consigo una persona (Kanika & P, 2016). Es por ello que de forma general los vehículos terrestres no tripulados se clasifican en:

- UGV Tele operado
- UGV Autónomo

Para definir de forma específica el vehículo hay que discernir las características específicas de un sistema UGV en particular. Las principales características para que cada sistema determine su taxonomía acorde a sus necesidades son (Gage, 1995):

- Propósito (Misión específica)
- Perfil de la aplicación (Ej. Ambientes peligrosos o fuertes, Limitación de tamaño)
- Funcionalidad, rendimiento o costo.
- Locación (Ej.: interior, exterior, tipo de terreno)
- Modo de locomoción (Ej.: ruedas, pistas de deslizamiento o piernas)
- Técnicas de navegación y control

Estos vehículos son también identificados como vehículos no tripulados diseñados para ambientes peligrosos, o desafíos complicados para una persona. Es así que se los cataloga en aplicaciones militares. Instituciones como la NASA o DARPA han colaborado con el Ejercito para el desarrollo de UGV's para navegación en todo terreno, búsqueda y rescate, transporte de minas o explosivos, entre otras (Board & Council, 2005).

2.2.1. UGV Tele operado

Aplicaciones definidas como peligrosas para una persona son el campo de uso predominante para este tipo de vehículos, ya que pueden resolver tareas de forma remota sin comprometer la seguridad de las personas (Chakraborty & Basu, 2010). El operador proporciona todos los procesos cognitivos y controla a distancia en función de una realimentación sensorial que por lo general resulta del uso de visión remota (Sathiyanarayanan, 2011).

Los componentes del sistema tele operado son (Correa A. C., 2005):

- Estación Remota. - Compuesta por computador, periféricos de entrada como controles y periféricos de salida como monitores y parlantes.
- UGV Esclavo. - Debe de tener un sistema de comunicación con la estación remota, generalmente equipado con brazos manipuladores o herramientas.

Un claro ejemplo de UGV teleoperado es el Andros Wolverine del fabricante Remotec que se muestra en la Figura ***1*** el cual se lo puede controlar por medio de un enlace de microondas.

Figura 1. UGV Teleoperado - Andros Wolverine
Fuente: (Correa A. C., 2005)

2.2.2. UGV Autónomo

En esencia un UGV autónomo es un robot móvil que realiza locomoción sobre la superficie de la tierra sin necesidad de un operario humano. Un UGV completamente autónomo debe poseer las siguientes habilidades (Sathiyanarayanan, 2011).

- Obtener información del entorno
- Trabajar por largos periodos de tiempo sin intervención humana.
- Desplazarse de un punto A hasta un punto B sin navegación humana.
- Evitar situaciones dañinas a menos de que ese sea su objetivo.
- Se repare a si mismo sin ayuda extra.
- Detectar objetos de interés como personas u otros vehículos.
- Aprenda u obtenga nuevas capacidades sin ayuda extra.
- Ajuste sus estrategias y se adapte a las condiciones del ambiente.

Para alcanzar estas prestaciones uno de los factores determinantes es el uso combinado de sensores. El uso de sensores basados en imágenes como cámaras monoculares o estereoscópicas (mono y color) y dispositivos de detección de rango como LIDAR o RADAR permiten que la

navegación se de en entornos desconocidos (Luettel & Wuensche, 2012). Un UGV Autónomo debe de poseer los siguientes componentes:

- Computador a bordo. - Gestiona dispositivos e implementa algoritmos para su autonomía.
- Sensores. - Tales como localizador laser, cámara, GPS y brújula.
- Vehiculó modificado. - Estas modificaciones incluyen control de aceleración y giro.

Una habilidad adicional como se muestra en (Universidad Carlos III de Madrid, 2016) es la de coordinación y cooperación con otros vehículos autónomos. En la (Figura 2) se muestra el vehículo iCab el cual es un vehículo de golf modificado para navegación autónoma en un campus universitario. Una particularidad de aquel vehículo es que identifica obstáculos específicos como árboles y obstáculos dinámicos dentro de la generación de un camino libre de colisión.

Figura 2. Vehiculó terrestre no tripulado autónomo iCab 2
Fuente: (Universidad Carlos III de Madrid, 2016)

2.2.3. Sistemas de Locomoción

Los UGV's se desplazan sobre una superficie terrestre, que no para todos los casos resulta ser una superficie regular, es por tanto que allí se contempla la versatilidad del vehículo acorde al terreno sea; plano, rugoso, con obstáculos o con cuesta. Esta característica relaciona la

complejidad del vehículo, es decir el número de actuadores y las cadenas cinemáticas (Nie, Corcho, & Spenko, 2013).

Ciertamente los sistemas de locomoción se identifican según sus características por lo tanto pueden poseer patas, ruedas, cadenas o un hibrido. En la (Figura 3) se muestran algunos de estos sistemas. Los robots con ruedas poseen una destacada versatilidad en superficies planas, en las cuales para alcanzar una posición específica o destino específico debe desplazarse en sin mayor complejidad.

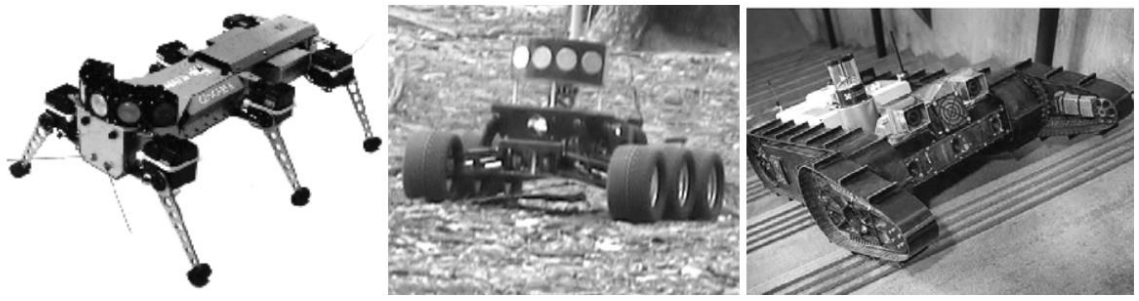


Figura 3. Sistemas de locomoción - Patas (Hermes), Ruedas (Journey), Cadenas (Scour)
Fuente: (Bambino, 2008)

2.2.4. Modelos de Locomoción

Considerando un sistema de locomoción por ruedas se entiende que dependiendo de la ubicación de sus ruedas y del tipo de ruedas que posea. De forma convencional existen tres tipos de ruedas (Bambino, 2008). La rueda fija es aquella que esta fija a la estructura y no cambia su orientación a menos de que el vehículo lo haga. La rueda orientada es la que posee rotación en su eje, agregándole mayor grado de libertad al vehículo. La rueda orientable no centrada o conocida como rueda loca y va pegada al eje vertical del vehículo. En la (Figura 4) se muestran los tipos de rueda, cabe señalar que adicional existe el tipo de rueda sueca la cual permite movimiento omnidireccional.

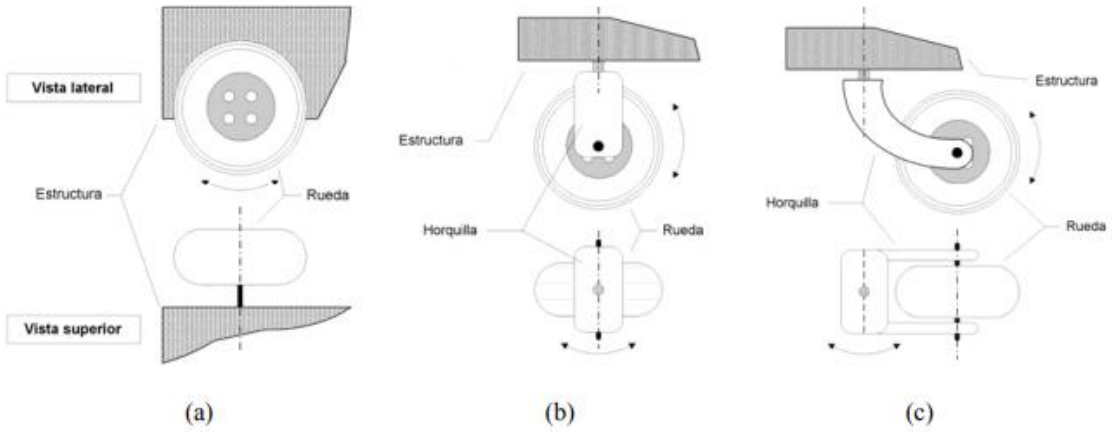


Figura 4.Tipos de Ruedas - Fija, orientable, rueda loca
Fuente: (Bambino, 2008)

La disposición de ruedas, patas u orugas permite alcanzar un sinnúmero de posibilidades, dependiendo del tipo, cantidad y combinación de las mismas. Esto genera modelos de locomoción específicos con una cinemática particular. Un concepto fundamental previo a la definición de los modelos el centro instantáneo de curvatura (CIC) o cetro instantáneo de rotación (CIR) el cual es la intersección de las normales a las trayectorias de dos puntos cualesquiera, que para el caso de las ruedas es perpendicular a su eje instantáneo (Gómez Cardenas, 2007). Los modelos de locomoción principales son:

2.2.4.1. Locomoción Diferencial

Se constituye de dos ruedas motrices fijas (que generan movimiento), idénticas y en paralelo. Cada una con un motor independiente, que permiten el cambio de dirección, y con ruedas en forma de esfera que le adicionan estabilidad y apoyo (Lens, 2015). En la Figura 5 se muestra la estructura de un vehículo con locomoción diferencial en el cual el CIR se encuentra perpendicular y fuera en dirección al sentido de giro.

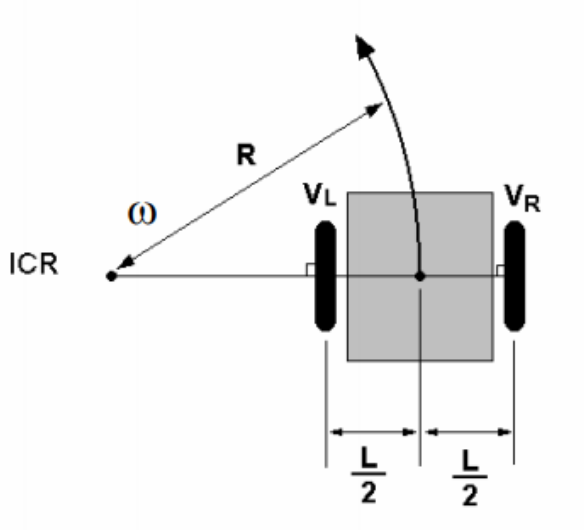


Figura 5. Estructura de Locomoción Diferencial
Fuente: (López Valverde, 2009)

2.2.4.2. Locomoción Sincrona

Se constituye por tres o cuatro ruedas motrices y directrices, las ruedas giran en la misma dirección y velocidad por acción de las poleas, el chasis conserva su posición (López Valverde, 2009). En la (Figura 6) se aprecia la estructura de este tipo de locomoción.

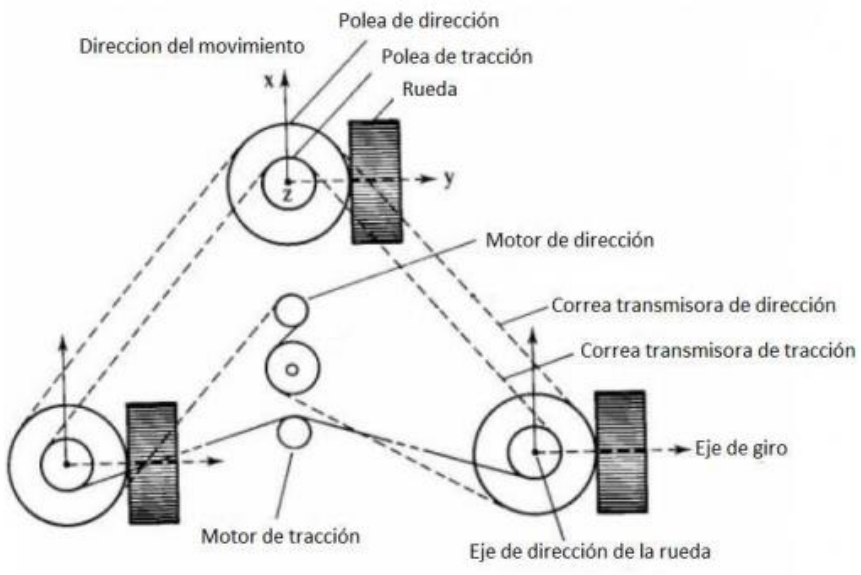


Figura 6. Estructura de Locomoción síncrona
Fuente: (López Valverde, 2009)

2.2.4.3. Locomoción Triciclo clásico

Esta locomoción se constituye por tres ruedas, una rueda delantera como directriz y las ruedas traseras como motrices. El CIR se localiza perpendicular al eje de la rueda directriz. En la (Figura 7) se detalla la estructura. Esta configuración tiene problemas de estabilidad en terrenos complicados ya que tiene pocos puntos de apoyo (R, Zhou, Adams, & Bodenheimer, 2003). Existen variaciones en las que las funciones de las ruedas cambian y la cinemática responde a esos cambios.

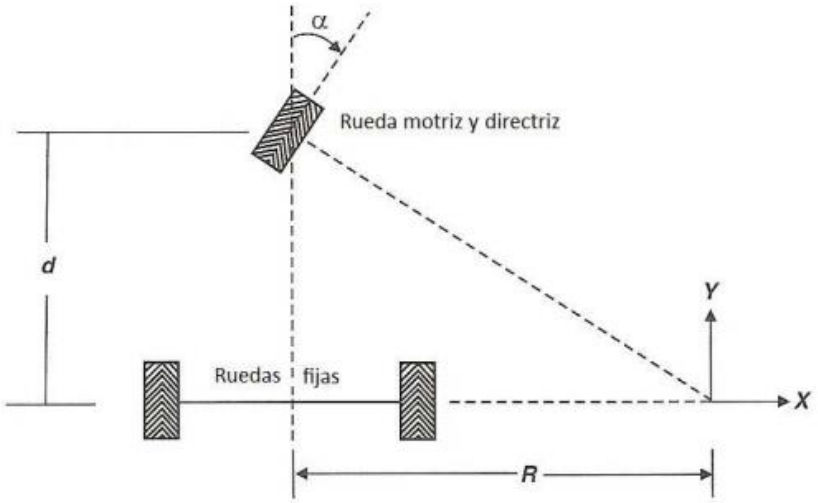


Figura 7. Locomoción en triciclo clásico
Fuente: (López Valverde, 2009)

2.2.4.4. Locomoción Ackerman

Es la popular configuración utilizada en la mayoría de los vehículos esta locomoción presentada en la Figura 8 Se compone de dos ruedas posteriores motrices y dos ruedas delanteras directrices. El problema principal en esta locomoción son las restricciones no holonómicas que posee, esto quiere decir que el vehículo no puede moverse de forma instantánea hacia los lados, el sistema permite que las ruedas rueden y giren (Murray, Li, Sastry, & Sastry, 1994), pero no resbalen por lo que posee puntos inalcanzables a simple dirección.

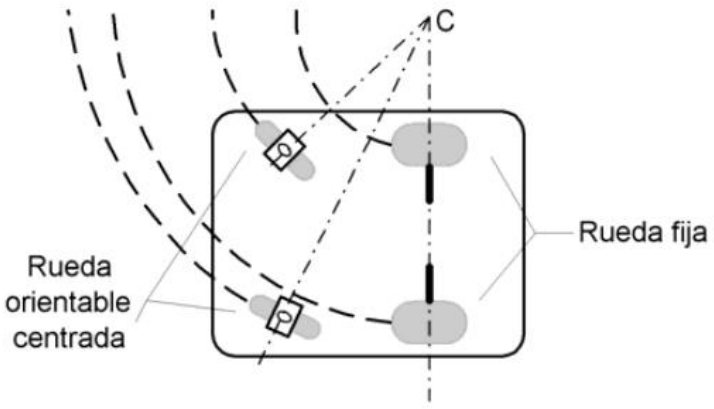


Figura 8. Locomoción Ackerman
Fuente: (Bambino, 2008)

El mecanismo de las ruedas frontales permite que la rueda interior a una curva posee un ángulo de giro mayor. Las ruedas poseen un mismo centro de rotación por lo que resultan cinéticamente equivalentes a una sola rueda virtual ubicada en el centro adquiriendo propiedades similares a la locomoción triciclo (Lens, 2015).

2.2.5. Cinemática de un UGV Ackerman

La cinemática de un UGV describe la variación de la posición respecto a las variables de actuación, sin considerar las causas físicas (Gómez Cardenas, 2007). Considerando el modelo de la Figura 9 y la conceptualización descrita en (Lens, 2015) la trayectoria seguida por el vehículo está caracterizada con la velocidad angular ψ_S y con el ángulo directriz $-\frac{\pi}{2} \leq \theta_S \leq \frac{\pi}{2}$. Como se definió anteriormente para un ángulo $\theta_S \neq 0$ el CIR se encuentra a una distancia radial r del eje horizontal. Este radio se determina por medio de la (Ecuación 1)

$$r = e' \cot \theta_S \quad (1)$$

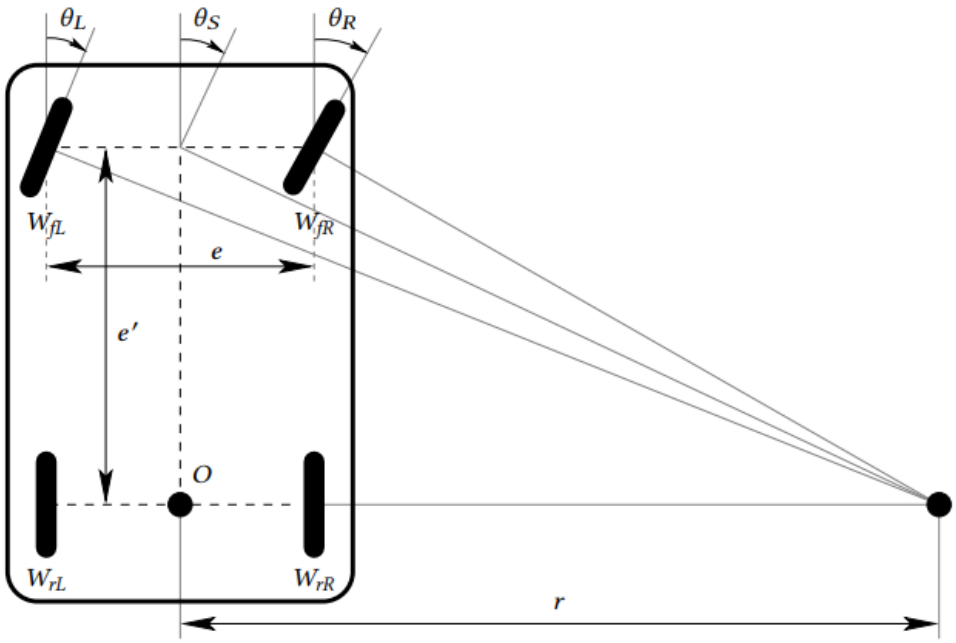


Figura 9. Configuración y parámetros para cinemática Ackerman
Fuente: (Lens, 2015)

Las ecuaciones son para un circulo de radio r son las siguientes

$$\tan\theta_S = \frac{e'}{r} \tag{2}$$

$$\tan\theta_L = \frac{e'}{r-\frac{e}{2}} \tag{3}$$

$$\tan\theta_R = \frac{e'}{r+\frac{e}{2}} \tag{4}$$

Donde se definen diferentes círculos dependiendo de la llanta, para ciertos casos de vehículos de cuatro llantas con mecanismos de giros a partir de transformaciones mecánicas pueden producir nuevas inclinaciones o a desplazamientos a lo largo de la vertical, por lo que se definen diferentes radios de giro.

2.3. Percepción Visual

Para una persona la percepción visual es la capacidad para interpretar las características del entorno y por efecto de la luz visible (efecto óptico) que llega a los ojos (Jeffrey, et al., 2011). En consecuencia, para un ser humano la percepción visual en tres dimensiones es respuesta a un proceso cognitivo al poseer dos ojos, los mismos que captan perspectivas solapadas. Esto da lugar al concepto de paralelaje binocular que como muestra en la (Figura 10) lo cual posibilita la percepción o visión tridimensional y la estereopsis visual (Reconstrucción 3D).

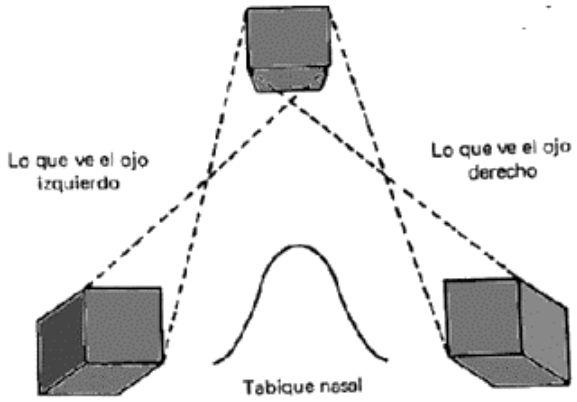


Figura 10. Paralelaje binocular humano
Fuente: (Casanova)

En el campo de la robótica móvil es necesario que los robots perciban lo que ocurre a su alrededor y es necesario dotar a las máquinas de un tipo de percepción artificial (Ruiz, 2015), lo suficientemente capaz para que el robot navegue de forma autónoma. Dar seguimiento al recorrido que realice un vehículo autónomo es esencial para preservar la integridad del vehículo. Los sistemas anticolisión poseen altas demandas de percepción del medio ambiente, ya que las situaciones de tráfico pueden ser extremadamente complejas (Gehrig & Stein, 2007). En el caso de la navegación autónoma cualquier falsa interpretación puede recaer en accidente, por ello es necesario el uso de visión artificial para la reconstrucción visual.

2.3.1. Visión artificial

La visión artificial o visión por computador es una disciplina de la robótica dedicada a interpretar, entender y reconstruir escenas tridimensionales partiendo de imágenes bidimensionales. El objetivo de la visión artificial resulta en modelar la visión humana a diferentes niveles, siendo capaz de tareas como: reconstrucción de escenas (Aguilar, y otros, Pedestrian Detection for UAVs Using Cascade Classifiers and Saliency Maps, 2017), detección de eventos (Aguilar, Cobeña, Rodriguez, Salcedo, & Collaguazo, 2018) (Aguilar, y otros, Real-Time Detection and Simulation of Abnormal Crowd Behavior, 2017), estimación de movimiento (Aguilar, y otros, Cascade Classifiers and Saliency Maps Based People Detection, 2017) (Lakhwani, Shah, Vaghela, Panchal, & Rathod, 2018), entre otras.

El proceso de la proyección resultante al plasmar un ambiente tridimensional en una imagen bidimensional no es un proceso reversible. La profundidad del ambiente tridimensional se torna en puntos fijos (Sucar & Gómez, 2011). Además, se genera el principio de que de diferentes escenas puede resultar la misma imagen. No obstante, existen alternativas que permiten recrear la tercera dimensión que se ha perdido. Una de aquellas es el uso de visión estereoscópica, utilización de dos imágenes, otra opción es el uso de las propiedades de la imagen como son la textura, sombreado, y así un estimado de la profundidad relativa (gradiente) (Sucar & Gómez, 2011).

2.3.2. Visión Monocular

Las imágenes bidimensionales muestran algunas características clave como; variaciones de textura, degradación, desenfoque, estabilización (Aguilar & Angulo, Robust video stabilization based on motion intention for low-cost micro aerial vehicles, 2014) o color (S. Guttman & R,

2007). Estas características son aprovechadas por medio de algoritmos de aprendizaje (Aguilar & Angulo, Real-time video stabilization without phantom movements for micro aerial vehicles, 2014). Uno de ellos son los Campos aleatorios de Markov (MRF) (Saxena, Schulte, & Ng, 2007), los cuales interpretan características en imágenes monoculares y las incorpora en sistemas estereoscópicos. La geometría de una cámara monocular vista en la (Figura 11), es uno de los modelos que se puede considerar en las cámaras monoculares. Para lo cual una escena es proyectada de forma ortográfica resultando en puntos que dependen directamente con la distancia focal y distancia a la imagen.

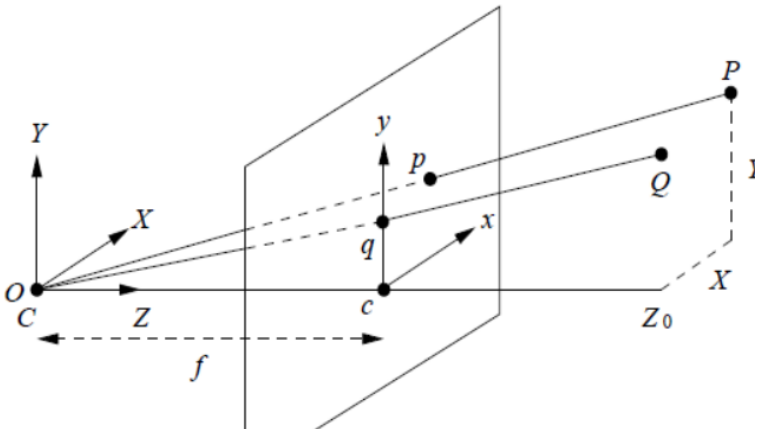


Figura 11. Modelo de una cámara por medio de proyección ortográfica escalada
Fuente: (Kheng, 2012)

Las formulaciones correspondientes para todos los puntos en la escena son:

$$x = sX \text{ , } y = sY \text{ , } s = \frac{f}{Z_0} \tag{5}$$

Otros modelos de cámaras monoculares son; modelo Pinhole (Matematikcentrum Lundus, 2015), modelo ortográfico paralelo, modelo de Gauss, entre otros (De Mata, 2016). No solo el modelo de la cámara permite estimar la profundidad, asignar o generar modelos a carreteras también optimiza el proceso (Joglekar, Joshi, Khemani, Nair, & Sahare, 2011). Una ventaja que

incurre este tipo de visión es que se puede reducir costos, al no necesitar más que una sola cámara (Akhlaq, Izhar, & Shahbaz, 2014).

2.3.3. Visión Estereoscópica

El método de visión estereoscópica es de gran importancia, debido a su analogía con la visión de las personas y otros seres vivos. Este método se coloca en los antes mencionados métodos pasivos, y consiste en el uso de dos o más cámaras separadas cierta distancia conocida para generar imágenes desplazadas de la misma escena en perspectivas diferentes como se muestra en la Figura 12 b). En ella se tiene que la separación relativa entre triángulos es menor que entre estrellas. Este fenómeno se debe a que la estrella se encuentra más cercana a los ojos como se muestra en la Figura 12 a). Aquella separación relativa entre objetos es denominada como disparidad.

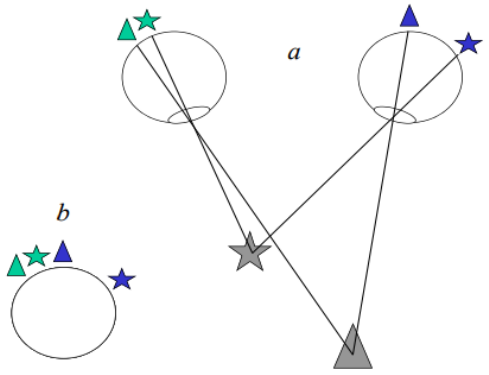


Figura 12. (a) Sistema biológico de visión estereoscópica,
(b) Superposición de imágenes de ambos ojos.
Fuente: (Hernández, Sanz, & Guijarro, 2011)

En la (Figura 13) se muestra dos imágenes estereoscópicas, captadas mediante un sistema artificial de dos cámaras alineadas horizontalmente, de tal manera que los objetos presentan desplazamiento horizontal.

La visión estereoscópica artificial posee la ventaja de obtener imágenes separadas siguiendo dos métodos:

- Dos o más cámaras alineadas y separadas una distancia.
- Una cámara móvil capaz de capturar imágenes mientras se desplaza en línea recta.

Para los dos casos las cámaras pueden tener sus ejes ópticos convergentes o paralelos, siento este último el modelo más utilizado.

Figura 13. (a) Imagen original estereoscópica izquierda;
(b) Imagen original estereoscópica derecha

El método del análisis de visión estéreo y reconstrucción tridimensional consiste en los siguientes pasos (Chen & Medioni, 1992):

- Adquisición de imagen
- Modelamiento de la cámara (geometría)
- Extracción de características
- Correspondencia de características
- Determinación de la distancia

2.3.3.1. Adquisición de imágenes

Las imágenes pueden ser adquiridas de diversas formas, tomando en cuenta las características mostradas en la (Tabla 1).

Tabla 1
Características en la adquisición de imágenes

Tiempo	Simultáneas	Δt pequeño	Δt grande
Posición	Igual	Ligeramente Distinta	Radicalmente Distinta
Dirección			
Condiciones Atmosféricas	N/A	N/A	Cualquiera que cambie la escena.

Resulta importante considerar la aplicación ya que por ejemplo imágenes tomadas con un vehículo aéreo de cartografía, son diferentes a las de un vehículo terrestre autónomo. En este último se requiere mayor precisión para la determinación de obstáculos.

Existen algunos sistemas de estéreo visión de forma comercial, los mismos que proveen de información directa al usuario final.

Tabla 2
Sistemas de estéreo visión comerciales

Fabricante	**Modelo**	**Res. (Mpx)**	**Rango (m)**	**Framerate (fps)**	**Linea Base (mm)**
PointGrey	Bumblebee2	0.8	-	20	120
IDS	Ensenso N35	1.3	0-3	10	120
Stereolabs	ZED	5.5	0-15	15	120
Intel	Realsense R200	0.17	3.5-10	60	70
Mobile Robots	MobileRanger	0.36	-	30	60
Videre Design	MEGA-DCS	1.3	-	-	80-240
e-Con Systems	Capella	0.36	-	-	100
LMI	Gocator 3100	1.3	0.25	5	-
Ricoh	SV-M-S1	1.3	0.8-1.2	30	200

Fuente: (Veitch-Michaelis, 2017)

Como aspectos adicionales se puede destacar que la Bumblebee2 posee una SDK que permite usar el algoritmo de emparejamiento local (SAD), el cual calcula el costo de una ventana alrededor de un pixel (Kanade & Okutomi, 1994). La cámara ZED utiliza un algoritmo de emparejamiento por características el cual requiere de una tarjeta Nvidia con CUDA habilitado. La MobileRanger también usa (SAD) pero a bordo de una tarjeta FPGA. Estos dispositivos permiten obtener imágenes estereoscópicas y sus características.

2.3.3.2. Modelo geométrico de la cámara

Los atributos geométricos y físicos de las cámaras definen el modelo o modelado. Usualmente se usa un modelo de Pinhole (Agujero de alfiler) con correcciones de distorsiones radiales y tangenciales de los lentes (Hartley & Zisserman, 2003). Este modelo describe una cámara estéreo ideal en la cual una imagen atraviesa una abertura proyectando una imagen invertida de forma directa. En la (Figura 14) se describe este modelo de la siguiente manera. P_I y P_D son los puntos de intersección con el plano de la imagen que representan al punto tridimensional $P(x, y, z)$. O_I y O_D son las proyecciones directas de los centros de las cámaras. La línea que separa los centros de las cámaras se llama línea base y la distancia focal es la distancia desde las proyecciones hasta los puntos de intersección.

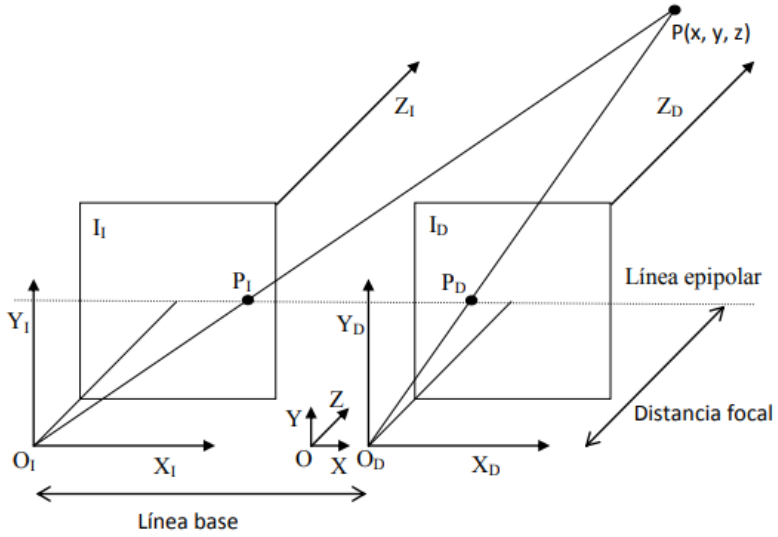


Figura 14. Modelo de pinhole para un sistema estéreo
Fuente: (Martínez, 2010)

Si las cámaras se encuentran alineadas se cumplen las siguientes relaciones:

$$x = \frac{x_{PI} * B}{x_{PI} - x_{PD}} \tag{6}$$

$$y = \frac{y_{PI} * B}{x_{PI} - x_{PD}} \tag{7}$$

Donde B es la línea base, x_{PI} : y_{PI} son coordenadas de P_I y x_{PD} es coordenada de P_D. Además, la diferencia (x_{PI} - y_{PI}) se denomina disparidad entre imágenes, y es inversamente proporcional a la profundidad z (Navas Flores, Suasnavas, & Ramiro, 2018).

Uno de los principales conceptos es el espacio epipolar, el cual está definido como el plano generado entre los puntos de intersección y cualquier punto tridimensional. La intersección del plano epipolar con el plano de la imagen es llamada línea epipolar. Todos los puntos a los que les corresponde una misma línea epipolar derecha deben poseer sus proyecciones derechas en la misma línea epipolar como se muestra en la Figura 15. No todas las líneas poseen proyección, y solo las que poseen definen el espacio de restricción epipolar.

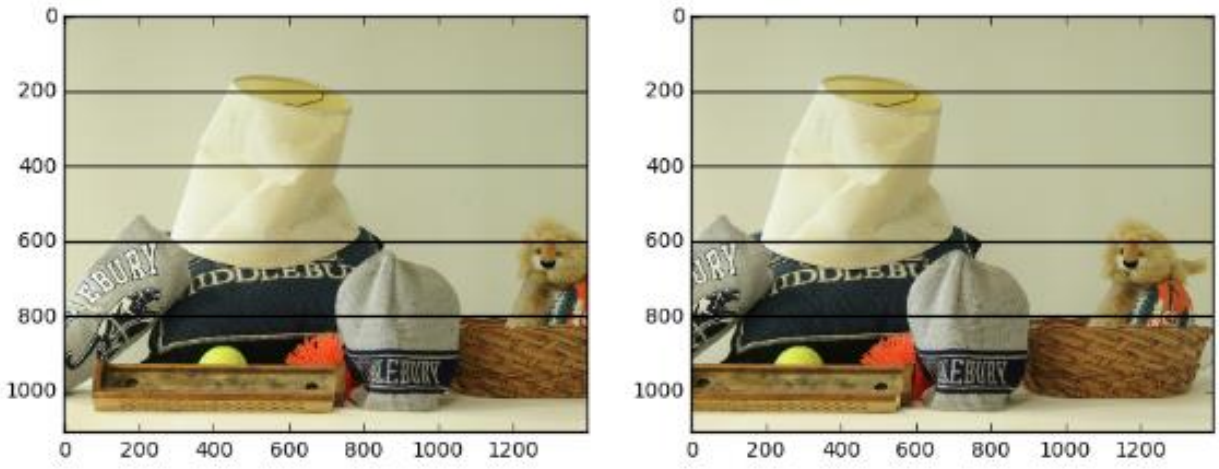


Figura 15. Par estéreo con sus correspondientes líneas epipolares
Fuente: (Hartley & Zisserman, 2003)

No siempre se forman pares de proyecciones alineadas horizontalmente, esto principalmente a errores tales como; por proceso de adquisición, deformación en la imagen, distorsión radial y tangencial. Es así que se requiere de un proceso de calibración para corregir las anomalías (Pajares & Cruz, 2007). La calibración es un proceso para obtener los parámetros de una cámara, para este tipo de cámaras existe la calibración coplanar la cual se la realiza por medio de una rejilla. Por otro lado, el no coplanar utiliza otros objetos para su calibración como por ejemplo un cubo.

2.3.3.3. Extracción de características

Este paso se identifican elementos propios de la imagen. Este se encuentra ligado al paso consecuente de correspondencia de características. Por lo tanto, es necesario definir las técnicas de obtención de características. Una de las técnicas se denomina técnica basada en área, la cual busca correlacionar un pixel con vecinos locales o vecinos correspondientes a la otra imagen par. La otra técnica se denomina técnica basada en características con lo que se identifica características que permitan identificar rasgos en representaciones basadas en bordes aislados, cadenas de puntos o áreas limitadas por bordes (Martínez, 2010).

Como premisa pertinente cabe detallar que los puntos de borde son importantes en cualquier sistema de visión estéreo. Otra característica importante resultan ser las regiones asociadas a escenas y delimitadas por bordes. Para la detección de bordes y áreas existen métodos tales como; gradiente morfológico, operador de Sobel, Deriche entre otros (Gonzalez, Melin, Castro, & Castillo, 2017). Una vez que se tenga las características es posible que se necesite realizar un paso de segmentación adicional, en el cual se obtiene atributos o propiedades más específicas. Es así que se asocian las características, como por ejemplo nivel medio de intensidad, textura o elongación.

2.3.3.4. Correspondencia de características

Los atributos de las características son almacenados en vectores, es así que es necesario establecer correspondencia local, la cual se basa en el grado de similitud. Para ello es necesario métricas como distancia; Euclidiana, de Mahalanobis entre otras (Portillo & Plata, 2008).

Tabla 3
Restricciones estereoscópicas para correspondencia de características

Epipolar	Las imágenes de una misma identidad 3D debe proyectarse sobre la misma línea epipolar.
Semejanza	Dos imágenes de una misma entidad 3D deben poseer propiedades o atributos similares.
Unicidad	Para cada característica debe existir una sola característica correspondiente en la otra imagen.
Orden posicional	La posición de las características no se altera de una imagen a otra.
Continuidad de la disparidad	Las variaciones en la disparidad deben ser suaves, es decir debe ser continuo salvo pocas discontinuidades.
Relaciones estructurales	Los objetos deben estar formados por estructuras geométricas entre dichos elementos.

Posterior se debe de comprobar la consistencia por medio de un proceso de correspondencia global. Los dos procesos son formulados en términos de restricciones definidos en la (Tabla 3). El proceso de correspondencia estereoscópica puede tomar varias de las restricciones y desarrollar un proceso como se muestra en la (

Figura ***16***), en el cual primero se aplica la restricción de unicidad, obteniendo un mapa de disparidad temporal, posterior se realiza la restricción de continuidad para obtener un mapa de disparidad final.

Figura 16. Proceso de restricciones para determinar la disparidad

2.3.3.5. Determinación de la distancia

Una vez obtenidas las disparidades entre las imágenes obtenidas de un par estereoscópico, es posible determinar la distancia. Un solo par de imágenes no determina una distancia certera, por lo que es necesario otro punto de vista de los mismos puntos para eliminar la ambigüedad. Por lo tanto, se recomienda realizar mediciones consecutivas. Para determinar y estimar la profundidad es necesario un poseso geométrico de triangulación y semejanza de triángulos. En base a la

representación de la proyección estéreo superior de la Figura 17 se puede establecer una relación entre la profundidad y la distancia focal así:

$$\begin{cases} O_I: \frac{\frac{b}{2}+X}{Z} = \frac{x_I}{f} \\ O_I: -\frac{\frac{b}{2}-X}{Z} = \frac{x_D}{f} \end{cases} \rightarrow \begin{cases} x_I = \frac{f}{Z}(X + \frac{b}{2}) \\ x_D = \frac{f}{Z}(X - \frac{b}{2}) \end{cases} \rightarrow d = x_I - x_D = \frac{fb}{Z} \rightarrow Z = \frac{fb}{d} \quad (8)$$

Se determina que la profundidad Z es inversamente proporcional a la disparidad.

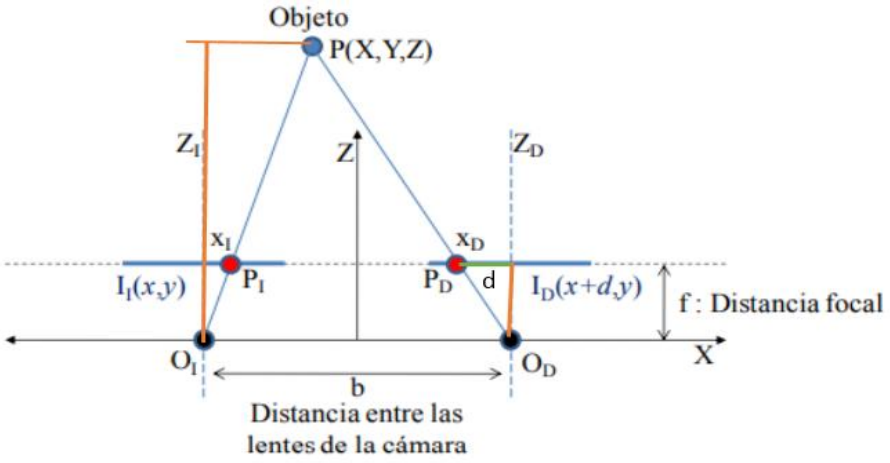


Figura 17. Representación de la proyección estéreo. Perspectiva superior
Fuente: (Hernández, Sanz, & Guijarro, 2011)

Como la profundidad Z es inversamente proporcional a la disparidad, existe una relación no lineal entre estos términos. Si la disparidad se acerca a 0, las pequeñas diferencias de disparidad dan lugar a grandes diferencias de profundidad Figura 18, entonces la disparidad es grande, las pequeñas diferencias de disparidad no cambian demasiado la profundidad.

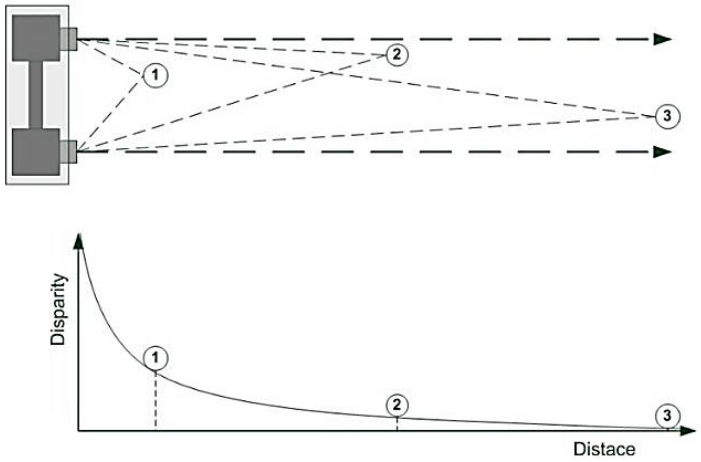


Figura 18. Relación inversa entre profundidad y disparidad.
Fuente: (Tezanos, 2015)

2.3.3.6. Estimación de Profundidad

La estimación o extracción de profundidad, es la técnica que permite de forma concreta medir la distancia de un punto o un conjunto de puntos interpretados de una escena o espacio real. Por un lado, los seres vivos perciben la profundidad por medio de la visión otorgada por los ojos, por otro lado, la inteligencia artificial y concretamente la robótica trata de replicar este hecho por medio de sistemas inteligentes. Los sistemas inteligentes o robóticos condiciones de percepción de profundidad limitadas a los sensores que usen para tal motivo.

El campo de la estimación de la profundidad por medio de robots puede ser complejo, esto debido a que se deben de desarrollar algoritmos con diversos conceptos matemáticos que modelen la realidad del entorno y en si la realidad del proceso cognitivo realizado por seres vivos. A estos se los conoce como sistemas de visión bio-inspirados (Frintrop, 2015), y en función de ello se ha desarrollado tecnología para percepción basados en el espectro de luz y sonido.

La electrónica define sensores para este propósito, los mismos que para este caso determinan la profundidad. La profundidad puede ser de dos tipos (Sanz, Mezcua, & Pena, 2012):

- **Profundidad Relativa. -** Podemos determinar si un punto está más cercano a lejos de otro.
- **Profundidad Absoluta. -** Cuando podemos determinar cuál es la distancia real entre un punto y el sensor.

De igual manera es necesario tener una medida absoluta que lidere la calibración y desarrollo de los algoritmos. Las estrategias o métodos para la estimación de profundidad pueden ser separadas en (Sanz, Mezcua, & Pena, 2012):

- **Métodos activos. -** Los cuales colocan energía dentro de la escena, proyectándola para procesar la energía reflejada. Estos métodos suelen tener una precisión elevada. Sus ejemplos más claros son:
  1. Profundidad Basada en luz.
  2. Profundidad Basada en Sonido (Ultrasonidos)
- **Métodos pasivos. -** Estos trabajan con el estado natural del ambiente, por lo general la luz ambiente es la principal fuente.

La visión estereoscópica es una herramienta que como se explicó en los apartados anteriores permite determinar la profundidad de un punto tridimensional. Esta tecnología al ser basada en la captura de imágenes de video resulta ser un método pasivo. Por otro lado, la tecnología más destacada para el desarrollo del método activo en la identificación de la profundidad son los localizadores laser. Estos generan perfiles bidimensionales del entorno y dependiendo si se desea aumentar la capacidad bidimensional que otorgan los localizadores laser, se puede reestructurar un sistema al generar movimiento en el localizador. De esta manera se genera un entorno tridimensional obtenido por sincronía de planos bidimensionales.

2.3.3.7. Localizadores Láser

La tecnología de localización láser fue introducida por medio del término LIDAR (Light detection and ranging), ente los años de 50´s. Para esa década la investigación se enfocó de forma inicial al análisis de fenómenos meteorológicos como interferencias magnéticas en la atmosfera (Collis & Ligda, 1964). Ese tipo de fenómenos basan su funcionamiento en la detección del espectro, tecnología que no se profundizará en este trabajo. Por lo contrario, lo que interesa para el presente trabajo es el tiempo de vuelo o ToF el cual se utiliza para la localización y determinación del rango.

La medición de distancias empezó a destacar con los dispositivos (EDM) o medición electrónica de distancias, a finales de los 60’s, los cuales miden por medio del cambio de fase de la onda electromagnética de la luz (Deakin, 2016). Los LIDAR modernos operan a mayor velocidad, mayor resolución, posicionamiento GPS, movimiento inercial por IMU’s, conectividad Wireless y alta capacidad de almacenamiento.

La medición de distancias por medio del LIDAR se lo realiza de forma continua, sea por tecnología de modulación o por pulsos de luz. Como factor determinante el momento de presentar un LIDAR los fabricantes deben considerar, la seguridad ocular y seguridad para la piel limita el uso del sensor. Es así que no es posible incrementar la potencia en los límites y en si el uso de rojo visible en valores alrededor de 650nm y el uso de infrarrojo en el rango de 1 a 1.5 µm como seguridad (Campbell, et al., 2002).

2.3.3.8. Funcionamiento

Este dispositivo es un sensor óptico el cual puede detectar distancias y espectros de la luz reflejada, de forma muy similar a como lo realiza un RADAR, el cual para distancias mayores a

los 100 metros determina la ubicación de objetos midiendo el tiempo de retorno de un pulso emitido. Este emite ondas de radio, a diferencia del LIDAR que envía pulsos laser. (Cano Olivera, 2010)

Un localizador en forma estática emite un haz de luz de forma directa, esperando a la respuesta reflejada. En el caso de requerir un escáner laser bidimensional es necesario la rotación del láser en su propio eje y la consecuente recopilación de la seña recibida. Esta información se correlaciona con los datos obtenidos por un encoder, logrando resultados que destacan por tener un ángulo de cobertura y una resolución angular. Para aplicaciones de vehículos en movimiento, un LIDAR 2D es suficiente ya que con él es posible utilizar el movimiento de este para alcanzar una tercera dimensión (Veitch-Michaelis, 2017). La Figura 19 muestra dos mecanismos por los cuales se escanea el has laser. (a) Muestra una disposición del LIDAR en la que por medio de un prisma o espejo rotativo el cual habilita la bidimensionalidad en el eje vertical. Este mecanismo está montado sobre una plataforma de rotación continua la cual genera movimiento en la tercera dimensión. (b) Un LIDAR comercial que ofrece una nube de puntos 3D de forma directa.

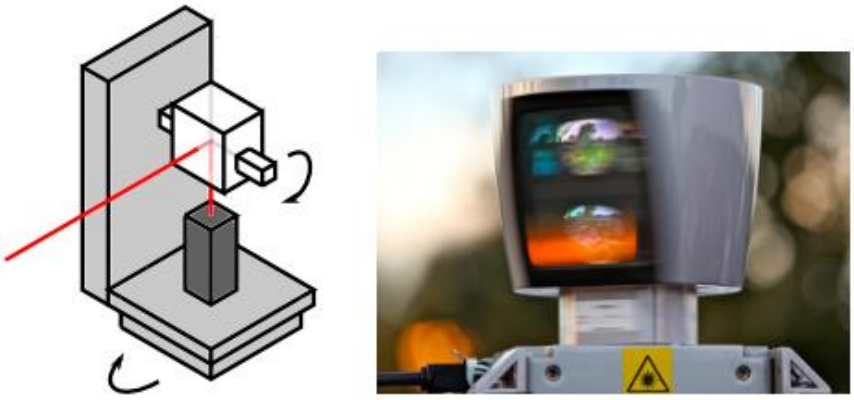


Figura 19. (a) Típica configuración de LIDAR 2D, (b) Velodhine HL-64

Fuente: Steve Jurvetson, Flickr, Licencia CC (20)

2.3.3.9. Sistema LIDAR por Tiempo de Vuelo o ToF

Esta técnica permite detectar puntos ubicados a más de 100 metros con una exactitud de centímetros. El funcionamiento radica en enviar un puso laser del cual debe recibirse el hecho antes de que pase un tiempo determinado, y así calcular el rango r definido por (Veitch-Michaelis, 2017):

$$r = \frac{ct}{2n} \tag{9}$$

Donde n es el índice de refracción y c es la velocidad de la luz en el vacío. En la (Figura 20) se precia una representación de lo que realiza el sensor ToF a manera conceptual.

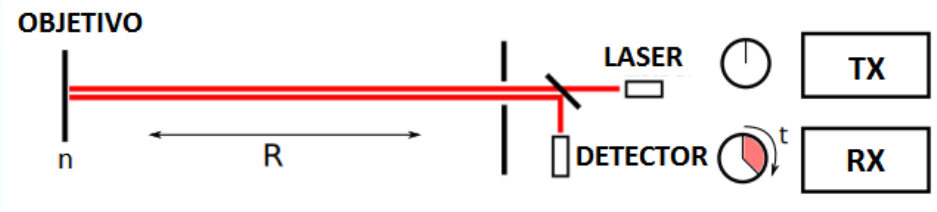


Figura 20. Imagen conceptual del LIDAR por tiempo de vuelo

La velocidad de la luz plantea un problema de temporización para los dispositivos electrónicos, una diferencia de 1 mm de distancia cambia a 6.6 ps. Sin embargo, los sistemas LIDAR pulsados permiten mediciones de largas distancias con alta precisión relativa. Por ejemplo, ahora se conoce la distancia Tierra-Lunar con una exactitud de milímetros (Pierrottet, et al., 2008).

2.3.3.10. Sistema LIDAR por Amplitud Modulada

También conocido como onda coherente u onda continua, este LIDAR puede obtener una precisión de milímetros. Es una alternativa que radica su funcionamiento por una diferencia de faces entre la señal emitida junto con la señal recibida. La diferencia de fase entre el has recibido

y el has enviado, se denomina ϕ y está relacionado con el rango de la siguiente manera definida por (Veitch-Michaelis, 2017):

$$r = \frac{\phi}{4\pi}\lambda + n\lambda \tag{10}$$

Donde n representa el múltiplo de longitudes de ondas moduladas recibidas y λ la longitud de onda recibida. En la ***Figura 21*** se encuentra el esquema para un LIDAR basado en amplitud modulada.

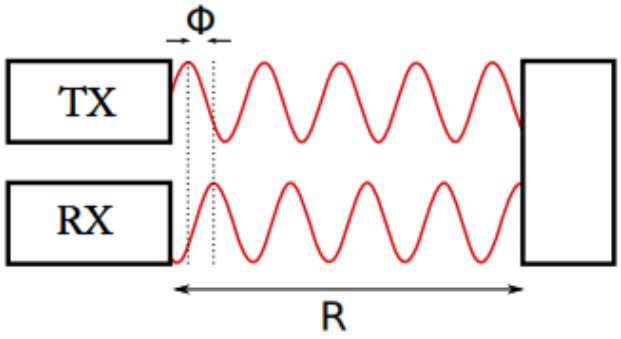


Figura 21. Esquemático para un LIDAR por amplitud modulada
Fuente: (Veitch-Michaelis, 2017)

2.3.3.11. Sistema LIDAR por Frecuencia Modulada

Al igual que el sistema LIDAR por amplitud modulada, en este caso se modula en frecuencia y la distancia se determina por medio de la medida entre la frecuencia de partida con la frecuencia de llegada (Pierrottet, et al., 2008). La diferencia de frecuencia entre el has recibido y el has enviado, se denomina Δf, llamada frecuencia de excursión, adicional la frecuencia de modulación se la define como f_{mod} y definen al rango de la siguiente manera establecida por (Veitch-Michaelis, 2017):

$$r = \frac{c(f_1+f_2)}{8\Delta f * f_{mod}} \tag{11}$$

Donde $(f_1 + f_2)$ son las frecuencias que limitan el ancho de banda.

2.3.3.12. Sistema LIDAR por Triangulación

Este sistema es comúnmente utilizado para detectar superficies dentro de un rango de 2 metros, bajo una precisión de ±1mm. Su funcionamiento radica en la utilización de un espejo rotativo encargado de remitir el haz de luz que choca con el objetivo y posteriormente se la detecta por medio de un video cámara arreglo de sensores ópticos a una distancia referencial del espejo (Cano Olivera, 2010). Es así que se determinan las coordenadas de los objetos encontrados de la manera establecida por (Peter Kaldén, 2015):

$$L = \frac{a*f}{x} \tag{12}$$

Donde L es la distancia al objetivo, a es la distancia entre los lentes y la salida del láser, f es la distancia focal y x es la posición del arreglo de sensores. Para clarificar en la ***Figura 22*** se encuentra un esquema conceptual del sistema LIDAR por triangulación.

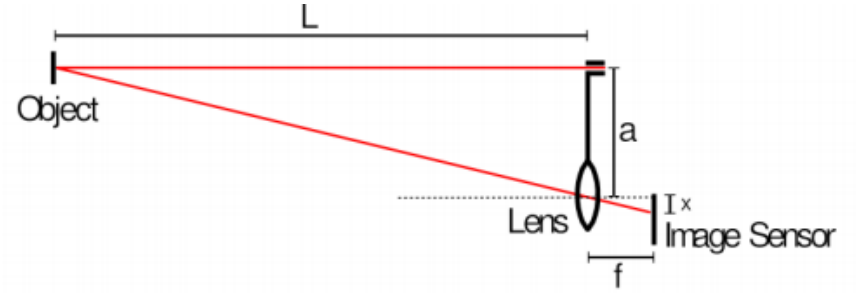


Figura 22. Esquema del sistema LIDAR por triangulación
Fuente: (Peter Kaldén, 2015)

Parámetros de los Localizadores Laser

- Potencia de Recepción

La potencia recibida por un LIDAR a partir de la distancia del objetivo *R* está definida por (Currie, Dell'Agnello, & Monache, 2011):

$$P_R = \frac{1}{4} * P_T * \tau_{total} * \rho \left(\frac{D}{R}\right)^2 \tag{13}$$

Donde P_T es la potencia transmitida, ρ es la reflectividad del objetivo, D es el diámetro de apertura, τ_{total} es el factor de transmisión debido a la atmósfera y a la eficiencia del instrumento.

- Exactitud Teórica

El rango de exactitud guarda relación inversa con la raíz cuadrada de la relación señal ruido o SNR (Baltsavias, 1999).

$$\epsilon R \propto \frac{1}{\sqrt{SNR}} \tag{14}$$

La SNR viene dada por

$$SNR \propto \Re_0 * \frac{P_T}{B} \tag{15}$$

Donde $\Re_0$ es la capacidad de ganancia del diodo receptor y B es el ancho de pulso efectivo a ruido.

- Máximo rango

Este máximo rango está determinado para LIDAR de señal continua y viene definido por (Baltsavias, 1999):

$$R_{max} = \frac{\lambda_{long}}{2} \tag{16}$$

Este rango se define como no ambiguo.

2.4. Navegación Autónoma

La robótica busca generar dispositivos y algoritmos computacionales capaces de replicar las acciones y características humanas. Una de aquellas es la navegación autónoma y en si el desplazamiento adecuado. La navegación por lo tanto se la puede definir como el problema que el dispositivo debe sustentar para desplazarse de un lugar a otro. Este problema tiene propiedades

teóricamente interesantes y de importancia práctica. La navegación es una tarea que un robot autónomo debe hacer correctamente para moverse con seguridad de un lugar a otro sin perderse o colisionar con otros objetos (Fong, Adams, Crabbe, & Schultz, 2003). Puede no solo ser una navegación terrestre sino también aérea (Aguilar, Verónica, & José, Obstacle Avoidance Based-Visual Navigation for Micro Aerial Vehicles, 2017), (Aguilar, Salcedo, Sandoval, & Cobeña, 2017), permitiendo utilizar algoritmos eficientes que permiten la navegación autónoma (Aguilar, Verónica, & José, Obstacle Avoidance for Low-Cost UAVs, 2017), (Aguilar, Angulo, & Costa-Castello, Autonomous Navigation Control for Quadrotors in Trajectories Tracking, 2017), (Aguilar, Casaliglla, Pólit, Abad, & Ruiz, 2017). Se conoce de tres problemas generales que subdividen la navegación autónoma los cuales son:

- Localización
- Planificación de trayectorias
- Control de movimiento

Dado que consideramos una UGV tipo Ackerman como plataforma, resulta necesario revisar los sistemas no holonómicos y establecer sus restricciones.

2.4.1. Sistemas No-holonómicos

La mayoría de los sistemas mecánicos están sujetos a restricciones de posición y velocidad, es decir que solo pueden alcanzar ciertos puntos dependiendo del giro, dicho de otra forma, varias relaciones entre las posiciones y las velocidades de los diferentes puntos del sistema deben ser satisfechas a lo largo del movimiento. Las restricciones se llaman holonómicas cuando es posible que el sistema pueda integrar su función de desplazamiento y alcanzar todo el espacio que lo

rodea. En el caso de restricciones no integrables, no todas las posiciones son alcanzables y el sistema se llama sistema no holonómico.

Los vehículos con ruedas convencionales están sujetos a restricciones no holonómicas, debido a que la física impide un movimiento perpendicular a las mismas, por lo tanto, la implementación de vehículos no holonómicos es un desafío para teoría de control no lineal.

Para controlar vehículos no holonómicos existen los métodos de bucle abierto y bucle cerrado, los métodos de bucle abierto buscan trayectorias factibles partiendo del estado inicial del sistema y conectado con el final. Es necesario considerar la evasión de colisiones, costo mínimo de camino, entre otras. En (Dubins, 1957) se estudió la trayectoria más corta entre dos puntos orientados, y (Reeds & Shepp, Optimal paths for a car that goes both forwards and backwards, 1990) se considera que la trayectoria más corta está compuesta de como máximo tres segmentos rectos y arcos. Sin embargo, en el método de bucle abierto, los resultados no son robustos a las perturbaciones, errores o ruidos en el sistema.

Para el sistema de bucle cerrado se considera, una función de entrada lo cual compensa ruidos y errores del sistema, esta función debe variar en el tiempo, permitiendo el desarrollo de trayectorias cartesianas, por medio de controladores PID (Normey-Rico, Alcalá, Gómez-Ortega, & Camacho, 2001).

2.4.2. Localización

El UGV como paso inicial para la navegación autónoma debe conocer donde se encuentra, por lo tanto, requiere conocer la posición y orientación exacta con respecto a un sistema de referencia, y así poder desplazarse por el entorno. Sin dicha información de esta localización, no se lograría evitar obstáculos de forma certera.

La localización consiste en determinar los componentes de traslación (t_x, t_y, t_z) y rotación $(\theta_x, \theta_y, \theta_z)$ respecto a un sistema absoluto. Para un UGV, conviene considerar, coordenadas bidimensionales por donde el vehículo posee tres grados de libertad. De esta manera el sistema queda con coordenadas (t_x, t_y, θ) asociadas a un UGV, como en la (***Figura 23***).

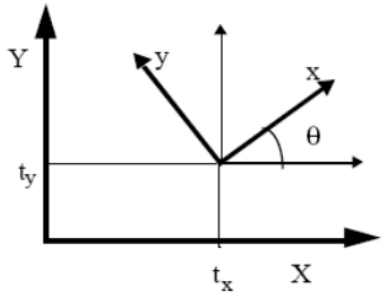


Figura 23. Sistema de referencia móvil asociada al vehículo
Fuente: (López Valverde, 2009)

La estimación de la posición puede ser dividida en dos partes; sistemas de estimación explicita y sistemas basados en el entorno (Figura 24).

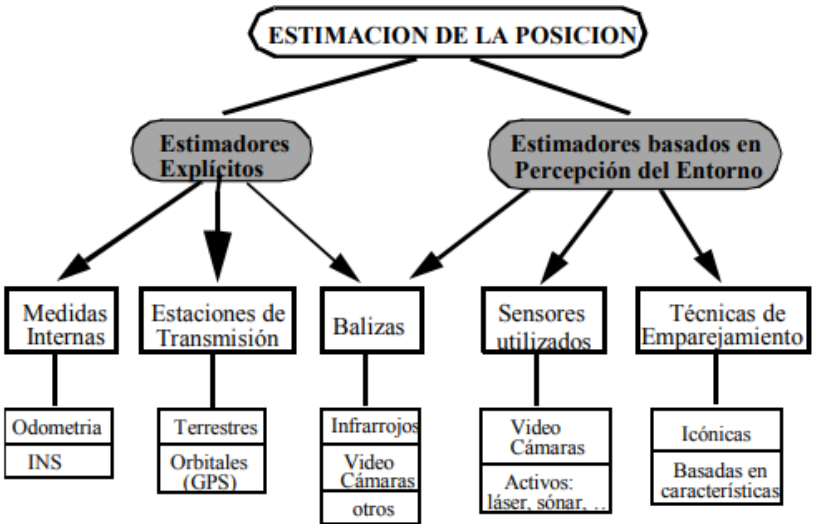


Figura 24. Sistemas de estimación de la posición
Fuente: (Jiménez & Baturone, 1996)

Los sistemas de estimación explicitas, permiten estimar la posición y orientación del vehículo, sin la necesidad de información del entorno. Para integrar la trayectoria está utilizan mediciones internas, por lo tanto, en base a esta información pueden darse sistemas por odometría y sistemas que naveguen por información inercial. Otros sistemas dentro de esta categoría son por medio de estaciones de transmisión y mediante el uso de balizas.

2.4.2.1. Sistemas por odometría

La odometría se remonta a la época de Arquímedes y tiene por objeto determinar la ubicación del vehículo a partir del número de vuelta que realicen sus ruedas (Sleeswyk, 1981), la forma más común de realizar este sistema, es por medio de codificadores ópticos, colocados sobre los ejes de las ruedas del vehículo, las ventajas de este sistema son su bajo costo y simplicidad, sin embargo necesita constante calibración debido al desgaste de ruedas, desajuste en los ejes, entre otros (Wang C. M., 1988).

2.4.2.2. Sistemas de navegación inercial

Este sistema estima la posición y orientación del vehículo, por medio de los datos inerciales provistos por el sensor, los mismos que son aceleración y ángulos de orientación, con los cuales se determina la velocidad y la posición respectivamente. Adicionalmente también es posible utilizar la orientación provista por magnetómetro. Estos sistemas no se ven afectados por las condiciones del suelo o irregularidades, esto los hace más precisos, aunque en contra parte son muy sensibles, y con un costo mayor

2.4.2.3. Sistemas basados en estación de transmisión

Este tipo de sistema se lo utiliza para la localización de dispositivos en áreas grandes y de gran cobertura. Se lo conoce también como sistemas basado en radio, los cuales se utilizan en

aplicaciones de localización marítima y aeronáutica. Su tecnología más desarrollada se denomina GPS (Global Position System) (Green, Massatt, & Rhodus, 1989), con la cual UGV's pueden contener antenas que permiten identificar su localización aplicando métodos de triangulación, entre el dispositivo y una constelación de satélites alrededor del planeta.

2.4.2.4. Sistemas basados en estimación por balizas

Este sistema permite determinar la ubicación del vehículo, conociendo la ubicación predefinida de balizas o targets con los que se triangula su ubicación y se determina la posición, para determinar la orientación es necesario conocer por lo menos dos balizas. La mayor cantidad de balizas menor es el error cometido, por lo que, este tipo de sistemas resultan ineficientes e inhabilitan la localización en entornos desconocidos.

Los sistemas basados en la percepción del entorno basan su funcionamiento en un sistema sensorial capaz de proveer información completa que permita determinar la localización del entorno. Estos sistemas generalmente utilizan distintos tipos de sensores como; cámaras, sonares, localizadores laser, localizadores ultrasónicos, entre otros. En secciones anteriores se describió de forma concreta que tanto localizadores laser como cámaras de visión permiten determinar la distancia entre el vehículo y un punto en el espacio. Es así que, de manera inversa y correlacionando múltiples valores, se puede determinar la localización del vehículo respecto al sistema percibido.

2.4.3. Mapeo

Un mapa almacena las relaciones entre elementos y una región. Existen muchos posibles formas de representar mapas, pero lo más común es la representación basados en características, en la cual el mapa es una lista de características junto con sus posiciones o sus relaciones

topológicas, el reto con los mapas caracterizados es detectar e identificar la característica, por ejemplo: un cono con característica naranja sea asociado en su entorno a lo largo del tiempo y no confunda con otros conos de la carretera, otra representación común es una serie de capturas referentes a la información del sensor, llamadas como firmas o vistas, el reto con estos mapas basados en vistas es el desarrollo de algoritmos para compararlas las vistas y extraer la posición relativa de los objetos, esta técnica se la utiliza cuando el sensor no es capaz de obtener adecuadamente las características o cuando existen o un set o un conjunto de características desconocidas.

Una tercera forma de representar y utilizada comúnmente en la robótica es el mapa de grilla Figura 25 el cual consiste en una discretización, del espacio en el cual cada celda tiene un valor asignado y una probabilidad de ocupación (Moravec & Elfes, 1985). El reto de realizar un mapa de grilla es que puede someterse a situaciones en las que la información del sensor no es suficiente para identificar características.

La elaboración de mapas junto con la localización de forma simultánea define al problema denominado SLAM, el cual es uno de los temas más investigados en el campo de la robótica. Esto es muy útil en la elaboración de mapas en ambientes desconocidos, en las que el robot entiende la información acerca de su localización, y de forma directa su trayectoria (Santos, Portugal, & Rocha, 2013).

Para generar mapas en 2D lo más utilizado son la técnica SLAM basada en grilla establecida para ambientes internos, sin embargo, este algoritmo no puede ser usado en ambientes 3D, debido a que una grilla es asociada a una altura definida, a pesar de haber cambios de altura la información es incorrecta (Fong, Adams, Crabbe, & Schultz, 2003). En un ambiente 3D, el espacio se lo divide en cubos llamados vóxeles, los cuales contienen el nivel de ocupación.

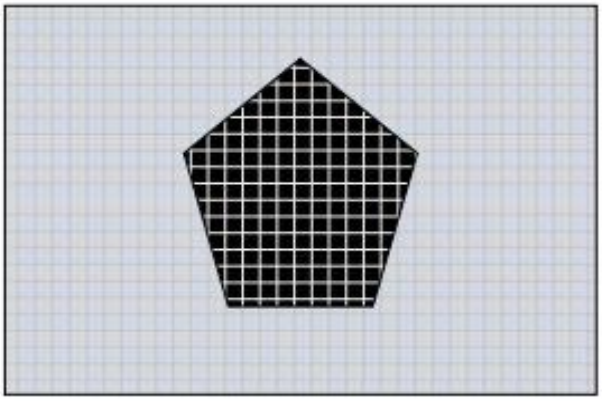


Figura 25. Mapa basado en grilla de ocupación
Fuente: (Wang X. , 2013)

2.4.3.1. Mapas basados en grilla de ocupación

Es ya denominado como el estándar para mapeo bidimensional, cada celda o grilla tiene un valor proporcional a su estado de ocupación. Básicamente el mapa basado en grilla de ocupación tiene 2 premisas (Wang X. , 2013).

- Todas las grillas o celdas del mapa son independientes de igual forma que su grado de ocupación
- La posición del vehículo se conoce de antemano

Para actualizar el mapa se utiliza, el filtro de valles el cual es un método probabilístico para estimar probabilidad desconocida, basándose en la medición actual y la estimación anterior. Este método va acorde las cadenas de Markov, determinando el presente estado, el estado anterior y el futuro estado, en función de si está o no ocupado.

2.4.3.2. Técnica para SLAM 2D

Para el desarrollo para el presente trabajo de investigación, se opta por un algoritmo llamado Core-SLAM Figura 26, el cual en su forma original contiene tan solo 200 líneas de código, por lo

tanto fue creado con el propósito de ser simple y fácil de entender con el mínimo perdida de desempeño (Steux & El Hamzaoui, 2010). El algoritmo consta de dos pasos principales:

- **Cálculo de distancia. -** para cada escaneo que llegue se calcula la distancia por medio de un algoritmo PF (filtro de partículas). El PF une empareja cada escaneo del láser con cada partícula representando una posible pose del robot y un peso asociado, que depende de las iteraciones anteriores. Luego se elimina las partículas con menor peso y nuevas partículas son generadas,
- **Actualización de mapas. -** las líneas correspondientes a los escaneos recibidos se dibujan en el mapa. No solo se dibuja un solo punto sino se dibujan todos los puntos que rodean.

Figura 26. Core-SLAM
Fuente: (Santos, Portugal, & Rocha, 2013)

2.4.4. Planificación de Trayectorias

La planificación de trayectorias para vehículos terrestres no tripulados UGV's ha sido ampliamente estudiado en la literatura. Usualmente los métodos de solución para problemas de planificación de trayectorias se han integrado ya en productos comerciales, como servicios de

mapas en línea, unidades de GPS portátiles y sistemas de navegación para automóviles. El inconveniente es que, pese al desarrollo ya implementado, los detalles a fondo de estos algoritmos comerciales no están disponibles. Es así que se recurre a métodos estándar discutidos en la literatura. Una revisión de algunos de estos métodos se encuentra en (Wu, Hartley, & Al-Dabass, 2005).

De forma estándar la planificación de trayectorias tiene por objetivo planificar un camino desde un punto conocido a otro punto en un mapa predefinido. A menudo el mejor camino o trayectoria se determina por la longitud mínima, tiempo mínimo o alguna métrica especificada. La planificación de trayectoria puede ser dividida en dos categorías: planificadores de rutas locales y planificadores de rutas globales.

Cuando el ambiente es totalmente conocido y antes de que el robot se mueva una trayectoria libre de colisiones con un costo bajo se obtiene desde un punto de inicio a un punto destino por medio de algoritmos de planificación de trayectorias globales. Por otro lado, los algoritmos de trayectoria locales toman en consideración puntos de referencia de la trayectoria global y hacen posible que se generen trayectorias parciales las cuales consideran la dinámica de los obstáculos, las restricciones cinemáticas del vehículo, por lo tanto se recalcula el mapa cada cierta razón de tiempo o distancia. Varios enfoques para la generación de trayectorias, por medio de: líneas de Clotoides (Gim, Adouane, Lee, & Dérutin, 2017), líneas de Bezier (Rastelli, Lattarulo, & Nashashibi, 2014) arcos y segmentos (Reeds & Shepp, Optimal paths for a car that goes both forwards and backwards, 1990) o spline (Piazzi, Bianco, Bertozzi, Fascioli, & Broggi, 2002).

2.4.4.1. Algoritmos de planificación global

En el estado del arte existen varios tipos de algoritmo para la planificación, cada uno con ventajas e inconvenientes siendo seleccionando según el problema a tratar, hay algoritmos que requieren mayor tiempo de ejecución, algoritmos que consideran restricciones cinemáticas o dinámicas del problema. Existe un conjunto de elementos que se deben definir en el ámbito de la planificación de trayectorias para un UGV, las cuales son consideradas a continuación (Latombe, 1990):

- **Configuración del robot. -** Parámetros mínimos para identificar la posición y orientación de este.
- **Configuración del origen. -** Estado que presenta el vehículo antes de iniciar.
- **Configuración destino. -** configuración de la meta que se desea alcanzar.
- **Espacio de configuraciones. -** conjunto de todas las configuraciones que puede alcanzar el vehículo.
- **Colisión. -** Intersección entre el espacio ocupado por los obstáculos y el vehículo.
- **Obstáculo. -** conjunto cerrado y acotado de puntos de un espacio inaccesible para el robot.
- **Región de obstáculo. -** conjunto de todos los obstáculos.
- **Región libre de colisión. -** conjunto de espacio que no presenta colisión.
- **Trayectoria. -** sucesión continua del camino adoptado por el UGV, que parte del origen al destino.

Los algoritmos de planificación pueden basar su funcionamiento en métodos tales como (Latombe, 1990): métodos RoadMaps, métodos de descomposición en celdas, métodos de campo

de potencial y métodos probabilísticos. Para lo cual los más importantes son los descritos a continuación:

- **Grafos de visibilidad. -** Se basa en la construcción de un grafo que representa las posibles configuraciones del vehículo junto con los costos y vértices. Para hallar el camino se conecta dichos puntos y se halla una secuencia de nodos (Muñoz, 1995).
- **Diagramas de Voronoi. -**Son grafos resultantes de maximizar la distancia entre el vehículo y los obstáculos, esto define un atributo de vecindad, que determina que no existe algún elemento de obstáculo en el vehículo (Aurenhammer, 1991).
- **Descomposición en celdas. -** De forma general estos métodos consisten en dividir la zona en espacios locales y globales resolviendo espacios libres de colisión y conectando a nivel superior, los espacios por medio de una función que determina conexiones próximas (Latombe, 1990).
- **Campos de potencial. -** Este método trata al vehículo como un punto o partícula sometida a diferentes fuerzas, una de las fuerzas de atracción se enfoca hacia el destino y fuerzas de repulsión evitan la colisión con obstáculos. Para definir las fuerzas se consideran un campo de potencial, perteneciente al escenario (Latombe, 1990).
- **Mapas probabilísticos (PRMs). -** Algoritmo eficiente en el cálculo de trayectorias para vehículos con muchos grados de libertad, los mapas probabilísticos pueden ser mapas de consulta única o múltiple, su funcionamiento radica en la elaboración de grafos de forma aleatoria verificando constantemente si tanto nodos como segmentos están libres de colisión y si pueden conectarse por un medio de planificador local (Kavraki, Kolountzakis, & Latombe, 1998).

- **Arboles aleatorios exploración rápida (RRT). -** Este algoritmo basa su funcionamiento en la construcción de un árbol de exploración. Esto se realiza de forma aleatoria por medio de vértices o puntos interconectados que parten explorando desde un punto origen hasta que una de las ramificaciones conecte con la meta. Existen métricas y conceptos importantes para comprender el algoritmo los cuales son:

ρ: Métrica la cual puede distancia euclidiana u otra función heurística

q_{ini}: es la configuración inicial del vehículo, tanto en posición como en orientación.

q_{goal}: Configuración final que se desea alcanzar

q_{rand}: Configuración aleatoria dentro del espacio

q_{near}: Configuración próxima a q_{rand}, y que ya pertenece al árbol.

q_{new}: Configuración nueva que se añade al árbol

ε: paso o segmento que el árbol puede crecer.

2.4.4.2. Algoritmo de planificación RRT

El algoritmo RRT original crea un árbol para que cubra de forma uniforme todo el espacio libre de colisión hasta encontrar la meta de forma aleatoria y extenderse generando una trayectoria, el algoritmo se muestra en la (Figura 27).

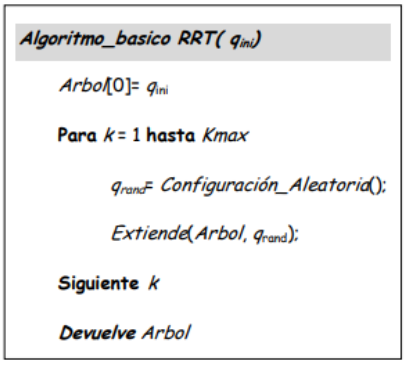


Figura 27. Algoritmo RRT básico
Fuente: (LaValle, 1998)

Dentro del algoritmo existen dos funciones: la primera es generar un punto aleatorio en el espacio y la segunda es extender o hacer crecer al árbol en dirección a la meta como se muestra en la ***Figura 28***.

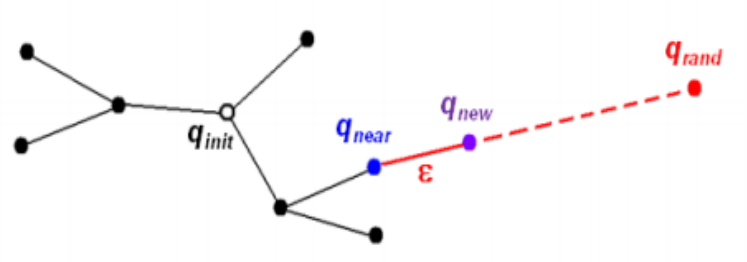


Figura 28. Esquema de expansión del RRT
Fuente: (López García, 2011)

Este algoritmo a lo largo del estado del arte a surtido diferentes modificaciones, las modificaciones que se pueden dar a este código dependen de las restricciones dinámicas empleadas en simulaciones controladas (Kim J. , 2003). Las restricciones dependen de la dinámica de la plataforma móvil o robot en el que se implementará. Una restricción crítica ocurre en plataformas no holonómicas, esto significa que la plataforma no puede modificar su dirección instantáneamente y será necesaria una modificación llamada SRRT que considera trayectorias curvas en el árbol (Moon, 2013). El algoritmo RRT requiere que se conozca el espacio de trabajo y que se tenga información adecuada del entorno, existe la posibilidad de realizar una optimización del entorno propuesta por (James, 2000). En esta variante llamada RRT-Conect los objetos se clasifican como valiosos y sin valor. Los objetos valiosos se colocan entre los puntos de inicio y final, considerando los límites de ellos y enfocando de mejor manera el crecimiento del árbol. Está demostrado que de esta manera el RRT elige la trayectoria más factible.

Modificaciones se ha realizado para que el algoritmo funcione en tiempo real y para que se desarrolle nuevos algoritmos con un enfoque de muestreo RRT* (Naderi, Rajamaki, & Hamalainen, 2015), (Aguilar W. G., Morales, Ruiz, & Abad, RRT* GL Based Optimal Path

Planning for Real-Time Navigation of UAVs, 2017). Su contribución se basa en la reconexión estratégica que permite el movimiento de la plataforma sin descartar las trayectorias previas. Además, no espera que el árbol esté completo en tiempo real. El algoritmo combina expansión de árbol y retroceso en dos modos; Desplazamiento dinámico de raíz y muestreo de parches. La desventaja es que trabajar en tiempo real requiere una gran capacidad de memoria y el entorno resulta limitado. Este algoritmo considera que el árbol debe enfocarse en crecer hacia la meta haciendo uso de una función heurística. En (Junxiang Ge, 2016), propusieron una aplicación de RRT en la ruta de un manipulador. La modificación que se produce en el algoritmo se realiza en la función de muestra, para lo cual se limita el espacio correspondiente a donde existe el punto objetivo. Este espacio se conoce como espacio útil y puede identificarse como una función circular con centro en el objetivo u otras formas como un cuadrado. En algunos casos, el espacio de muestra debe modificarse, utilizando funciones que permitan que el espacio de muestra se comporte dinámicamente.

Si el espacio dimensional en el que se realiza el muestreo aumenta el algoritmo RRT reduce su rendimiento. En (Risheng Kang, 2016) se plantea una variación de RRT basada en el muestreo incremental llamado Fast Convergence Rapidly-exploring Random Tree (FCRRT). Las principales mejoras son que la exploración y la optimización se realizan por separado para permitir la retención de la fuerza de escaneo. Y el uso de RRT* GL (Aguilar W. G., Morales, Ruiz, & Abad, RRT* GL Based Path Planning for Virtual Aerial Navigation, 2017) que acelera la tasa de convergencia. RRT* GL es retrasar las comprobaciones de colisión para algunos o todos los nodos. Con esto se logra que solo un pequeño grupo de nodos se involucre final.

Aplicaciones que tienen esta herramienta dentro de la planificación de movimientos como; La planificación intraoperatoria de movimientos tridimensionales en operaciones clínicas llevadas a

cabo por robots (Yan-Jiang Zhao, 2015), la virtualización de entornos (Aguilar & Morales, 3D Environment Mapping Using the Kinect V2 and Path Planning Based on RRT Algorithms, 2016) la planificación de rutas de transporte (Salgado, Tierra, & Aguilar, Travel Planning in Public Transport Networks Applying the Algorithm A* for Metropolitan District of Quito, 2017), (Salgado, Tierra, Sandoval, & Aguilar, 2017), la anticipación eficiente de colisiones (Lee, 2016), la evasión de obstáculos; lo cual aumenta la garantía del algoritmo RRT y lo traduce en un algoritmo predictivo (Lite, 2016). Otros algoritmos para la planificación del movimiento en vehículos autónomos consideran; La variación angular entre vértices u orientación con la que el árbol crece (Kim M. H., 2014), el uso de áreas potencialmente peligrosas (Purcaru, 2013), trayectorias dinámicas (Sara Bouraine, 2016) y uso redes neuronales (Verma, 2014), todo con el fin de mejorar el algoritmo y permitir que se desarrolle una navegación autónoma.

2.4.5. Reconstrucción 3D

Parte de las características detalladas en la percepción o identificación de profundidad con una cámara de estéreo visión permite la percepción 3D, para ello es necesario tener cámaras completamente calibradas para posterior se realice la triangulación de puntos correspondientes de varias imágenes, es necesario aplicar una geometría para la proyección y cotejo de los puntos. Consecuentemente se debe integrar las texturas considerando los puntos de interés y los modelos generados.

La creación de modelos fotorrealistas tridimensionales (3D) calibrados de escenas observadas ha sido un tema de investigación activo durante muchos años. Dichos modelos tridimensionales son muy útiles tanto para la visualización como para la medición en diversas aplicaciones como construcción, militares, mineros, forenses, arqueológicos, de realidad virtual, y otros (Se, 2008).

El uso directo de las características de una cámara estereoscópica permite realizar un mapeado del entorno en tres dimensiones sin necesidad de ningún otro dispositivo adicional o complementario, esto se lleva a cabo uniendo puntos y secciones de interés dando resultados que permiten percibir el entorno.

El mapeo de rango estéreo de velocidad tiene muchas ventajas. Es pasivo y no emite ninguna radio o energía de luz. Con una geometría de imagen apropiada, ópticas y cámaras de alta resolución, el estéreo puede producir una imagen de rango denso y preciso. La información de rango se alinea con la información visual en las coordenadas de imagen comunes (Yoshida, 1996).

Para poder efectuar el mapeo 3D es necesario conocer la posición actual de la cámara, Debido a su alto rango de medición y precisión, los escáneres láser y los sistemas de cámara estéreo son los que se utilizan para realizar SLAM hasta el momento. Pero existen algunas restricciones: la visión estereoscópica requiere que coincidan los puntos correspondientes de dos imágenes, pero existen varios factores que afectan la obtención de un entorno usando cámaras especialmente debido a sus complejas características de error y alto ruido de medición. Dependiendo de factores interferentes externos (por ejemplo, luz solar) y configuraciones de escena, es decir, distancias, orientaciones y reflectividad, la misma escena implica grandes fluctuaciones en las mediciones de distancia desde diferentes perspectivas (May, 2009).

La imagen estéreo puede recuperar la estructura del entorno al combinar las funciones detectadas en múltiples imágenes de la misma escena. Es muy computacionalmente intensivo ya que los datos 3D se calculan a partir de las imágenes. Los datos de profundidad podrían ser más ruidosos. A diferencia de los escáneres láser, las cámaras pueden capturar imágenes completas en microsegundos, por lo tanto, pueden usarse como sensores móviles o funcionar en entornos

dinámicos (Panich, 2010). Los requisitos de costo, tamaño, masa y potencia de las cámaras estéreo son mucho más bajos que los de los telémetros de exploración.

2.4.5.1. Captura de información 3D

El objeto de crear un mapa de percepción 3D es determinar características visuales a áreas de construcción con geometría específica, todos los procesos empiezan con la extracción de las imágenes de disparidad previamente descritas. Al manipular estas imágenes se identifica que dentro de cada una existe grado de incertidumbre Δd en cada pixel, este grado de incertidumbre se genera a partir de la gradiente entre imágenes de disparidad. Esto se traduce a un factor de error Δz, donde z es la disparidad calculada y se da la siguiente ecuación:

$$\Delta z = \frac{z^2}{b \cdot f} \cdot \Delta d \tag{17}$$

Para generar un modelo de precepción ya sea utilizando grillas de ocupación u otro método, es necesario modelar el sensor. Una vez obtenido el modelo del sensor, se interpola la geometría de los puntos para cada disparidad o factor z, con lo cual se genera un banco de datos que permiten generar un mapa de ocupación (Hong & Chen, 2004). Algunos consideran inherente incluir los errores que posee el vehículo al realizar movimiento y esto se lo ve en (Souza, Maia, & Gonçalves, 2012).

2.4.5.2. Construcción del modelo

Los registros de datos previamente obtenidos deben ser superpuestos entre varios para reducir ruido, estos deben contener un buen detalle que permita la visualización y manipulación de los datos, para construir modelos geométricos tridimensionales, es necesario hacer mallas de triangulación las cuales consideran el mayor número de triángulos y la cantidad de los mismos reflejan la complejidad de la superficie.

Las mallas triangulares que consisten en una gran cantidad de triángulos se usan a menudo ya que pueden representar superficies complejas Los modelos se pueden obtener creando mallas de superficie desde vistas individuales primero y luego uniéndolas (Levoy, 1994). Si hay una superposición significativa entre las vistas individuales, este enfoque es bastante ineficiente debido a la necesidad de puntadas repetidas. El enfoque volumétrico es más eficiente ya que los puntos 3D se acumulan primero en las estructuras de la cuadrícula vóxel. Solo se crea una malla triangular para todas las mediciones (Se, 2008).

El paso principal para la obtención del mapa en tres dimensiones es poder realizar los objetos con sus múltiples vistas en un solo modelo. El objetivo de la integración es llegar a una descripción de la topología general del objeto que se está escaneando. La topología completa de una superficie se realiza mediante la creación de nuevos escaneos de rango uno por uno en la malla final del triángulo (Levoy, 1994). Por lo general para solucionar los errores clásicos que pueden ocurrir en un proceso de obtención de malla, se usan los siguientes pasos:

- Eliminar las partes superpuestas y redundantes de las mallas.
- Unir una malla contra otra.
- Eliminar los triángulos pequeños introducidos durante el recorte

El modelo a ser obtenido creando superficies o mallas de las primeras vistas individuales y luego coserlas (Turk & Levoy, 1994). Si allí existe una superposición significativa, entre las vistas individuales, existe la necesidad de coser repetidamente. El efecto volumétrico es más notable a medida que se acumula los cosidos, estas estructuras se almacenan en una red de vóxeles, posteriormente se extrae una malla de la superficie en la cual se ignoran pequeñas

desviaciones triangulares y se aumenta los cubos. Después se genera una malla triangular referente a las texturas y se las filtra generando foto realismo (Soucy, Godin, & Rioux, 1996).

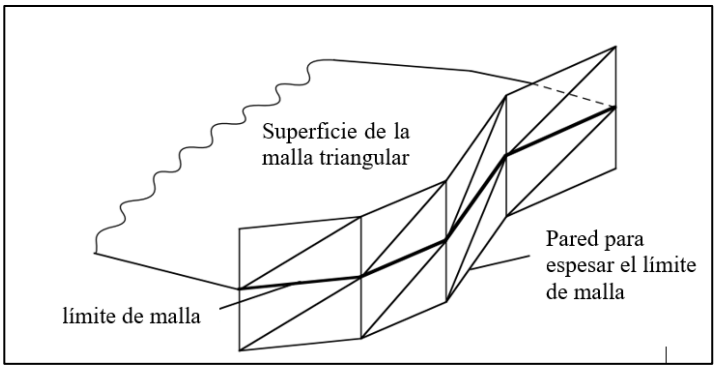


Figura 29. Representación de malla triangular con cortes en tres espacios
Fuente: (Levoy, 1994)

2.4.5.3. Textura y renderización

La realización de un renderizado contrasta con las modelos foto realísticos obtenidos esto permite producir imágenes que se pueden visualizar en tres dimensiones, los avances en renderizado habilitan el uso de computadoras de gran capacidad lo cual mejora el estilizado y realizan procesos estéticos como en (Gonzalez & Woods, 2002)**.** Este como otros algoritmo y sistemas producen una reedición monoscopica, por medio del procesamiento de la geometría. Solo recientes técnicas que utilizan múltiples tipos de vista pueden habilitar un renderizado de mayor capacidad.

Es incierta la resolución o la capacidad visual necesaria para garantizar que el entorno se replique, algoritmos innovadores limitan las imágenes generadas en función de la percepción humana en lugar de desbordar información excesiva se aumenta el foto realismo o datos estilizados. El observador no necesita información excesiva, una imagen sintética y con una foto realismo adecuado lleva el equilibrio entre la percepción humana del entorno y su factibilidad crítica al momento de renderizar (Stavrakis).

CAPITULO III

3. CARACTERIZACIÓN DEL SISTEMA

En el presente capítulo se detalla el procedimiento efectuado para la caracterización del sistema de percepción 3D y planificación de trayectorias para un UGV. El sistema como ya se estableció necesita el uso de un LIDAR 2D, una cámara de estéreo visión y una plataforma móvil UGV. Es clave caracterizar estos dispositivos, establecer sus parámetros y especificaciones con el fin de que se desarrollen los algoritmos. Resulta clave definir los perfiles de profundidad e ambos sensores para el desarrollo de la percepción tridimensional y la planificación.

El LIDAR 2D utilizado es un localizador láser de medio alcance, preciso y de alta velocidad, el cual requirió acondicionamiento y posterior caracterización de comportamiento. Por otro lado, la cámara de estéreo visión es una cámara de alta resolución la cual requiere del uso de una tarjeta gráfica. La cámara estereoscópica como se explicó en el anterior capitulo, tienen la capacidad de estimar la profundidad. Con eso se pretende mejorar el sistema de planificación, haciendo trabajar conjuntamente al sensor laser y la estéreo cámara, de manera que en todo momento se conozca la profundidad del entorno en un plano perpendicular a lo que el láser detecta. El objetivo de realizar esta combinación es que el vehículo pueda detenerse cuando encuentre obstáculos que se encuentren a una altura diferente de la que el láser puede percibir.

Complementariamente la plataforma UGV es una plataforma Ackerman diseñada para el uso de niños por lo que requiere de consideraciones específicas, esto junto al uso correcto de los parámetros de los sensores, consideraciones de comunicación y pruebas de calibración, determinaran la precisión tanto en la percepción como en la trayectoria.

Posterior a definir las características de los sensores es necesario establecer las condiciones de trabajo para el desarrollo de pruebas para cada sensor, condiciones tanto estáticas como en movimiento. Las condiciones establecidas para el espacio de trabajo son locación, iluminación, posición cartesiana e inclinación. Con estas condiciones se procede a obtener los perfiles de profundidad respectivos en un espacio controlado.

3.1. Especificaciones técnicas del LIDAR 2D

El localizador láser es utilizado comúnmente en el levantamiento de mapas en dos dimensiones debido a la precisión y exactitud con la que genera sus mediciones, así también la resolución y la cantidad de puntos en el perfil que recrea. El Centro de Investigación Científica y Tecnológica del Ejército - CICTE proveyó al proyecto con el localizador láser Hokuyo UST-20LX mostrado en la (Figura 30). El sensor posee las características mostradas a continuación:

Tabla 4

Especificaciones Técnicas Hokuyo UST-20LX

Voltaje y Corriente de Alimentación	10 - 30 VDC y <150mA
Fuente de Luz	Diodo laser semiconductor de λ=905nm
Rango de detección	0.06m a 20m - Máxima detección: 60 m
Velocidad de escaneo	25ms (Velocidad del Motor 2400rpm)
Exactitud y Precisión	< 30mm , ± 40mm
Angulo de Barrido	270°
Resolución Angular	0.25°
Resistencia a luz ambiente	15000 lx o menos
Condiciones Ambientales	-10°C a 50°C al 85% RH
Interface de comunicación	Ethernet 100BASE-TX
Tecnología de Funcionamiento	Escaneo por tiempo de vuelo ToF

Fuente: (Acroname, 2014)

Figura 30. Localizador Láser
Hokuyo UST-20LX

Este sensor utiliza una fuente laser rotativa para determinar la posición de los objetos en un campo de visión de 270° a una distancia máxima de 20 metros. Es así que con 1080 pasos el sensor alcanza la resolución angular de 0.25° como se muestra en el esquema de la Figura 31.

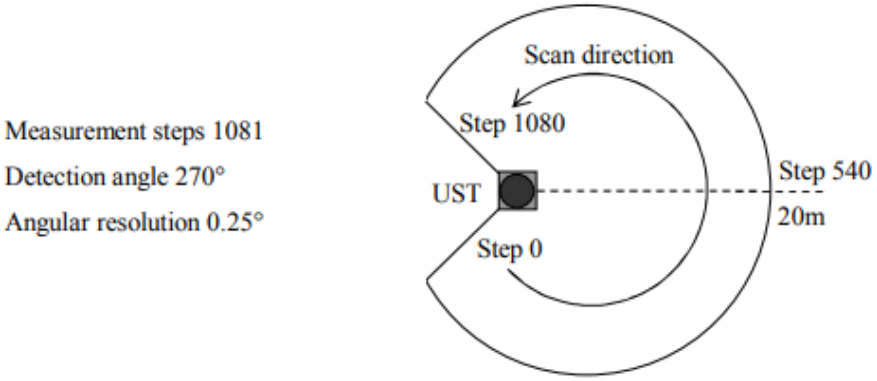


Figura 31. Esquema de detección láser UST
Fuente: (Acroname, 2014)

Acorde con el diagrama de la (Figura 31), se define que el sensor requiere de una fuente de alimentación continua, para lo cual se utiliza una batería de litio LIPO de 3 celdas a 11.1 VDC y 1300mAh mostrada en (Figura 32) lo cual garantiza una autonomía de por lo menos 8 horas de uso continuo. La fuente de luz láser pertenece a la clase 1. Esta clasificación es un parámetro de

seguridad que define el daño potencial en ojos y piel humana. Para este caso la clasificación detalla que el láser es seguro en condiciones de uso normales y siempre y cuando no se exponga la visión de forma directa y prolongada o sea por un aumentador óptico (Laser Safety Facts, 2018).

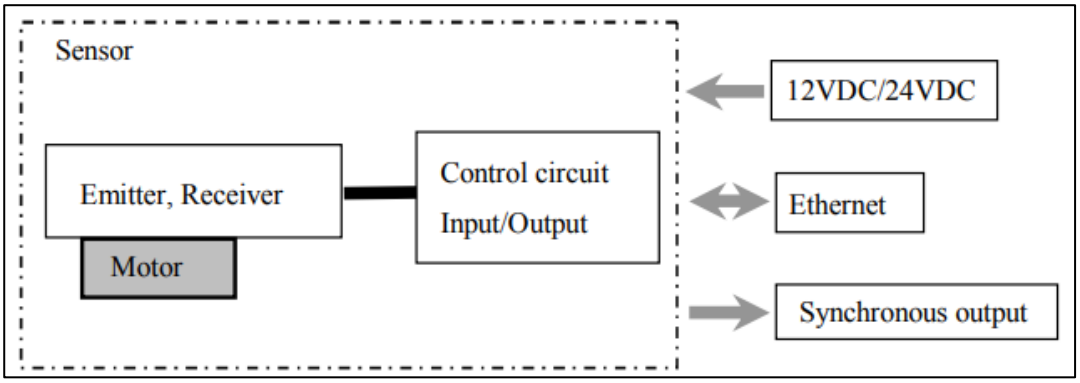


Figura 32. Diagrama de la estructura del sensor UST-20LX
Fuente: (Acroname, 2014)

Figura 33. Batería Turnigy LIPO,
11.1VDC - 1300mAh

Los datos generados por el sensor son emitidos por medio de un canal de comunicación Ethernet acorde al diagrama mostrado en la Figura 32. Por lo cual se dispone de dos ordenadores con una tarjeta de red que soporte 100 Mbit/s con auto-negociación de velocidad. El un ordenador con sistema operativo Windows 7 Profesional y otro con sistema operativo Ubuntu 16.04. El enlace de conectividad ethernet se define con una dirección IP estática y su respectiva

dirección de subred descritas en la (Tabla 5). Adicionalmente hay que tener en cuenta que el sensor ocupa el puerto 10940, el cual debe de estar en libertad.

Tabla 5
Direccionamiento de red

Dispositivo	Dirección IPv4	Mascara
Sensor UST-20LX	192.168.0.10	255.255.255.0
Ordenador Windows 7	192.168.0.11	
Ordenador Ubuntu 14.04	192.168.0.9	

Los parámetros que definen la detección como; rango, exactitud, precisión, dispersión acorde al color, resolución y ángulo de barrido son fundamentales para el desarrollo del sistema por lo cual son caracterizados y verificados por medio de un método empírico. Este método empírico hace referencia a lo realizado por (Costella & Rodríguez, 2017) los cuales caracterizaron un sensor Hokuyo URG-04LX el cual es un sensor similar, pero que presenta menores prestaciones. Por lo tanto, las hipótesis y pruebas planteadas allí resultan válidas.

3.2. Caracterización y verificación de parámetros del LIDAR 2D

Se realizaron las mediciones por medio de la aplicación Urg Viewer (Multi-echo); descargada de la web oficial del distribuidor URG Network e instalada en el Windows. En la (Figura 34) se aprecia la interfaz del programa el cual permite conectarse de forma directa a la dirección del sensor, funciones adicionales son; ver en tiempo real los valores de distancia e intensidad de reflexión, grabado de lecturas definidas, lectura específica de un valor de los 1080 correspondiente a 270° con un offset de 90° y grabar en archivo plano separado por comas o csv.

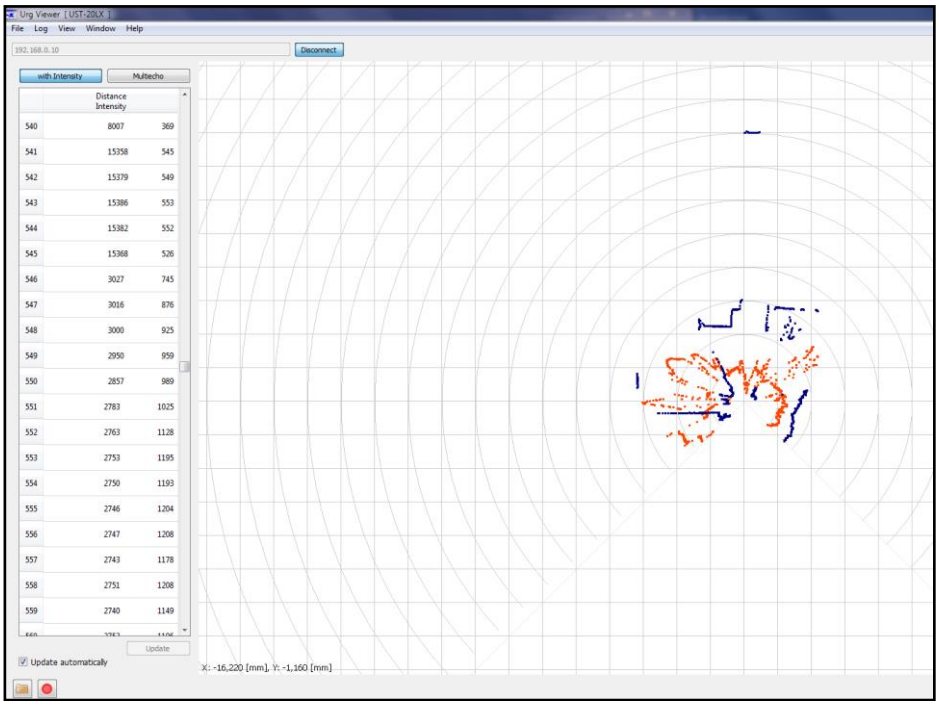


Figura 34. Urg Viewer (Multi-echo)

3.2.1. Rango de detección

Las pruebas fueron ejecutadas en base a las especificaciones referenciales del sensor, es así que corresponde distancias una distancia mínima de 60 mm y una máxima de 20000 mm. Se ejecutaron las mediciones tomando en cuenta que en la tecnología de tiempo de vuelo por láser responde ante colores diferentes y que el nivel de iluminación sea menor a 15000lx. Como se sugiere en el método se mantuvo encendido el sensor por una hora antes de empezar las mediciones. Esta condición permite que el sensor se estabilice en las condiciones nominales de velocidad, temperatura y energía.

Se consideró que el sensor tendrá visión directa si se lo coloca en el UGV, por lo tanto, se las mediciones se realizaron de forma directa y evalúa el paso intermedio denominado paso 540. De forma inicial se consideró láminas de fomis de dimensión 30 x 21 cm como objetivo como s

muestra en la Figura 35. Las pruebas descritas se realizaron en color blanco con coeficiente de reflexión de 0.7 a 0.85 y en negro con coeficiente de 0.03 a 0.07.

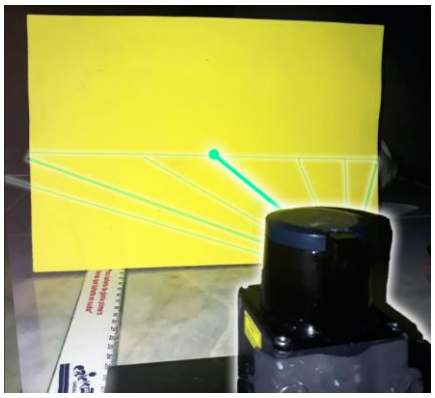


Figura 35. Medición con el sensor UST-20LX

Para verificar la distancia de detección mínima, las condiciones para efectuar la validación del rango mínimo fueron las siguientes:

- Localización del sensor: 10 cm de altura desde el piso
- Localización del objetivo: 60 mm del sensor
- Tiempo de muestreo: 6 (s)
- Periodo de muestreo: 30ms
- Número de Lecturas: 100
- Nivel de iluminación: 20 lx

Los resultados en la (Figura 36) muestran la variabilidad existente en las mediciones del sensor para el valor de detección mínima, que según (Costella & Rodríguez, 2017) se generan por color del objetivo, temperatura, estabilización electrónica, tiempo, y ángulo del objetivo. La línea de color azul corresponde a las mediciones realizadas para el objetivo de color negro y en la cual se determinó un valor medio de 64 mm, valor máximo de 74 mm y mínimo de 51 mm. Por otro

lado, el objetivo de color blanco al cual corresponde la línea de color rojo se le determino un valor medio de 120 mm, valor máximo de 136 mm y mínimo de 110 mm.

El rango de detección mínima muestra un alto grado de precisión que para ambos casos es menor a lo establecido de 40 mm. La exactitud de la medición varía del material empleado y duplica en lo establecido de que el valor debe de ser menor a 30mm.

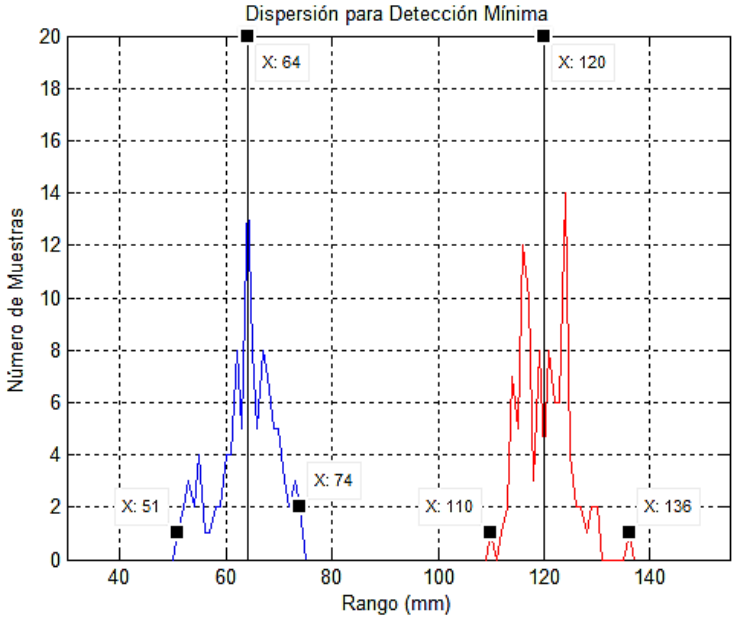


Figura 36. Dispersión para detección mínima para sensor UST20-LX

Por otro lado, para verificar la distancia de detección máxima, las condiciones las siguientes:

- Localización del sensor: 10 cm de altura desde el piso
- Localización del objetivo: 20000mm del sensor
- Tiempo de muestreo: 6 (s)
- Periodo de muestreo: 30ms
- Número de Lecturas: 100
- Nivel de iluminación: 31 lx

Los resultados en la (Figura 37) muestran mayor variabilidad existente en las mediciones del sensor para el valor de detección máxima. La línea de color azul corresponde a las mediciones realizadas para el objetivo de color negro y en la cual se determinó un valor medio de 18380 mm, valor máximo de 20350 mm y mínimo de 15520 mm. Por otro lado, el objetivo de color blanco al cual corresponde la línea de color rojo se le determino un valor medio de 18770 mm, valor máximo de 20350 y mínimo de 15520.

El rango de detección máxima muestra un pobre grado de precisión que por más de 4m para ambos casos, lo cual es mayor a lo establecido de 40 mm. La exactitud tampoco se encuentra en sus parámetros adecuados. Se rescata de esta prueba que el sensor garantiza repetitividad en sus mediciones.

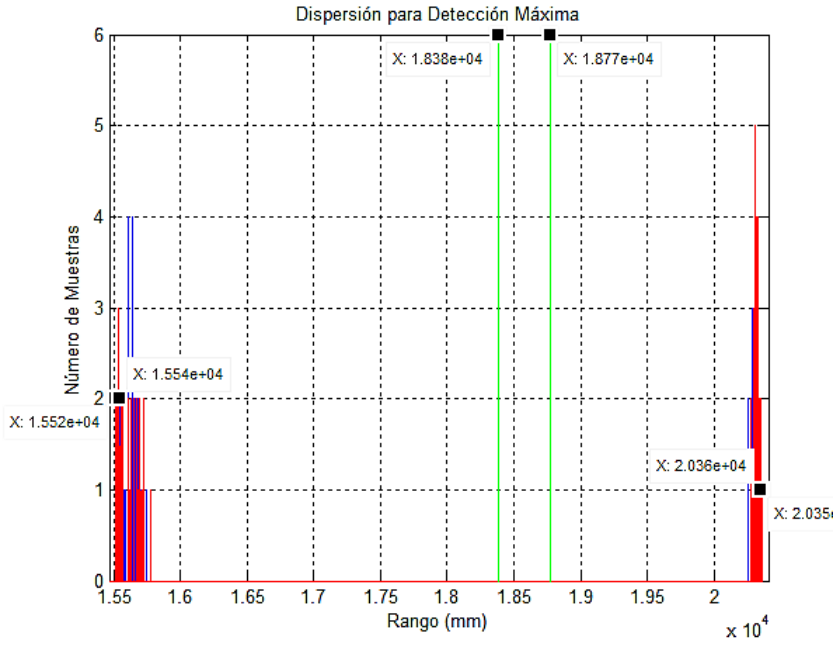


Figura 37. Dispersión para detección máxima para sensor UST20-LX

Para elaborar un mapa 2D requerimos tener definido un rango, el cual presente precisión y exactitud. Por lo tanto, se busca en tiempo real un rango máximo al cual le corresponde valores

certeros para las condiciones establecidas. Es así que se determina que para un valor máximo de 12000mm existe una variabilidad mínima.

Es así como, para establecer la distancia de detección máxima las condiciones del rango máximo fueron las siguientes:

- Localización del sensor: 10 cm de altura desde el piso
- Localización del objetivo: 12000mm del sensor
- Tiempo de muestreo: 6 (s)
- Periodo de muestreo: 30ms
- Número de Lecturas: 100
- Nivel de iluminación: 20 lx

Los resultados en la (Figura 38) muestran menor variabilidad existente para el rango de detección establecido. La línea de color azul corresponde a las mediciones realizadas para el objetivo de color negro y en la cual se determinó un valor medio de 11857 mm, valor máximo de 12101 mm y mínimo de 11698 mm. Por otro lado, el objetivo de color blanco al cual corresponde la línea de color rojo se le determino un valor medio de 11974 mm, valor máximo de 12139 mm y mínimo de 11678 mm.

El rango de detección establecido muestra un buen grado de precisión cercano a ± 30cm para ambos casos, lo cual es mayor a lo establecido de 40 mm. Por otro lado, la exactitud presenta valores menores a 15 cm. Si bien estos valores no son los referenciales, para la aplicación de un UGV pequeño un error de máximo 30 cm es aceptable.

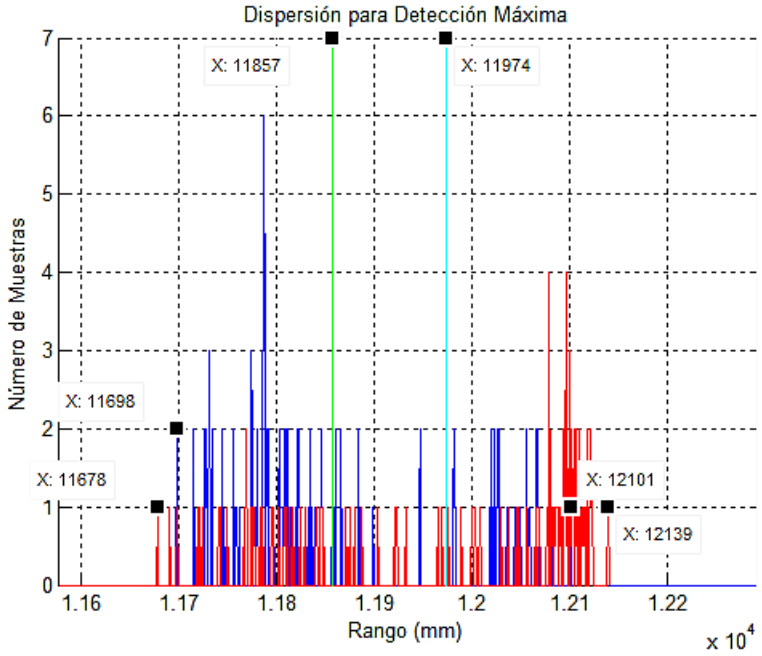


Figura 38. Dispersión para detección máxima establecida para sensor UST20-LX

Para este sistema se define que el rango de detección se extiende desde un valor nominal mínimo de 60 mm y un valor nominal máximo de 12000 mm. Estos valores dependen de forma directa por color de la superficie objetivo. Por lo tanto, a continuación, se analiza los efectos ante el color.

3.2.2. Dispersión acorde al color

Para comparar el efecto que tienen los colores del objetivo hay que analizar la dispersión de forma continua para una única distancia objetivo. Ya que solo se requiere ubicar la máxima dispersión, se grafica la dispersión por medio de una función de suavizado. La distancia escogida como objetivo es de 8000mm ya que es un valor dentro del rango y cercano al límite donde se registra mayor dispersión.

Es así que, para establecer la dispersión acorde al color las condiciones fueron las siguientes:

- Localización del sensor: 10 cm de altura desde el piso

- Localización del objetivo: 8000mm del sensor
- Tiempo de muestreo: 6 (s)
- Periodo de muestreo: 30ms
- Número de Lecturas: 100
- Nivel de iluminación: 20 lx
- Colores: Blanco, Amarillo, Rojo, Negro, Aluminio Platinado

Los resultados en la (Figura 39) muestran la disposición acorde a cada color fijado, para lo cual se determina que el color cuya dispersión se encuentra con la mayor exactitud y precisión es el color Amarillo. Debido a que su valor de precisión se encuentra entre ±30mm y su valor de precisión media es de 3.2 mm. Por otro lado, se demostró que el color Negro presenta la peor precisión y exactitud, y se debe evitar utilizar ese tipo de superficies. Adicionalmente, es factible utilizar colores blanco y rojo los cuales presentaron características aceptables.

Es necesario tener en cuenta al momento de diseñar el entorno que el color es una variable que interviene de forma directa tal y como se verificó. Previo a dimensionar el espacio de trabajo en necesario determinar de forma concreta la distancia con mejor respuesta para lo cual lo siguiente es utilizar la intensidad de recepción en función de la distancia.

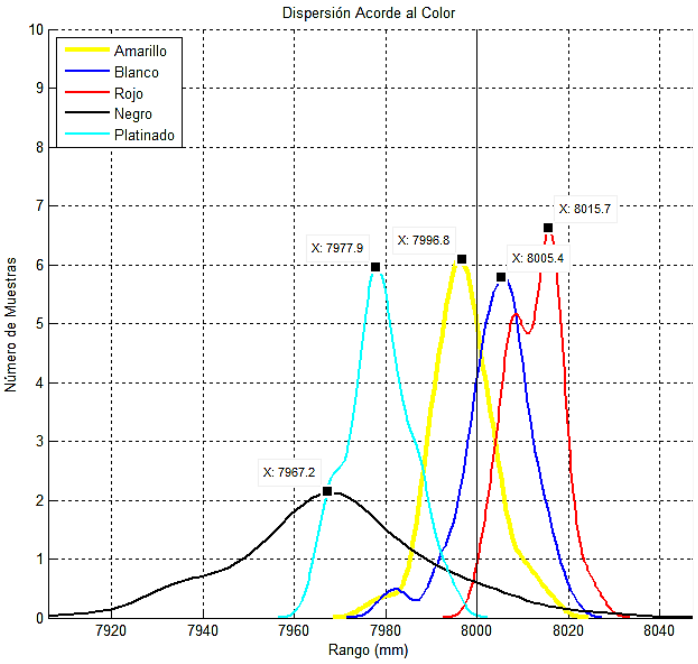


Figura 39. Dispersión acorde al color en 8000mm

3.2.3. Intensidad luminosa acorde al ángulo

La intensidad luminosa está definida como "la magnitud que revela el flujo de luz que emite una fuente en un sentido determinado por cada unidad de ángulo sólido" (Pérez Porto & Merino, 2018). Por ende, la intensidad de recepción para este sensor define como el flujo de luz que retorna de forma directa, y que se relaciona directamente con la medición

Para establecer la dispersión de la intensidad luminosa acorde al ángulo las condiciones fueron las siguientes:

- Localización del sensor: 10 cm de altura desde el piso
- Localización del objetivo: 4000mm del sensor.
- Ángulo del objetivo: 0°, 45° y 60° del plano perpendicular al sensor.
- Tiempo de muestreo: 6 (s)
- Periodo de muestreo: 30ms

- Número de Lecturas: 100
- Nivel de iluminación: 19 lx
- Color Objetivo: Amarillo

En la (Figura 40) se muestra la dispersión de distancias para una distancia fija, pero con los ángulos planteados. La respuesta particular que se obtiene es que para un valor de inclinación de 45° el sensor responde de mejor manera. La variación de ángulo no influye mayormente a la precisión de la distancia. Consecuentemente en la (Figura 41) se muestra la intensidad luminosa acorde a cada ángulo fijado, para lo cual se determina que el ángulo cuya dispersión se encuentra con el mayor valor es la referente al ángulo cero y esto verifica que al modificar el ángulo la magnitud del láser se desvía disminuyendo la recepción. Adicionalmente, se encuentra una relación directa entre la distancia y la intensidad luminosa.

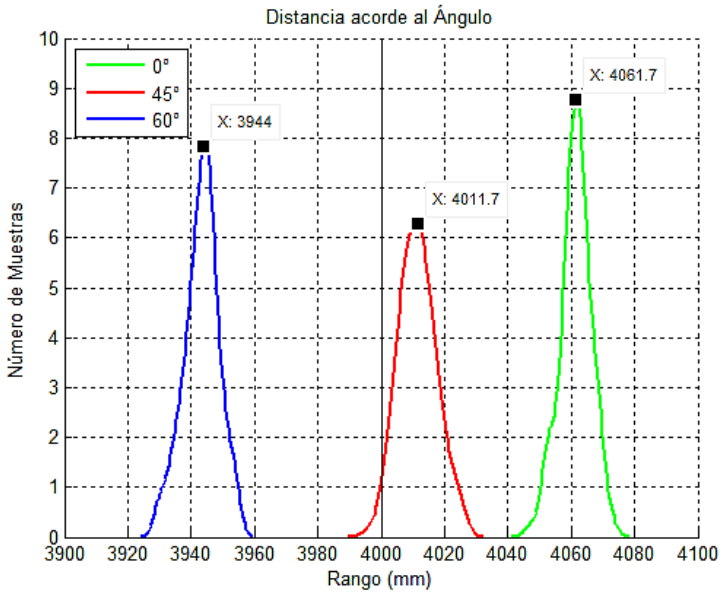


Figura 40. Dispersión de distancias acorde al ángulo del objetivo

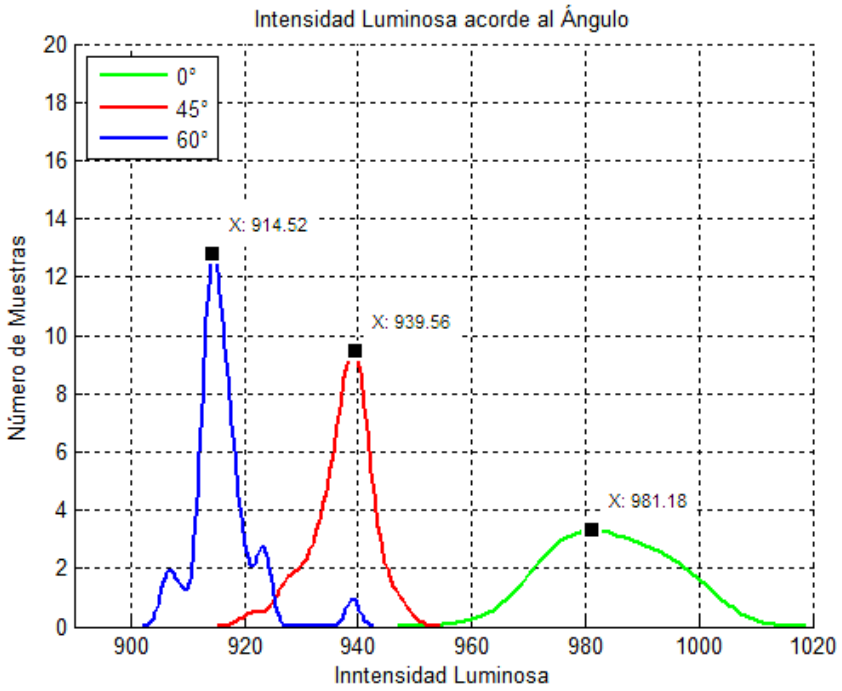


Figura 41. Dispersión de intensidad luminosa acorde al ángulo del objetivo

Mayor intensidad no garantiza necesariamente que el valor alcance exactitud y precisión, pero tampoco influye en errores significativos al momento de determinar las distancias.

3.2.4. Tabulación de parámetros del sensor UST20-LX

Las especificaciones técnicas del dispositivo son válidas en tanto se encuentre el sensor en las condiciones óptimas que el proveedor anticipa. El dispositivo en puntos anteriores se sometió a pruebas para garantizar las características de este, por lo tanto, se redefinen ciertos parámetros y se definen nuevos parámetros que en proveedor no anticipa. Estos nuevos parámetros son referenciales para este dispositivo en particular, se recomienda que estos valores no deben sobrepasarse, porque los errores generados podrían afectar a la etapa de planificación de trayectorias. En caso de que se requiera mayor certeza se debería realizar pruebas adicionales y recurrentes.

Tabla 6
Parámetros particulares del sensor Hokuyo UST-20LX

Rango de detección	0.06m a 12m - Máxima detección: 20 m
Tasa de muestreo	30ms
Exactitud	< 60mm
Precisión	30 mm < σ > 40cm
Angulo de escaneo	270°
Número de lecturas	1080
Offset del ángulo	90°
Colores de los objetivos	Amarillo o Blanco
Angulo de inclinación de objetivos	0° a 45° al eje perpendicular del sensor
Tiempo de estabilización	1 hora

Bajo estas consideraciones el sensor puede ser colocado a bordo de un UGV en situaciones controladas desarrolladas acorde a lo presentado. Una vez que el sensor se encuentra caracterizado se presentan las condiciones de trabajo.

3.3. Condiciones de trabajo para el LIDAR 2D

3.3.1. Locación e Iluminación

Varias características ya establecidas sirven como base para definir la locación. La principal condición es que el entorno sea de color amarrillo a blanco. Poseer dimensiones que no superen los 12 metros. Para ello se considera una azotea cubierta la cual a la hora de realización de la muestra presento el nivel de iluminación mostrado en la (Figura 42). Según (Costella & Rodríguez, 2017) el nivel de iluminación promedio de la luz solar en exterior es de 60000 lx por lo tanto se debe controlar que la luz solar no sea intensa. Las paredes se acogen a las consideraciones establecidas y son paredes amarillas crema. El espacio es identificado con el

sensor como se muestra en la (Figura 43) y las condiciones en las que se realizó la identificación del perfil de la locación fueron las siguientes:

- Dimensión de locación: Ancho 5.7m y Largo 13 m (Valores Máximos)
- Localización del sensor: 3.15 m del lado izquierdo y 0 m del lado frontal
- Altura del sensor: 10 cm de altura desde el piso
- Tiempo de muestreo: 6 (s)
- Periodo de muestreo: 30ms
- Número de Lecturas: 100
- Nivel de iluminación: 1969 lx
- Hora de realización: 17:00 a 18:00
- Color Pared: Amarillo - Crema

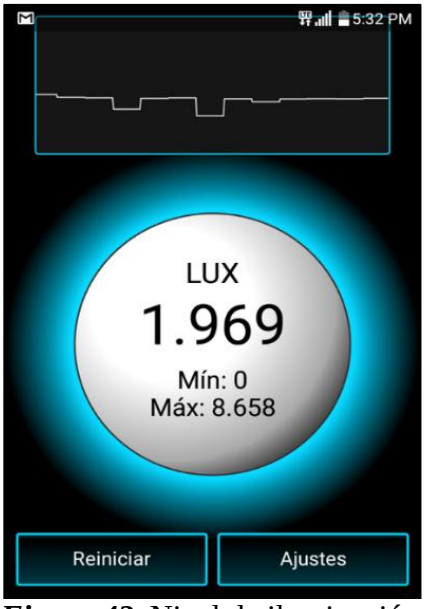


Figura 42. Nivel de iluminación para la locación

Existen otras condiciones adicionales que la locación debe prestar para la elaboración de mapas 2D. Una de las cuales es que existan lugares que no se alcance a observar de forma directa, ya que el objetivo es que el mapa alcance dichas zonas. Otra condición es que el lugar preste el suficiente espacio para desplazarse sin impedimento visual u obstáculo, ya que se requiere verificar la totalidad del perfil de forma inicial.

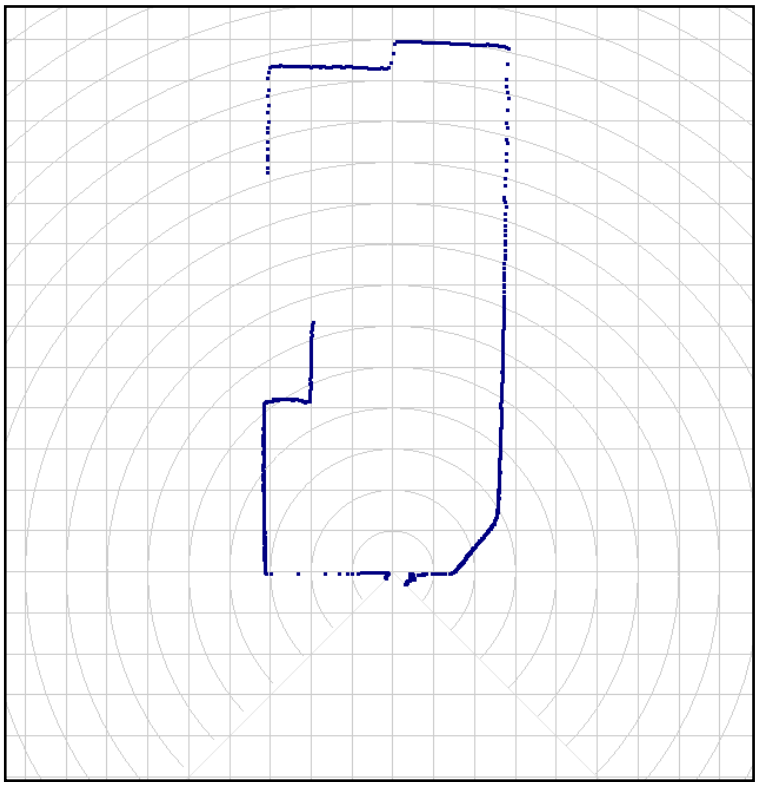


Figura 43. Perfil de la locación en software URG-viewer

Una restricción adicional de la locación es que el ambiente no debe poseer espejos los cuales debido a su coeficiente de reflexión generan distorsión es las mediciones y por ende falsos positivos en las dimensiones, lo cual dificulta el mapeo como se muestra en la (Figura 44). Esta prueba fue realizada por medio de la herramienta hector-mapping en el sistema operativo ROS (ROS.org, 2018).

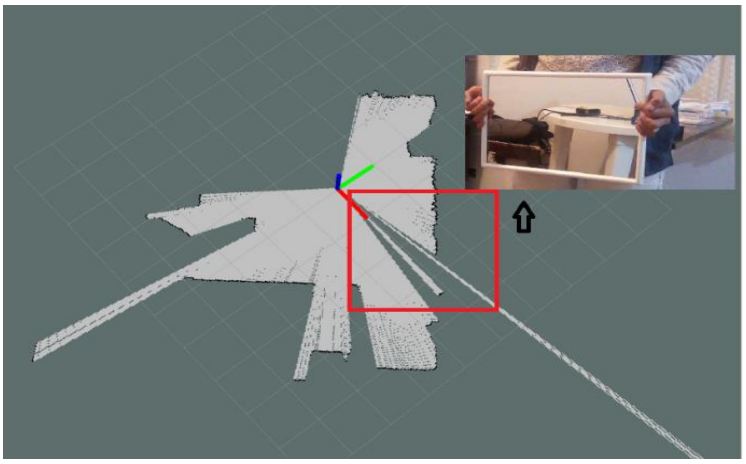


Figura 44. Reflexión ante espejos Hector-Mapping

3.3.2. Posición del sensor UST-20lx a bordo

La colocación del sensor debe garantizar que el vehículo no intervenga en la medición, por lo tanto, hay que considerar el ángulo de visión posea una cobertura de 270° como se ha establecido. Esto implica que el ángulo fuera de cobertura es de 90 grados y como se había definido existe un offset de 90°. Por lo tanto, la pose del sensor debe considerar lo siguiente; estar adecuada en el eje longitudinal central del vehículo como se muestra en la. Y la distancia entre el lado frontal del vehículo debe ser de al menos la mitad del ancho del vehículo, debido a la correspondencia geométrica entre lados de triángulos rectángulos, como se muestra en la (Figura 45). Esto depende de la altura del sensor y si a aquella altura parte del vehículo queda bajo se puede aproximar el sensor al lado frontal.

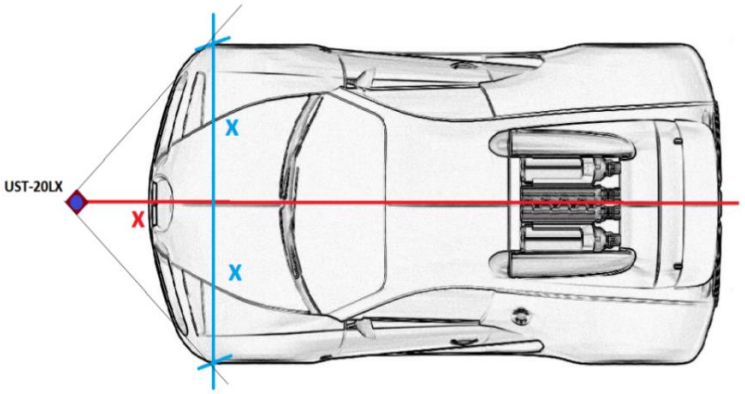


Figura 45. Posición del sensor UST-20LX
a borde de un UGV

La inclinación del sensor se establece sea horizontal ya que el desplazamiento de un UGV es de igual manera. Adicionalmente de esta manera se aprovecha el máximo rango y cobertura.

Para el desarrollo del sistema para la generación de mapas 2D se utilizó un vehículo tele operado de pequeña dimensión pero que permite desarrollar lo propuesto. Este vehículo mostrado en la (Figura 46) tiene la capacidad de desplazarse en cualquier dirección lo cual permite realizar mapas de mayor capacidad. Se colocó el sensor de forma central y en la parte frontal del vehículo, ya que no posee coraza que impida la exploración directa. Es necesario indicar que este vehículo solo se lo ocupa para la interpretación y prueba del sistema para generación de mapas ya que por su dimensión no soporta la carga que conlleva el ordenador portátil.

Figura 46. Vehículo para prueba en la
generación de mapas 2D

3.3.3. Requerimientos para ordenador en la generación de mapas 2D

Ya se identificó previamente ordenadores portátiles, uno de los cuales pose el sistema operativo Ubuntu 16.04. Este sistema operativo permite desarrollar e integrar programas de código abierto en; c, cpp, Python, entre otros, lo cual abre las posibilidades en el desarrollo del sistema UGV. Es así que de forma general las características para el ordenador portátil son:

- Sistema operativo Ubuntu 16.04 en 64 bits
- Conectividad Ethernet 100BASE-TX
- Memoria RAM de al menos 4 GB
- Tarjeta gráfica dedicada

Al tener definidas las condiciones para el uso del sensor láser, se procede a realizar la adquisición de la profundidad en la locación, esta vez de forma directa en Ubuntu.

3.4. Adquisición de Profundidad 2D

Para la adquisición de los valores provistos por el sensor en el ordenador establecido es necesario instalar el paquete de controladores los controladores se encuentran desarrollados en lenguaje C (URG Library document, 2018), para manipular los valores se determina necesario emplear ese lenguaje de programación y consecuentemente el IDE de Eclipse.

Para la adquisición de los valores del sensor es necesario tener un patrón de referencia y un valor a correlacionar. Por lo tanto, se establece que el patrón se realizará en milímetros desde Urg Viewer, permitiendo una comparativa directa. Posterior a tener establecida la conexión con las respectivas direcciones IP, se verifica la conectividad del sensor. Para visualizar el perfil de datos obtenidos por medio del controlador se emplea el complemento SDL como en la (Figura 47) para lo cual se correlaciona con los resultados obtenidos en URG.Viewer mostrados en la (Figura 48).

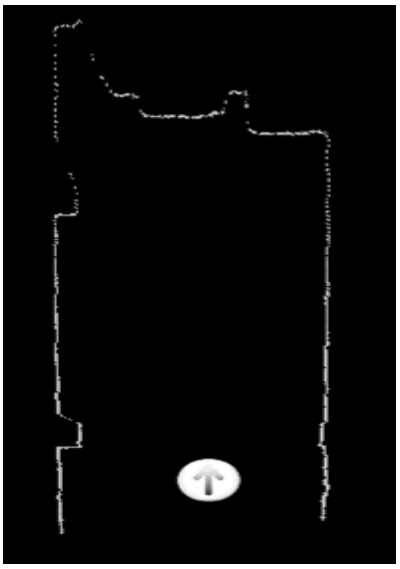


Figura 47. Perfil de la locación obtenida por SDL

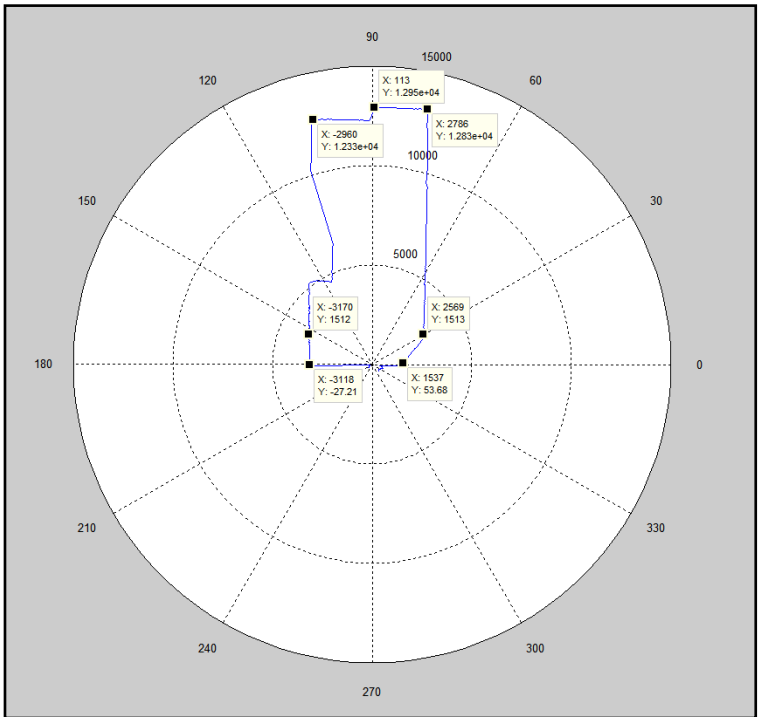


Figura 48. Perfil de la locación obtenido por URG-Viewer

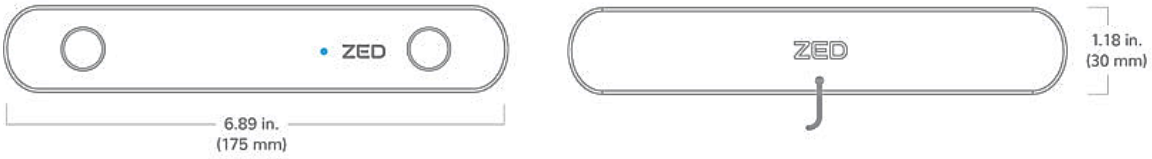
Una vez se obtiene el perfil de profundidad se puede integrar y adaptar algoritmos como SLAM revisado anteriormente.

3.5. Especificaciones técnicas de la cámara estereoscópica ZED

La cámara estereoscópica elegida para realizar este trabajo es la cámara ZED, desarrollada por Stereolabs. Se adquirió esta cámara debido a sus características mostradas en la (Tabla 7), y también porque es capaz de adquirir video 3D de alta definición, así como lograr una percepción de profundidad muy buena (Navas Flores, Suasnavas, & Ramiro, 2018), la (Figura 49) muestra la apariencia de la cámara estéreo.

Figura 49. Cámara estereoscópica ZED
Fuente: (Navas Flores, Suasnavas, & Ramiro, 2018)

La cámara tiene un aspecto muy elegante, y es fácil de colocar debido a su pequeño tamaño y a su forma. Estos aspectos para este trabajo en particular tienen importancia debido a que será montado sobre el vehículo y es necesario que se pueda ubicar en un lugar adecuado y que no exista ningún problema cuando se exponga al movimiento del vehículo, las dimensiones de la cámara se ilustran en la (Figura 50)



a) b)

Figura 50. Dimensiones de la cámara estereoscópica ZED
a) vista frontal b) vista trasera
Fuente: (Navas Flores, Suasnavas, & Ramiro, 2018)

Las especificaciones técnicas dadas por el fabricante en su página web (Stereolabs, 2018) se muestran en la (Tabla 7).

Tabla 7
Especificaciones técnicas de la cámara estereoscópica ZED

Ítem	Detalles		
Características	Captura de video 3D de alta resolución y alta velocidad de captura de imágenes. Percepción de profundidad en interiores y exteriores de hasta 20 m. Seguimiento de Posición y Orientación Mapeo Espacial.		
Video	**Resolución**	**Cuadros por segundo**	**Resolución de salida**
	2.2K	15	4416x1242
	1080p	30	3840x1080
	720p	60	2560x720
	WVGA	100	1344x376
Profundidad	Formato	Rango	Línea base
	32 bits	0.5 - 20 m	120 mm
Movimiento	Precisión de la Pose	Tecnología	Frecuencia
	Posición: +/- 1mm Orientación: .1°	Odometría visual basada en la profundidad en tiempo real y SLAM	100 Hz
Lente	Lente doble de cristal con distorsión reducida Campo de visión: 110 ° *f* / 2.0 apertura		
Sensor	**Tamaño del sensor**	**Controles de cámara**	**Formato del sensor**
	1/3 ". Sensor de iluminación trasera con alta sensibilidad a la luz baja	Ajuste la resolución, la velocidad de fotogramas, la exposición, el brillo, el contraste, la saturación y el balance de blancos	Formato nativo 16: 9 para un mayor campo de visión horizontal
Conectividad	**Conector**		**Potencia**
	Puerto USB 3.0 con cable integrado de 1.5 m		Potencia a través de USB 5V / 380mA
	Opciones de montaje		**Temperatura de funcionamiento**
	montaje de rosca UNC de 1/4 "-20		0 ° C a + 45 ° C (32 ° F a 113 ° F)

Para hacer uso de las funciones de la cámara es necesario contar con el SDK, en el cual vienen incluidas las librerías para desarrollo que nos permiten usar las funcionalidades de la cámara. Como se ha descrito anteriormente la cámara necesita de una GPU NVIDIA y una versión compatible de la plataforma de computación en paralelo CUDA.

El contenido de las librerías que vienen incluidas en el kit de desarrollo de software (SDK) están desarrolladas en C++. Es así que funciones y clases del kit son explicadas de forma general en la página web de ZED, en la sección de desarrolladores (Stereolabs, 2018).

Para este trabajo se desarrollan los algoritmos en C++, esto debido a los múltiples beneficios que otorga especialmente en su apartado visual compatible con OpenCV. También se puede realizar la integración con el lenguaje de programación Python, existe un kit desarrollado por Stereolabs en el cual por medio de *Cython* se integra las funciones directamente de *C++* (Github, 2018). Se verificó el funcionamiento de la SDK en Python y se comparó los resultados en ambos entornos de programación, para luego definir a C++ como el lenguaje de programación más ventajoso para este trabajo.

En Python las clases se definen dentro de una carpeta llamada *pyzed*, en las cuales dependiendo de cada tipo de clase utilizada se colocan las definiciones que son las que ejecutarán el código necesario. En C++ se tiene mayor versatilidad y apertura para cambiar parámetros, adicional se tiene más información disponible, y soporte para los desarrolladores en la página principal de STEREOLABS.

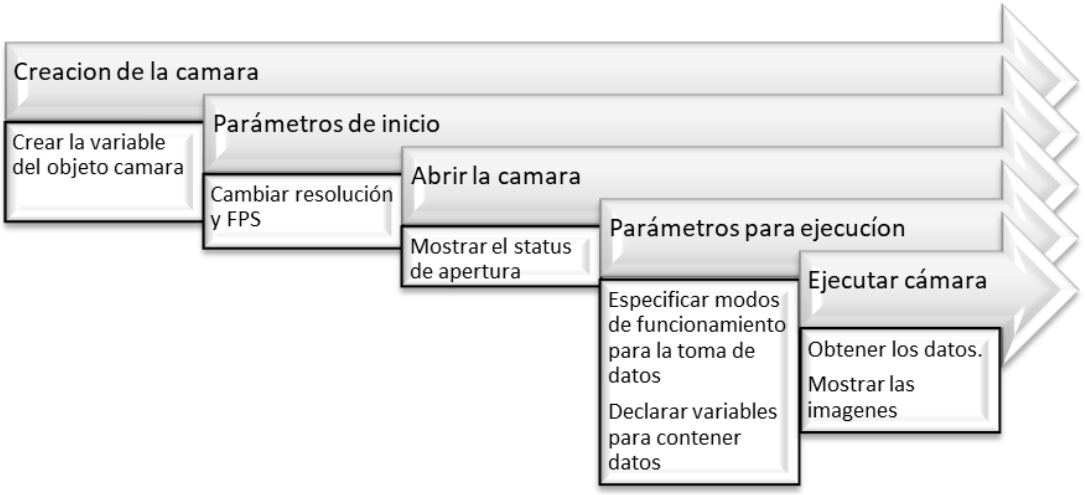
Las principales clases que se utilizará para realizar la estimación de profundidad se muestran en la (Tabla 8). El uso de objetos facilita la manipulación de las clases, cada objeto se nombra con sus métodos, los cuales contienen datos para gestionar las imágenes y los parámetros. Es necesario asignar una variable para cada objeto junto con su método o acción, esto permitirá acceder a datos necesarios o ejecutar acciones requeridas. Por ejemplo, si se quisiera definir la resolución de una imagen VGA, se asignaría el atributo VGA a la clase RESOLUTION, generando la acción llamada: RESOLUTION_*VGA*.

Tabla 8
Definición de clase

CLASES	ACCIONES	
RESOLUTION	RESOLUTION_HD2K RESOLUTION_HD720	RESOLUTION_HD1080 RESOLUTION_VGA
MEASURE	MEASURE_DISPARITY MEASURE_DEPTH MEASURE_CONFIDENCE MEASURE_XYZ	MEASURE_XYZRGBA MEASURE_XYZBGRA MEASURE_XYZARGB MEASURE_XYZABGR
VIEW	VIEW_LEFT VIEW_RIGHT VIEW_LEFT_GRAY VIEW_RIGHT_GRAY VIEW_LEFT_UNRECTIFIED VIEW_RIGHT_UNRECTIFIED	VIEW_SIDE_BY_SIDE VIEW_DEPTH VIEW_CONFIDENCE VIEW_NORMALS VIEW_DEPTH_RIGHT VIEW_NORMALS_RIGHT
DEPTH_MODE	DEPTH_MODE_NONE DEPTH_MODE_PERFORMANCE DEPTH_MODE_QUALITY	DEPTH_MODE_MEDIUM
SENSING_MODE	SENSING_MODE_STANDARD SENSING_MODE_FILL	
UNIT	UNIT_MILLIMETER UNIT_METER UNIT_FOOT	UNIT_CENTIMETER UNIT_INCH
COORDINATE_SYSTEM	COORDINATE_SYSTEM_IMAGE COORDINATE_SYSTEM_LEFT_HANDED_Y_UP COORDINATE_SYSTEM_RIGHT_HANDED_Y_UP COORDINATE_SYSTEM_RIGHT_HANDED_Z_UP COORDINATE_SYSTEM_LEFT_HANDED_Z_UP	

3.6. Adquisición de la profundidad

Para adquirir los datos de profundidad o realizar cualquier método en general lo primero que tenemos que hacer es ejecutar la cámara, y así recibir los datos. Según los parámetros que necesitemos. Por lo tanto, es necesario llamar a las funciones con las que se podrá utilizar el SDK y asignarles una variable de la cual se pueda obtener los atributos de la clase. Para ejecutar la cámara los pasos son los siguientes:



.

Figura 50. Proceso de ejecución de estéreo cámara ZED.

Para la obtención de parámetros específicos, se debe de cambiar los parámetros de inicio, con el fin de obtener un mejor resultado según el trabajo que se realizará con la cámara y en función al desempeño que se desee. Las siguientes clases pertenecientes a esta etapa inicial y se han denominado como:

- RESOLUTION
- DEPTH_MODE
- SENSING_MODE
- UNIT

Para obtener los datos de profundidad, mientras la cámara se está ejecutando es necesario declarar variables que contengan los datos de imagen adquiridos, para lo cual se declara las siguientes variables.

- image = core.Mat()
- depth = core.Mat()
- point_cloud = core.Mat()

Estas variables serán cargadas con los valores de imagen, profundidad y la nube de puntos respectivamente. Mientras se encuentra el programa en modo de ejecución, colocamos en los datos en las respectivas variables usando los siguientes comandos:

- cam.retrieve_image(mat, sl.VIEW.VIEW_LEFT)
- cam.retrieve_depth(mat, sl.VIEW.VIEW_DEPTH)
- cam.retrieve_point_cloud(mat, sl.VIEW.VIEW_MEASURE_XYZRGBA)

Con la variable *image* se puede mostrar la imagen que está tomando la cámara en ese momento al elegir la opción *VIEW_LEFT* mostraremos la imagen que está capturando el lente izquierdo de la cámara, si en su lugar lo cambiamos por *VIEW_DEPTH* visualizaremos el mapa de profundidad. Para mostrar en pantalla simplemente se requiere del uso de *OpenCV*, por lo cual para mostrar será necesario colocar el comando *cv2.imshow* las imágenes mostradas luego de esta ejecución en la (Figura 51) a) se muestra un entrono interior, mientras que en la (Figura 51) b) se muestra la respuesta expresada por medio del mapa de profundidades resultante.

a) b)

Figura 51. Proceso de ejecución de estéreo cámara ZED
a) imagen de lente izquierdo b) mapa de profundidad

Finalmente se obtiene los datos de las distancias de la profundidad y se decide alinear el valor de las distancias obtenidas hacia el centro de la pantalla y se usa el cálculo de la distancia euclidiana como se sugiere (Sun, Li, Kang, & Shum, 2005), (Lecumberry, 2005).

3.7. Caracterización de los parámetros de la cámara para la obtención de profundidad.

Es necesario conocer las diferentes respuestas de profundidad que otorga la cámara al cambiar sus parámetros, para ello se ha expuesto a la cámara a diferentes condiciones externas para que realice la estimación de profundidad. En las pruebas realizadas se tomó en cuenta el entorno y los colores en los objetos ubicados en el centro de la pantalla a una distancia específica.

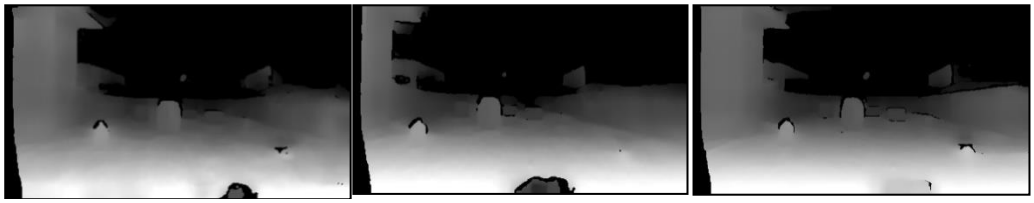
Como entorno para realizar esta prueba se ha elegido un entorno abierto externo, en el cual se ha colocado diferentes objetos a diferentes distancias y se ha colocado objetivos de color rojo colgados en el aire para tener una base más clara al momento de la comparación, el entorno con el cual se trabajó se muestra en la (Figura 52).

Figura 52. Entorno exterior de prueba para la cámara en la mañana

Bajo el escenario exterior se realizó variaciones en la resolución que entrega la cámara (VGA, HD720 y HD1080) y para cada resolución se cambió la calidad de la percepción de profundidad entre (PERFORMANCE, MEDIUM y QUALITY), los resultados son mostrados a continuación:

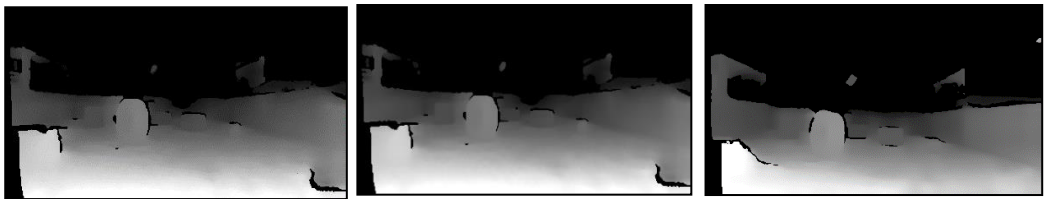
a) **b)** **c)**

Figura 53. Resultados mapas de profundidad, Resolución VGA
a) Modo Performance b) Modo Medium c) Modo Quality



a) **b)** **c)**

Figura 54. Resultados mapas de profundidad, Resolución HD720
a) Modo Performance b) Modo Medium c) Modo Quality



a) **b)** **c)**

Figura 55. Resultados mapas de profundidad, Resolución HD1080
a) Modo Performance b) Modo Medium c) Modo Quality

De los resultados presentados, se puede notar que en el modo QUALITY los obstáculos toman una forma más clara y cercana a la real, y el modo PERFORMANCE no cambia mucho del modo Medium. En cuanto al cambio de resolución hace que se vean mejor los detalles de la imagen y se muestre todo a detalle, sin embargo, se decide utilizar la configuración mostrada en la (Figura 53) a), para ahorrar capacidad de cómputo y poder trabajar a una cantidad mayor de capturas por

segundo (FPS), esto se debe a que cada configuración de resolución tiene un máximo de FPS, VGA trabaja a 100 FPS , HD720 trabaja a 60 FPS y 1080 trabaja a 30 FPS.0

Para realizar las siguientes pruebas también se plantea que exista una variación de iluminación, los cambios de la luz pueden generar diferentes respuestas debido a la acción de la reflexión (Matheis, 2016). Se crea un escenario de pruebas para hacer las pruebas más específicas, como se muestra en la (

Figura ***56***), y de esta manera obtener mejores resultados. La prueba de colores es realizada en el atardecer como se muestra en la (Figura 57) a). Esto con el fin de mostrar la respuesta de la cámara y en si la percepción ante variaciones de color, sus resultados se muestran en la (Figura 58). A partir de esta prueba se comenzará a probar usando la resolución VGA con calidad PERFORMANCE.

Figura 56. Escenario preparado para las pruebas

a) b)

Figura 57. Entorno de prueba para mediciones de distancia en el atardecer a) imagen de lente izquierdo b) mapa de profundidad

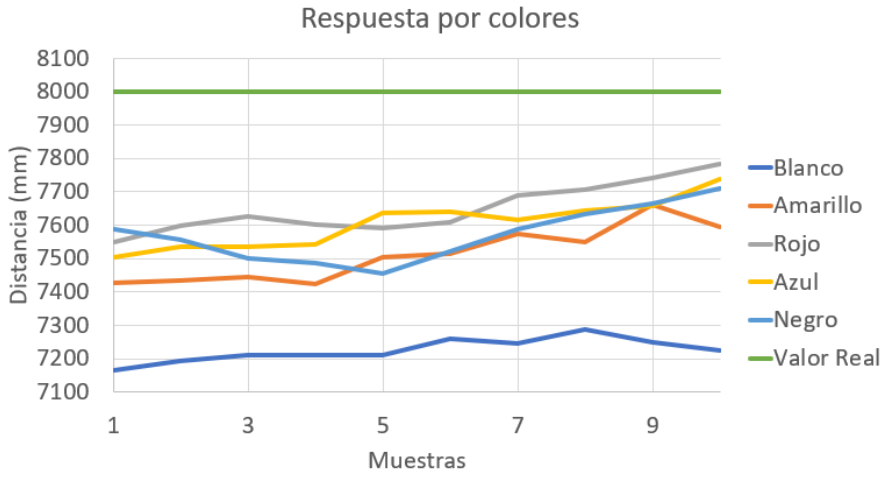


Figura 58. Resultado de prueba con diferentes colores.

Luego de realizar las pruebas con colores se elije al color amarillo debido a que tiene una respuesta aceptable y a la vez estable. Para las siguientes pruebas se realiza un cambio de distancia, ahora el valor real de distancia a alcanzar es de 5m. Luego se procede a realizar las pruebas variando la calidad de percepción de profundidad, estos resultados se muestran en la (Figura 59). Finalmente se usa la calidad QUALITY y se varía la resolucion para intentar alcanzar los parametros mas altos, y su resultado se muestra en la (Figura 60).

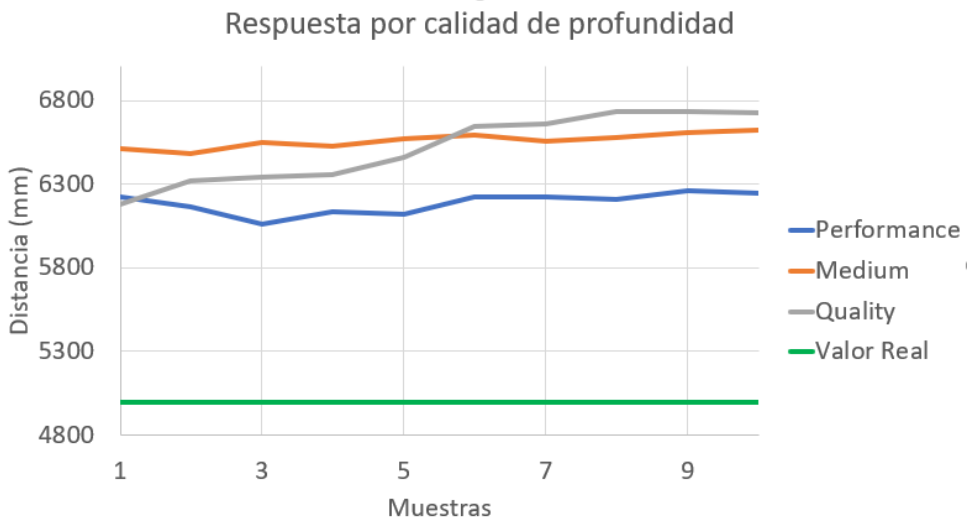


Figura 59. Resultado de prueba con color amarillo y diferentes calidades de profundidad

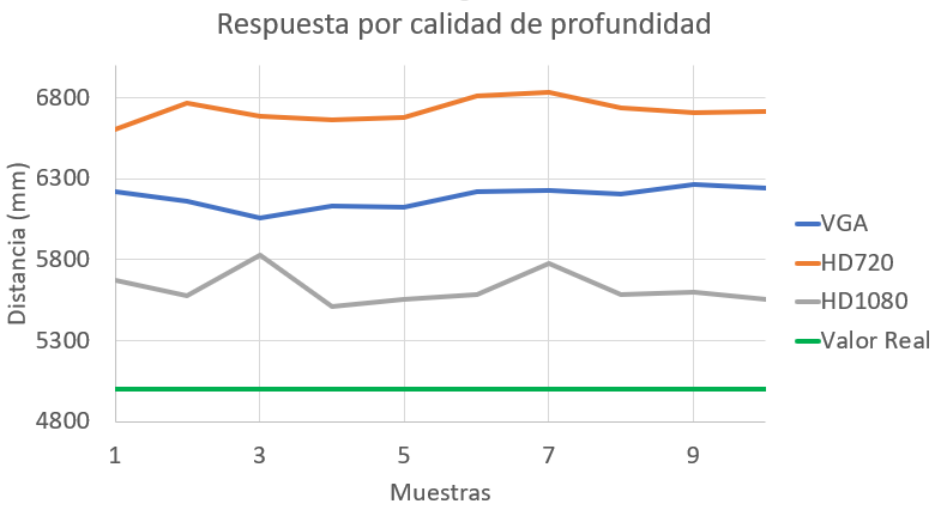


Figura 60. Resultado de prueba con color amarillo, calidad QUALITY y variando la resolución.

Para terminar, se realiza una prueba en la noche para lo cual también nos acercamos aún más hacia la pared con la cual se ha estado realizando las mediciones, en la (Figura 61) se muestra los resultados para esta prueba.

a) b)

Figura 61. Entorno de prueba para mediciones de distancia en la noche a) imagen de lente izquierdo b) mapa de profundidad

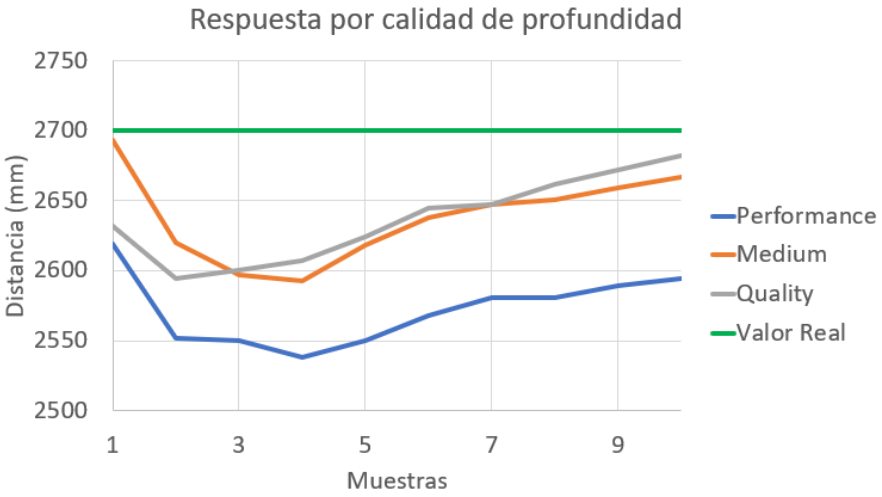


Figura 62. Resultado de prueba a 2.7 m
y diferentes calidades de profundidad

Al reducir la distancia se tiene una mejora notable, como se muestra en la (Figura 62), en comparación con los resultados expresados a largas distancias. Por este motivo se plantea simular un obstaculo previo a la pared que se tiene detrás, como ilustra la (Figura 63), ademas tambien se hará pruebas en las cuales el obstaculo se encuntre a 45 grados del frente, es decir desplazado hacia un lado en la (Figura 64), esta prueba se lleva a cabo usando la resolucion VGA con el modo de profundidad PERFORMANCE y se muestra los resultados de medicion de profundidad en la (Figura 65).

Figura 63. Prueba con obstaculo en frente a 1.1 m de distancia en la noche
a) imagen de lente izquierdo b) mapa de profundidad

Figura 64. Prueba con obstaculo a 1.1 m desplazado a 45 grados en la noche

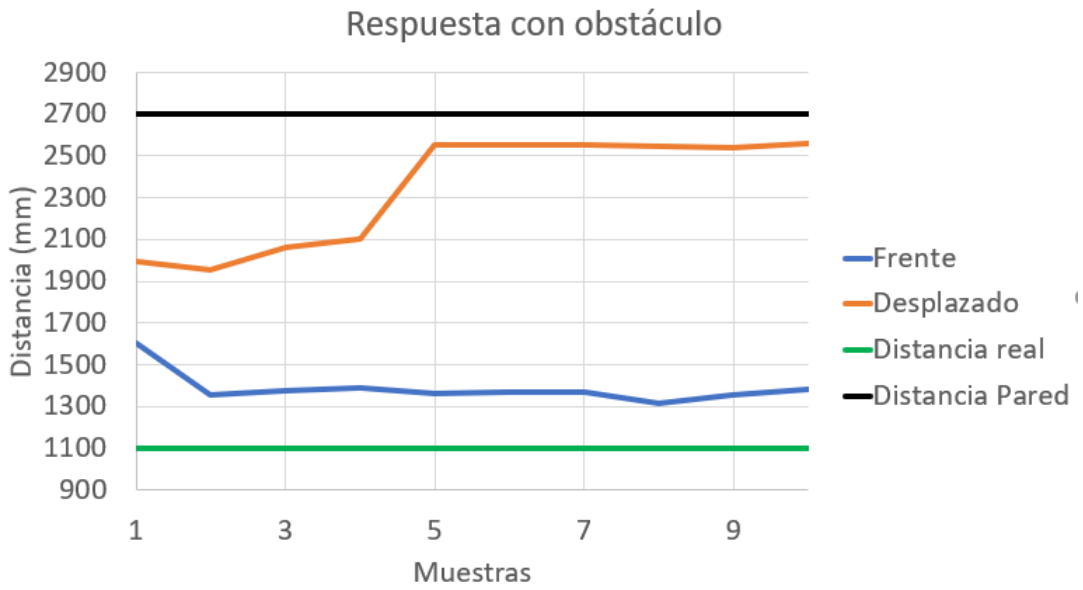


Figura 65 . Resultado de prueba con obstaculo presente

Este resultado muestra que cuando existe un obstaculo en frente, lo detecta como algo principal y pese a que detrás se encuantra la pared con una mayor area, la profundidad encontrada por el obstaculo hacer que varíe considerablemente la distancia percibida, por lo cual en caso de existir un obstaculo el vehiculo reaccionaria deteniendose. En el caso de que el obstaculo aun no

se encuentra en el centro genera una pequeña reaccion que haría que el sistema se ponga alerta y en caso de seguir avanzando de igual manera llegaria a detenerse.

3.8. Especificaciones técnicas del UGV

El vehículo tiene la finalidad de ser un UGV, por lo tanto, se requiere que sea autónomo, y para el cual necesitamos que todo el sistema este montado sobre el vehículo y para definir su funcionamiento es importante basarse en las limitaciones que tienen los equipos, actuadores y sensores, en esta sección se muestra las especificaciones técnicas, las condiciones de uso y el alcance tecnológico al que se puede llegar.

3.8.1. Computador

El dispositivo que está montado en el vehículo y que realizara las funciones de cómputo y procesamiento, es el computador “acer, Aspire 4752 ms2347”, en la (

Figura ***66***) se muestra el aspecto de la computadora y en la (Tabla 9)**Error! Reference source not found.** se muestra las características técnicas.

Figura 66. Computador acer Aspire 4752
Fuente: (Aspire, 2018)

Tabla 9
Especificaciones técnicas de computador "Acer, Aspire 4752"

Características externas	Puerto Ethernet (RJ-45) Puerto para pantalla (VGA y HDMI) Dos puertos USB 2.0 y Un puerto USB 3.0 Jack de audio Lector óptico
Sistema operativo	Linux, Ubuntu 16.04.4 LTS 64 bits
Procesador	Intel® Core i7 2670QM 2,20GHz x 8
Memoria	3,7 GB
Disco	263,4 GB
Pantalla	14'' HD 1366x768
Gráficos	NVIDIA® GeForce GT 540M/PCIe/SSE2

Fuente: (Aspire, 2018)

3.8.2. Controladores gráficos

Para que funcione adecuadamente el computador con la cámara estereoscópica, es necesario tener los controladores adecuados de NVIDIA, que sean compatible con el GPU y con CUDA. Como se recalca la versión de CUDA tiene que ser soportada por la capacidad computacional del ordenador, y tiene que tener una versión de Kernel compatible.

Existen muchas gamas de GPU, y aunque la computadora cuenta con una esta versión se está quedando cada vez entre las menos potentes del mercado (Aspire, 2018). Si tomásemos en cuenta lo mejor en cuanto al software disponible para nuestra versión de GPU, el software a instalar serían los siguientes según la ayuda que nos presenta NVIDIA en su página:

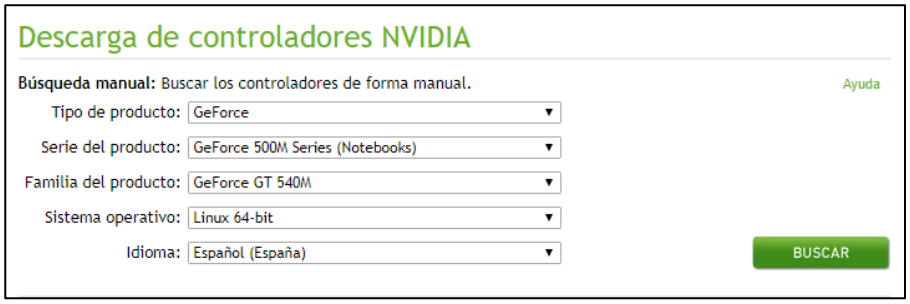


Figura 67. Gestor de descargas para controladores NVIDIA
Fuente: (Aspire, 2018)

Como resultado nos entrega la versión más actual que funcionaría con la computadora que es la siguiente

- Versión: 390.25
- Fecha de publicación: 2018.1.29

A continuación, se necesita instalar el CUDA, de la misma manera se pueda usar la versión más actual del CUDA que se encuentra en línea el CUDA 9.1.85, y de la misma manera se pueda usar el asistente para la descarga CUDA Toolkit 9.1 según nuestro sistema (NVIDIA, 2018) Como se muestra en la (Figura 68).

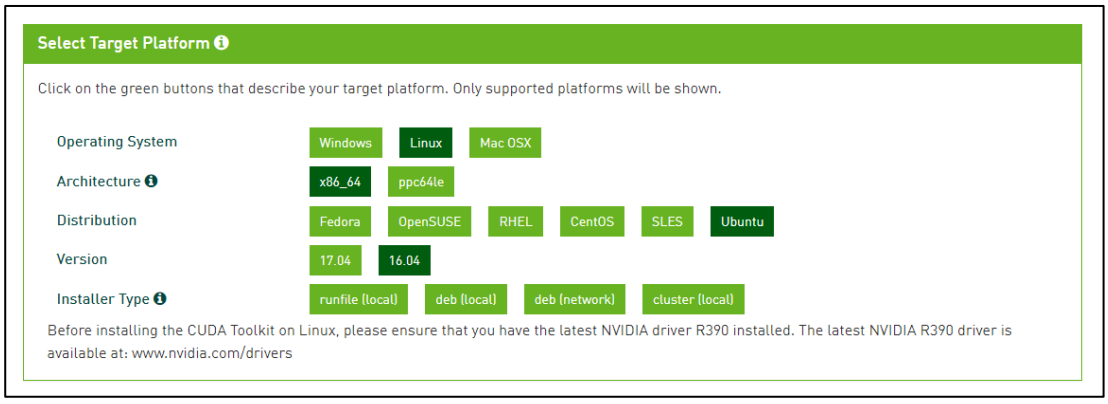


Figura 68. Asistente de descarga CUDA Toolkit
Fuente: (NVIDIA, 2018)

La cámara estereoscópica ZED también tiene una versión de SDK disponible para el CUDA 9.1, se trata del SDK 2.3 que también cuenta con mejoras significativas en comparación a sus versiones anteriores (Stereolabs, 2018).

Pese a que tenemos a disponibilidad el software mostrado anteriormente, no es lo que se debe hacer para la instalación. Existen más factores que analizar a la hora de realizar la instalación, y como se explica en la página de NVIDIA para la instalación de CUDA se debe asegurar todas las especificaciones (NVIDIA, 2018). Por lo tanto, se tiene que cumplir con:

- Verifique que el sistema tenga una GPU compatible con CUDA.
- Verifique que el sistema esté ejecutando una versión compatible de Linux.
- Verifique que el sistema tenga instalado gcc.
- Verifique que el sistema tenga instalados los encabezados del kernel y los paquetes de desarrollo correctos.
- Verificar capacidad de computo

Una vez se ha realizado las pruebas previas a la instalación, se dieron los siguientes resultados:

- 01:00.0 VGA controlador compatible: NVIDIA Corporation GF108M [GeForce GT 540M] (rev a1)
- Gcc: 5.4.0 20160609
- Glibc: 2.23
- kernel: 4.13.0-36-generic
- Capacidad de computo: 2.1

Debido a que nuestra capacidad de computo es inferior a 3, que es lo que se necesita para que funcione el SDK 2.3 de la estéreo cámara (Stereolabs, 2018), se busca una versión anterior, que es la SDK 2.2.0 que necesita una capacidad de computo mayor a 2. Esta versión funciona con CUDA 8 (Stereolabs, 2018), además también encontramos que nuestra versión de kernel es incompatible con CUDA como se muestra en la (Figura 69), por lo cual se procede a actualizar a la versión 4.4.0-16-generic.

Finalmente se procede a instalar como se indicó anteriormente los controladores gráficos de NVIDIA y CUDA, y como resultado final de nuestra instalación, se tiene los siguientes controladores instalados:

- ZED SDK: v2.2.0
- Controlador NVDIA: 384.11
- CUDA: V8.0.61
- kernel: 4.4.0-16-generic
- Gcc: 5.4.0 20160609
- Glibc: 2.23

Distribution	Kernel	GCC	GLIBC	ICC	PGI	XLC	CLANG
x86_64							
RHEL 7.x	3.10	4.8.2	2.17	15 16	16.3+	NO	3.8+
RHEL 6.x	2.6.32	4.4.7	2.12				
CentOS 7.x	3.10	4.8.2	2.17				
CentOS 6.x	2.6.32	4.4.7	2.12				
Fedora 23	4.2.3	5.3.1	2.22				
OpenSUSE 13.2	3.16.6	4.8.3	2.19				
SLES 12	3.12.28	4.8.6	2.19				
SLES 11 SP4	3.0.101	4.3.4	2.11				
Ubuntu 16.04	4.4.0	5.3.1	2.23				
Ubuntu 14.04	3.13	4.8.2	2.19				
ARMv8 (aarch64)							
Ubuntu 14.04	3.13	4.8.2	2.19	NO	NO	NO	NO
POWER8(*)							
RHEL 7.x	3.10	4.8.2	2.17	NO	NO	13.1	NO
Ubuntu 16.04	4.4.0	5.3.1	2.23	NO	NO	13.1	NO

Figura 69. Requerimientos del sistema para CUDA
Fuente: (NVIDIA, 2018)

3.8.3. UGV Ackerman a batería

Se dispone como pieza fundamental de este trabajo un vehículo a baterías, comúnmente usado para diversión infantil, el cual tenía varios cambios del modelo original mostrados en la página oficial del vehículo, que se conoce con su nombre técnico como: “Kinder Elektro Auto MRT

CONCEPT" y la imagen del vehículo utilizado se muestra en la (Figura 70). el detalle de sus especificaciones de fábrica se muestra en la (Tabla 10).

Figura 70. Vehiculo Kinder Elektro Auto MRT CONCEPT, usado en el proyecto

Tabla 10

Especificaciones técnicas del vehículo Kinder Elektro Auto MRT.

Ítem	**Detalles**
Modelo	Concept MRT
Motor	Elektro Motor
Potencia	35W*2
Velocidad Max	2-4 km/h
batería 12V7AH	batería 12V7AH
Duración	Rango de 1-2 horas
Alcance de control remoto	15 m
Transmisión	hacia adelante
tiempo de carga	alrededor de 12 horas
batería	12V7AH
Peso del dispositivo	16 kg
Carga máxima	30 kg
Conector auxiliar MP3	Ja

Fuente: (Crossquads, 2018)

El vehículo es controlado por medio de dos señales de control remoto, la primera para acelerar sea hacia a delante o hacia atrás, la otra señal para cambiar el sentido de movimiento derecha o

izquierda. Estas señales actúan directamente sobre dos motores DC, en el caso de la tracción el motor es ubicado en el eje posterior del vehículo., y en el caso del motor que direcciona al vehículo se tiene un mecanismo Ackerman como se ilustra en la siguiente figura.

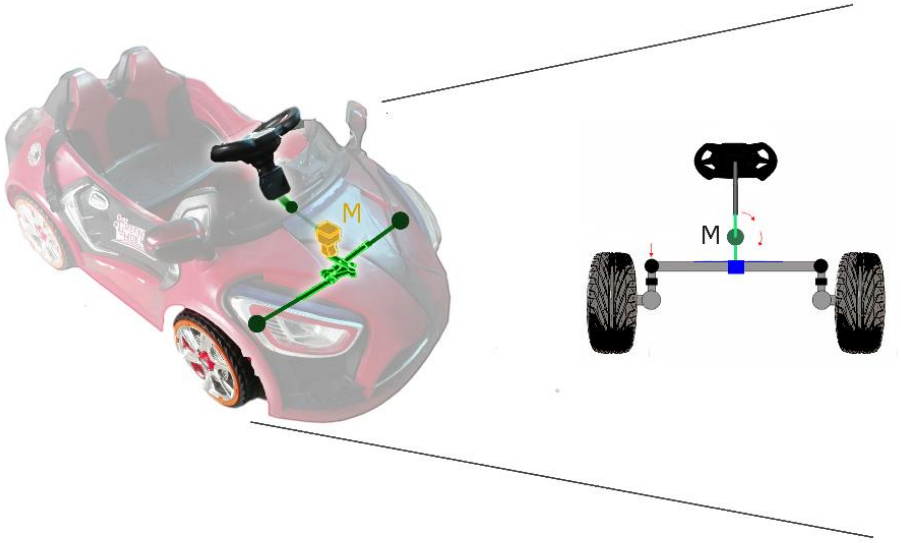


Figura 71. Sistema de direccionamiento del vehículo.

3.8.4. Caja de control y recepción remota

Para usar este vehículo, no se realizará una intervención en el control remoto del mismo, debido a que es un control analógico, y se controla por medio de palancas para mover hacia las diferentes direcciones. Es así como se realizará una intervención directa a la caja de control FY 6V 27mhz, en los relés internos que tiene el dispositivo, afortunadamente este receptor tiene salidas directas de 5v para poder accionar sus relés y también tiene los conectores listos para accionar los dos motores que posee el vehículo. En la (Figura 72) se presenta el dispositivo, visto tanto desde fuera como dentro, este dispositivo trabaja a 6v y a 27 MHz como su nombre indica.

Figura 72. Caja de control y recepción por Control remoto FY 6V 27mhz
Fuente: (Amazon, 2018)

3.8.5. Motores DC

El vehículo tiene integrado dos motores XXX RS550S, como ya se explicó uno para la dirección y otro para la tracción del vehículo, en la (Figura 73). Se muestra la forma del motor, y en la (Tabla 11) sus características técnicas.

Figura 73. Motor de vehículo
de juguete XXX RS550S
Fuente: (Amaguaña, Collaguazo, Tituaña, & Aguilar, 2018)

Tabla 11
Especificaciones técnicas de Motor XXX RS550S

Ítem	Detalles
Nombre	Motor eléctrico magnético de DC del alto esfuerzo de torsión 10000RPM / 5000RPM 12V 50mA / 6V 40mA
Peso neto:	41 g
Voltaje Nominal	DC 12V 50mADC 6V 40mA
Corriente Nominal	40 mA
Velocidad Nominal	10 000RPM
Torque nominal:	80 g-cm
Velocidad sin carga:	13000RPM
Eficiencia Máxima	49.6%
Corriente en Máxima eficiencia	2A
Velocidad a máxima energía	7845
Corriente en máxima energía	2.6 A
Torque con carga máxima	191 g-cm
Corriente con carga máxima	5.6 A

Fuente: (Amaguaña, Collaguazo, Tituaña, & Aguilar, 2018) (AliExpress, 2018) (Team-Paragon, 2013)

3.8.6. Baterías

Las baterías son dos First Power FP675 de 6v cuya imagen se muestra en la (Figura 74), y los detalles técnicos se ilustran en la (

Tabla *12*).

Figura 74. Batería recargable FirstPower FP675.

Tabla 12
Especificaciones Técnicas de batería FirstPower PF675.

Ítem	Detalles
Voltaje nominal	6v
Capacidad (AH)	7.5 Ah
Resistencia interna (ohm)	16
Peso	1.12 Kg.
Dimensiones:	151 x 34 x 100 mm (Largo x Ancho x Alto).
Terminal	T1

Fuente: (FirstPower, 2018)

3.8.7. Controlador Arduino

Para controlar los motores, se utiliza el Arduino Mega 2560, el cual nos proporciona con varias características que son necesarias para controlar los motores y a su vez intervenir en el receptor remoto, se usa también este dispositivo debido a la conexión por medio del puerto serial con el computador, ya que desde aquí podremos enviar ordenes hacia el Arduino dependiendo las necesidades del momento, en la (Figura 75) se muestra la imagen del Arduino.

Figura 75. Arduino MEGA 2560
Fuente: (Arduino, 2018)

3.8.8. Driver para motores

El dispositivo que permite controlar los motores para la navegación del vehículo será el driver para motores llamado Monster motor shield. Se ha elegido este dispositivo debido a sus características de driver de motor y su integración con Arduino, pero sobre todo por sus características de potencia la imagen del driver se presenta en la (Figura 76) y las especificaciones técnicas en la (Tabla *13*)

Figura 76. Monster motor shield
Fuente: (GeekFactory, 2018)

Tabla 13
Especificaciones técnicas de Monster Motor Shield VNH2SP30

Ítem	Detalles
Voltaje máximo	16 V
Corriente Máxima	30 A
Capacidad continua de corriente	14 A
Frecuencia de PWM máxima	20 khz
Resistencia en encendido de mosfet	mosfet: 19 miliohms

Fuente: (GeekFactory, 2018)

3.8.9. Sensor Inercial

Para este caso se utilizó el dispositivo MPU-6050, esta combina un giroscopio de 3 ejes y un acelerómetro de 3 ejes en la misma matriz de silicio (Tdk IvenSense, 2018). Es muy preciso, ya que contiene hardware de conversión de análogo a digital de 16 bits para cada canal. Por lo tanto, captura los canales x, y, z al mismo tiempo. El sensor usa el bus I2C para interactuar con el Arduino (Arduino, 2018).

Este dispositivo será utilizado para conocer el ángulo al cual se coloque el vehículo y con ello corregir la dirección dependiendo del dato real necesitado. Este sensor irá conectado con el Arduino que es el que procesará las señales recibidas del mismo. El dispositivo será colocado en el volante del vehículo, puesto a que este es el que define el ángulo, al que se desplazará debido a su acción directa sobre las llantas frontales. En la (Figura 77) se muestra la apariencia física de este sensor.

Figura 77. MPU-6050
Fuente (Arduino, 2018)

Para un seguimiento preciso de los movimientos rápidos y lentos, las piezas cuentan con un rango de escala de giroscopios programable por el usuario de ± 250, ± 500, ± 1000 y ± 2000 (dps), y un acelerómetro programable por el usuario completo rango de escala de ± 2g, ± 4g, ± 8g y ± 16g. Las características incluyen un sensor de temperatura integrado y un oscilador en el chip

con una variación de ± 1% sobre el rango de temperatura de funcionamiento (Tdk IvenSense, 2018).

3.9. Integración del sistema

En la (Figura 78). Se ilustra el diagrama representativo de las conexiones del hardware que ya han sido descritos anteriormente. Todos los dispositivos van colocados sobre el vehículo, tanto los sensores LIDAR y ZED, como los actuadores y controladores que van dentro en la compartición existente del vehículo bajo su asiento en la Figura 79 se muestra el vehículo implementado con todos los elementos.

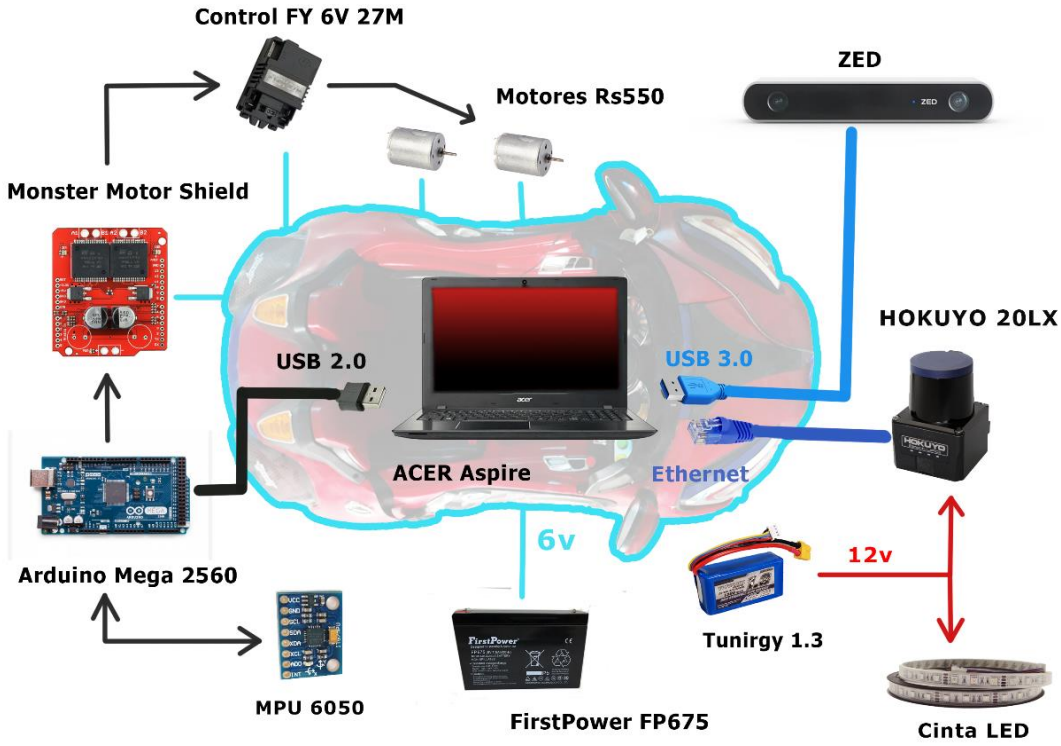


Figura 78. Diagrama de conexiones general

El diagrama general de conexiones comienza por la conexión de las baterías que entregan 6v al vehículo, lo que también hace funcionar las luces propias del vehículo. La alimentación también llega hacia el driver, motores y controlador remoto. El Arduino Mega 2560 se conecta por medio de un USB 2.0 al computador y se alimenta del mismo puerto, en el Arduino también va conectado el MPU-6050, el cual recibe tanto la alimentación como la comunicación directamente del Arduino. El driver Monster motor shield va sobrepuesto en el Arduino, pero utiliza la alimentación de 6v del vehículo para poder controlar y las acciones de movimiento e inversiones de giro.

La estéreo cámara ZED ocupa el puerto USB 3.0 y también se alimenta de allí. El sensor LIDAR HOKUYO se conecta al computador por medio del puerto ethernet, y se alimenta de la batería externa Tunirgy de 12v como se describió anteriormente. Esta misma batería a su vez acciona la cinta led colocada bajo el vehículo para poder identificarlo en modo nocturno.

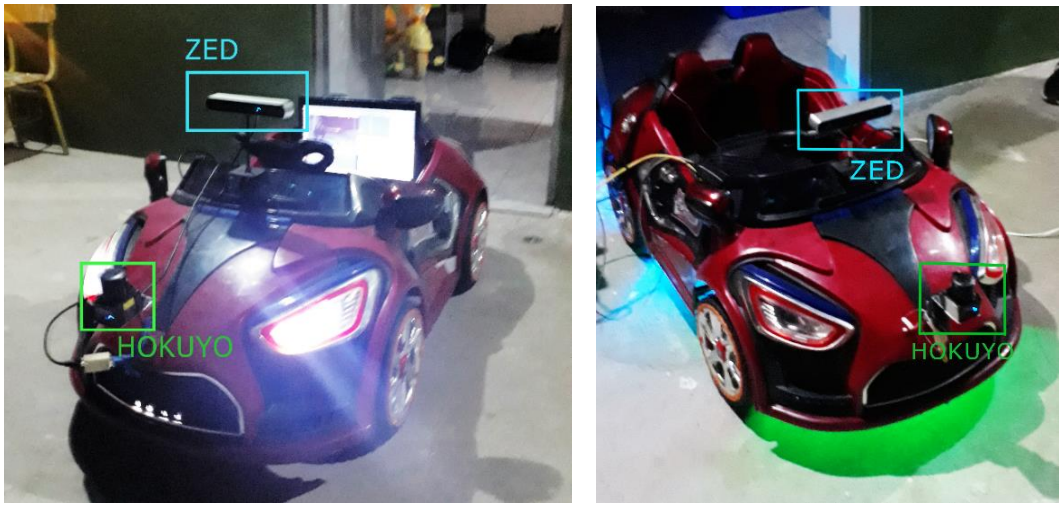


Figura 79. Resultado final del montaje de todos los elementos sobre el vehículo

CAPITULO IV

4. PLANIFICACION DE TRAYECTORIAS

Este capítulo describe el desarrollo e implementación del algoritmo de planificación de trayectorias basado en la combinación de la percepción provista tanto por el LIDAR 2D como por la cámara de estéreo visión, sobre el UGV con locomoción Ackerman de cuatro ruedas con dirección frontal y tracción posterior. La planificación de trayectorias como se define en (Gim, Adouane, Lee, & Dérutin, 2017) requiere considerar las restricciones cinemáticas, es por ello que de forma inicial en este capítulo se desarrolla el modelamiento cinemático del vehículo.

La implementación del algoritmo de planificación se conjuga en la navegación autónoma, la cual tiene por objetivo resolver tres problemas (Sclutz 2003), localización y mapeo, planificación de trayectorias y control de movimientos. Para el primer punto se aplica el algoritmo Core SLAM el cual permite generar los mapas 2D los cuales traducen un entorno desconocido en un entorno conocido. La planificación de trayectorias como se anticipo fue desarrollada por medio del algoritmo RRT basado en espacios de riesgo y para sistemas No-Holonómicos. Finalmente, el control de movimientos se lleva a cabo por medio de un sistema de control en el cual la cámara de estéreo visión permite la realimentación y corrección de trayectorias.

Para el desarrollo del algoritmo se utilizó el entorno de desarrollo integrado Eclipse por medio de las herramientas de desarrollo del lenguaje C y C++, llamado CDT. En (Janes, Sillitti, & Succi, 2012) califica a esta opción de desarrollo como una opción de software libre con alto desempeño comparable a otros lenguajes como Java y JavaScript. El desarrollo implementado en Eclipse es ampliamente utilizado en el desarrollo de aplicaciones científicas ya que permite una

amplia experimentación, implementación, interconectividad y el uso de herramientas anexas como las herramientas para la paralización de procesos (Watson & DeBardeleben, 2006).

El desempeño del algoritmo se evalúa por medio de parámetros definidos como fijos y variables. Los resultados establecen un UGV autónomo con la mayoría de las características definidas por (Sathignary 2011); entorno definido, capacidad de desplazarse de A hasta B, evitar colisiones y adaptable a las condiciones del ambiente como obstáculos dinámicos.

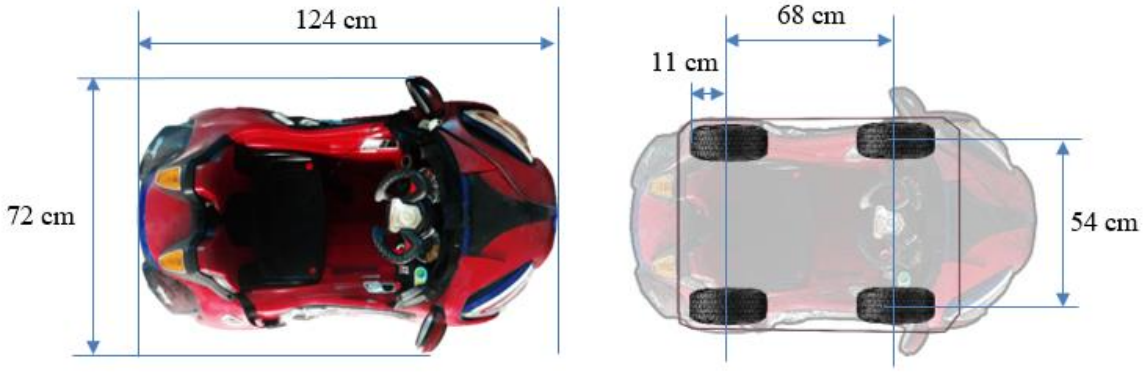
4.1. Modelamiento cinemático

Es necesario conocer las restricciones físicas del vehículo, debido a que gran parte de la estimación de movimientos se basará en modelados físicos. El modelo cinemático del vehículo entregará como resultados las reacciones físicas que se efectuarán en función del tiempo, mismas que luego serán usadas para describir el movimiento en el plano de acuerdo con los principios básicos de la física como por ejemplo el movimiento rectilíneo uniforme (MRU), movimiento rectilíneo uniformemente variado (MRUV), movimiento circular uniforme y movimiento circular uniformemente variado. La planificación de la trayectoria implica definir la geometría del vehículo y el comportamiento de las trayectorias. De forma general se definen tres tipos de trayectoria; en línea recta y en arcos circulares derecho e izquierdo (Gupta, Divekar, & Agrawal, 2010).

4.1.1. Geometría del vehículo

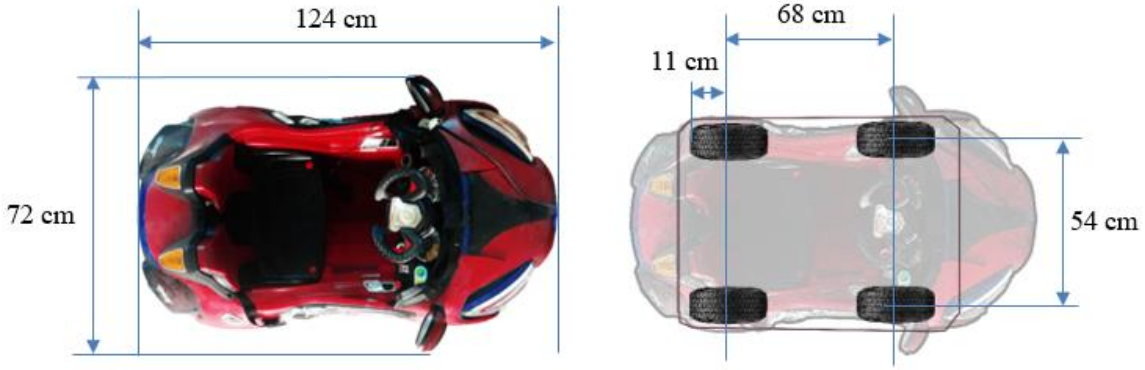
Es necesario conocer a detalle las condiciones del vehículo, y los principales factores que lo determinan matemáticamente en un plano con coordenadas fijas por el cual se desplazará y también por medio del cual se toma como referencia total durante el viaje.

La distancia vertical entre cada llanta es igual a 68 cm y de separación horizontal entre ellas es de 54cm, y las llantas son todas del mismo tamaño, con un radio de 11 cm. Estos parámetros son los que definen las restricciones del sistema, debido a que estos influyen en los movimientos y las llantas son los puntos de soporte que actúan directamente con la superficie de desplazamiento. También es importante considerar la longitud total del vehículo en la que se incluye el chasis, pues pese a que no influyen directamente en los movimientos, ocupan un lugar en el espacio y pueden provocar colisiones durante su navegación, en la (



(a) (b)

Figura 80) se muestra el contorno del cual se obtiene el modelado matemático y su relación en cuanto a la longitud real del vehículo por medio de una simulación.



(a) (b)

Figura 80. Geometría del vehículo
(a) Vehículo completo (b) Geometría del modelo

El vehículo cuenta con un sistema de 4 llantas, las cuales 2 permaneces estáticas y las otras dos giran por medio de un volante, cambiando únicamente el ángulo de ellas por medio de un mecanismo interno, a esto se lo define como un modelo de dirección de Ackerman según la teoría que dice: El Principio de Dirección de Ackerman define la geometría que se aplica a todos los vehículos sean cuales tengan 2 ruedas de dirección o 4 ruedas de dirección para permitir que se genere el ángulo de giro (Choi, Park, Lee, & Lee, 2008).

4.1.2. Condición de Ackerman

Para realizar el movimiento del vehículo, en el que se incluyen movimientos curvos es necesario contar con un sistema de movimiento que es el que entrega una fuerza que mueva para crear el movimiento del vehículo y un sistema de direcciones que se encarga de entregar direcciones a través de cambios angulares.

El sistema de dirección viene dado por medio del giro del volante, el cual actúa por medio de un mecanismo con elementos que permiten transformar el giro del volante en un desplazamiento lineal que varía la orientación de las ruedas directrices (Sánchez, Oliva Meyer, & Sánchez Lozano, 2013). El giro de las ruedas debido a la transformación no genera el mismo ángulo para ambas ruedas como se muestra en la (Figura 81), a este mecanismo también lo suelen llamar mecanismo trapezoidal.

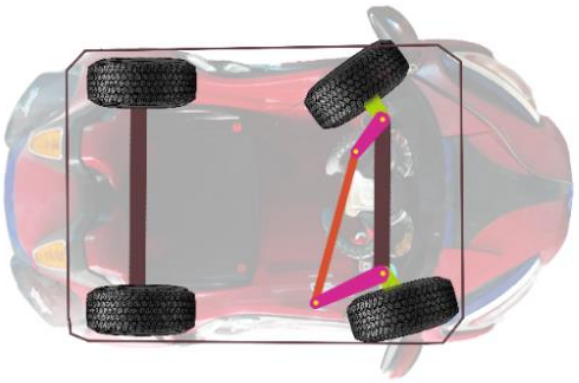
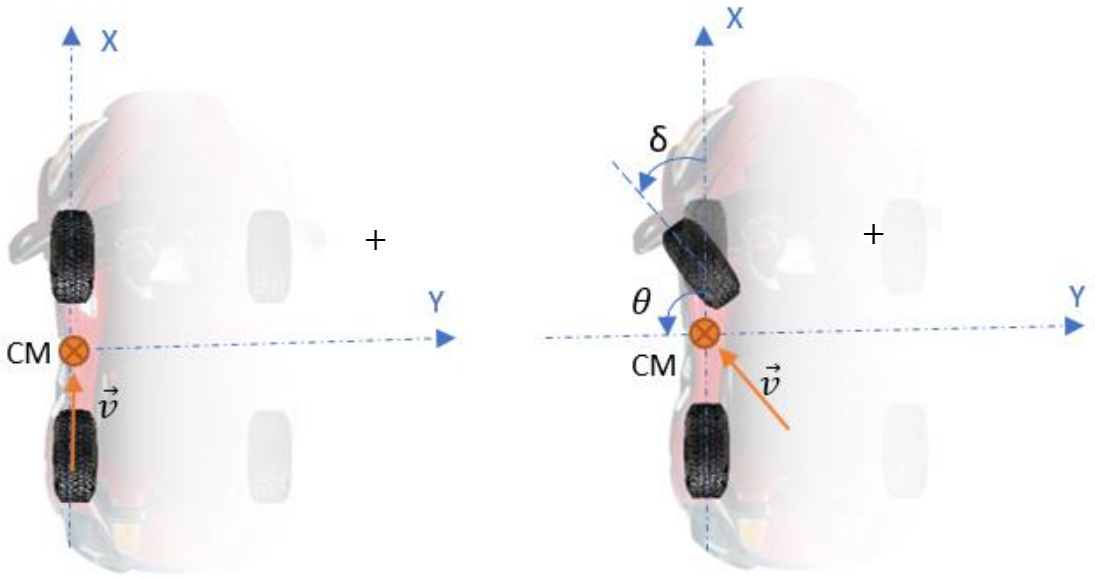


Figura 81. Sistema de dirección

Para realizar el giro el sistema se basa en el funcionamiento simple de una bicicleta, donde la llanta de adelante da la dirección del sistema y se genera a partir de ello un radio de giro que depende del ángulo de la llanta delantera, si únicamente nos concentramos en un lado del vehículo se puede comprender mejor este tipo de funcionamiento (Torres Moreno, 2014), previo a realizar el análisis del modelo Ackerman en la (Figura 82) se muestra el funcionamiento de la geometría lateral.



(a) (b)

Figura 82. Análisis de curvas con dos llantas
(a): estado normal (b): con ángulo de giro

La fuente que produce el movimiento es un motor y se encuentra en una de las llantas traseras, pero afecta a ambas llantas generando un movimiento sincronizado por medio de un mecanismo produciendo la fuerza que mueve al vehículo. Cuando se realiza el giro de la llanta se genera un ángulo δ que crea desde la línea vertical que indica la posición normal y la dirección que toma la llanta, además se crea una velocidad angular que da como resultado una trayectoria curva que se puede aproximar a un movimiento circular. Desde el centro de masa (CM) se crea un vector de velocidad resultante que toma el ángulo de giro de la llanta como se muestra en la (Figura 82), este ángulo en realidad es una aproximación sin tomar en cuenta todos los factores físicos que se involucran, como lo hacen en sus apartados algunos autores como (Zong, Gong, & Guo, 2012) (Correa J. , 2010) (Hernandez Beleño, Bernardes Vítor, Vaqueiro Ferreira, & Siqueira Meirelles, 2014).

Conocer la geometría lateral del vehículo nos facilita la explicación del modelamiento dinámico del sistema Ackerman, debido a que tiene un funcionamiento similar, pero en su lugar se considera la geometría completa del vehículo.

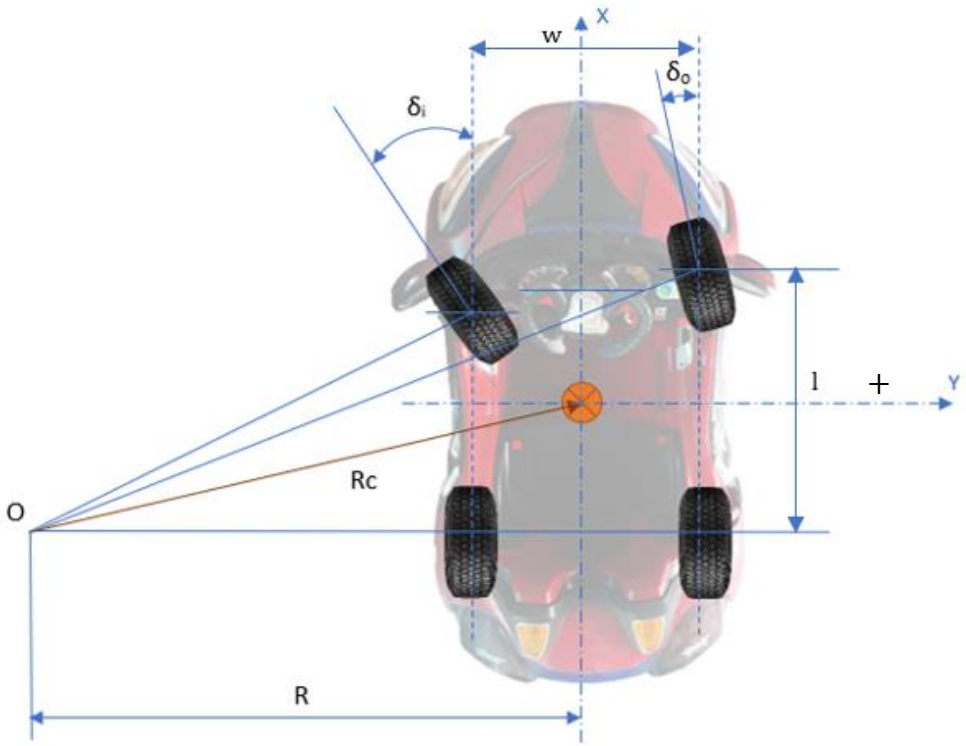


Figura 83. Modelamiento dinámico del vehículo

Como se puede observar en la (Figura 83), nuestro modelo de vehículo, en donde se cambia el eje X a la mitad del vehículo, así como su centro de masa y el radio de giro R se toma desde este eje debido a que la curva resultante se tomará desde este punto central.

El concepto de Ackerman es hacer que las cuatro ruedas se muevan sin deslizarse, alrededor de un punto común durante un giro (Bonnick, 2008). Este punto común se toma dependiendo el grado de curva de las llantas directrices, y será el que define nuestra trayectoria, a este punto en común se conoce como centro instantáneo de curvatura (Gupta, Divekar, & Agrawal, 2010). En la (Figura 84) se ilustra cómo se genera el movimiento a través del centro de curva instantáneo definido como O. Partimos de las relaciones trigonométricas posibles, obteniendo los ángulos de inclinación δ_i y δ_o de la siguiente manera:

$$\tan \delta_i = \frac{l}{R-\frac{w}{2}} \tag{18}$$

$$\tan \delta_o = \frac{l}{R-\frac{w}{2}} \tag{19}$$

Despejando e igualando R se tiene que

$$\frac{l}{\tan \delta_o} - \frac{w}{2} = \frac{l}{\tan \delta_i} + \frac{w}{2} \tag{20}$$

Y simplificando esto se llega a la condición de Ackerman que cumple nuestro mecanismo trapezoidal mostrado en la figura 2.

$$\cot \delta_o - \cot \delta_i = \frac{w}{l} \tag{21}$$

Si promediamos los ángulos obtenidos por ambas llantas y lo llevamos a un único ángulo ubicado en el centro este es el conocido como ángulo de Ackerman que se calcula a partir de los ángulos δ_o y δ_i según la fórmula (3).

$$\cot \delta = \frac{\cot \delta_o + \cot \delta_i}{2} \tag{22}$$

Y si tomamos el triángulo rectángulo formado con el radio de giro obtenemos:

$$\tan \delta = \frac{l}{R} \tag{23}$$

Para poder simular y estimar los movimientos consideramos que todos los movimientos que realizará el vehículo son círculos basados en el radio de giro que se pude obtener de:

$$\delta_i + \delta_o = \delta = \delta_{ack} \tag{24}$$

$$R = \frac{l}{\delta_{ack}} \tag{25}$$

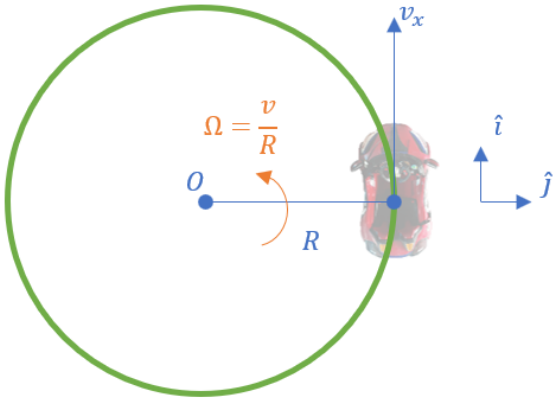


Figura 84. Representación de movimiento circular

4.1.3. Transformación de coordenadas

El vector de velocidad fija del cuerpo gira en todo el circulo de radio R definido de la siguiente manera:

$$\overrightarrow{v}^{x,y} = \begin{bmatrix} v_x \\ 0 \end{bmatrix} \tag{26}$$

Tomando en cuenta la velocidad angular (ω) que adopta el giro tenemos:

$$v = \omega\pi \tag{27}$$

$$v_x = R * \Omega \tag{28}$$

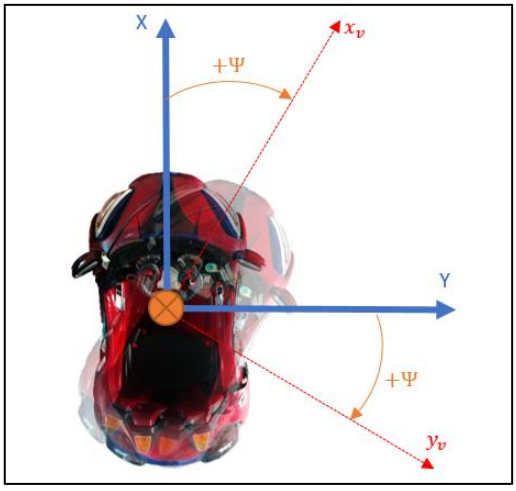


Figura 85. Coordenadas de cuerpo fijo

Partiendo desde nuestra definición del sistema de coordenadas, en el cual el eje x siempre toma la cabeza del vehículo, se ilustra en la (Figura 85). Cuando el vehículo realice giros, los giros que se han definido, son positivos para el lado derecho y negativos para el lado izquierdo tomando el centro como el ángulo equivalente a cero grados.

Cuando el vehículo gira cambia su posición por lo tanto su sistema de coordenadas las cuales su dirección y sentido sin definidas de acuerdo con el vector de velocidad resultante, de ahí es como se ha colocado el nombre de las nuevas coordenadas con un subíndice v, para distinguirlas.

Asumiendo que este giro conlleva a un desplazamiento en los ejes considerado como Ψ y partiendo de la ecuación 11 tenemos:

$$v_x = R * \dot{\Psi} \tag{29}$$

$$\dot{\Psi} = \frac{v_x}{R} \tag{30}$$

Donde Ψ es la dirección en la punta del vehículo y $\dot{\Psi}$ es la velocidad de giro. Partiendo desde la ecuación 8, usando la dirección tenemos:

$$\dot{\Psi} = \frac{v_x}{R} = v_x\left(\frac{l}{R}\right) \tag{31}$$

$$\dot{\Psi} = v_x\left(\frac{\delta_{ack}}{l}\right) \tag{32}$$

$$\dot{\Psi} = \frac{v_x}{l}(\delta_{ack}) \tag{33}$$

Para poder realizar la simulación, usaremos el cálculo de trasformación para las coordenadas fijas, para lo cual a las coordenadas originales las llamaremos con mayúsculas como (X, Y) y a las coordenadas del vehículo como (x, y).

$$\hat{\imath} = (c\Psi)\hat{I} + (+s\Psi)\,\hat{J} \tag{34}$$

$$\hat{j} = (-s\Psi)\hat{I} + (c\Psi)\,\hat{J} \tag{35}$$

$$\begin{bmatrix}\hat{\imath}\\ \hat{\jmath}\end{bmatrix} = \underbrace{\begin{bmatrix} c\Psi & s\Psi \\ -s\Psi & c\Psi \end{bmatrix}}_{[T]_{XY}^{xy}} \begin{bmatrix}\hat{\imath}\\ \hat{\jmath}\end{bmatrix} \tag{36}$$

Donde la matriz $\begin{bmatrix}\hat{\imath}\\ \hat{\jmath}\end{bmatrix}$ representa a la matriz del eje de coordenadas principal (X, Y) y la matriz de transformación para llevar hacia las coordenadas del vehículo (x, y) es $\begin{bmatrix} c\Psi & s\Psi \\ -s\Psi & c\Psi \end{bmatrix}$. Entonces si aplicamos la transformación tenemos:

$$\begin{bmatrix}\hat{\imath}\\ \hat{\jmath}\end{bmatrix} = \underbrace{\begin{bmatrix} c\Psi & -s\Psi \\ s\Psi & c\Psi \end{bmatrix}}_{[T^t]_{xy}^{XY}} \begin{bmatrix}\hat{\imath}\\ \hat{\jmath}\end{bmatrix} \tag{37}$$

Por lo tanto, el vector de velocidad del vehículo $\overrightarrow{v_{veh}}^{XY}$ al aplicar la matriz de transformación lleva como resultado al mismo vector, pero en relación con el sistema de coordenadas del vehículo $\overrightarrow{v_{veh}}^{xy}$.

$$\overrightarrow{v_{veh}}^{XY} = [T^t]\, \overrightarrow{v_{veh}}^{xy} \tag{38}$$

$$\begin{bmatrix} v_X \\ v_Y \end{bmatrix} = \underbrace{\begin{bmatrix} c\Psi & -s\Psi \\ s\Psi & c\Psi \end{bmatrix}}_{[T^t]} \begin{bmatrix} v_x \\ 0 \end{bmatrix} \tag{39}$$

$$v_{veh}{}^{X} = (c\Psi \cdot \mathrm{v_x})\hat{\imath} \tag{40}$$

$$v_{veh}{}^{Y} = (c\Psi \cdot \mathrm{v}_x)\hat{\jmath} \tag{41}$$

Finalmente, para definir la pose del vehículo las coordenadas originales realizamos la integral de la velocidad del vehículo en sus propias coordenadas $(v_{veh}{}^{X}, v_{veh}{}^{Y})$ e integramos el ángulo del giro realizado Ψ .

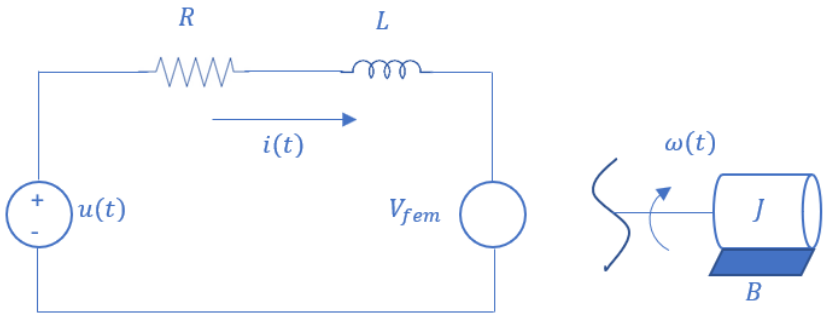
$$X_{veh} = \int v_{veh}{}^{X}\, dt \tag{42}$$

$$Y_{veh} = \int v_{veh}{}^{Y}\, dt \tag{43}$$

$$\Psi = \int \dot{\Psi}\, dt \tag{44}$$

4.1.4. Modelamiento del motor

Como se indicó anteriormente el vehículo cuenta con un motor dc para generar la tracción trasera que induce al movimiento al vehículo, para realizar el modelamiento del motor se parte desde su representación teórica (Chapman, 2014).



(a) (b)

Figura 86. Representación de motor DC,
a) equivalencia eléctrica b) equivalencia mecánica

En la (Figura 86) se interpreta la equivalencia eléctrica y mecánica respectivamente, con las cuales podremos realizar los cálculos teóricos, en donde cada variable tiene es siguiente significado:

- R: Resistencia de los bobinados
- L: Inductancia
- J: Momento de inercia
- B: Coeficiente de fricción viscosa entre el rotor y el estator
- $u(t)$: Voltaje
- $\omega(t)$: Velocidad angular
- $\tau(t)$: par del motor

- $i(t)$: corriente del sistema
- V_{fem}: Tensión eléctrica inducida
- tm: Constante de tiempo mecánico
- te: Constante de tiempo mecánico

Partimos definiendo el valor de voltaje que cae sobre la bobina en función del tiempo para lo cual se tiene (Zill, 2012) (Meza & Ramos, 2015):

$$V_L: L\frac{di(t)}{dt} \tag{45}$$

Luego usando las leyes de Kirchhoff, tenemos:

$$L\frac{di(t)}{dt} = u(t) - Ri(t) - V_{fem}(t) \tag{46}$$

Y se obtiene la ecuación de representación eléctrica, y luego se procede a usar la parte mecánica.

$$\tau(t) = B\omega(t) + J\frac{d\omega(t)}{dt} + Ti \tag{47}$$

$$J\frac{d\omega(t)}{dt} = \tau(t) - B\omega(t) - Ti \tag{48}$$

Finalmente usamos las ecuaciones para relacionar la parte eléctrica con la parte mecánica, para la cual tenemos los siguientes términos:

- Ti: Par de fricción inicial
- K_t: Constante de acople de torque
- K_E: Constante fuerza electromotriz

Y estas ecuaciones son:

$$\tau(t) = K_t\, i(t) \tag{49}$$

$$V_{fem}(t) = K_E\, \omega(t) \tag{50}$$

Según las características del motor se tiene características constantes como:

$$R = 2.4\,\Omega \tag{51}$$

$$L = 0.5\,H \tag{52}$$

$$Tm = 3.1\,ms \tag{53}$$

Puede obtener los valores a los que el motor responde de acuerdo con la carga para los cuales se tiene:

- Sin Carga

$$V = 13000\,RPM \tag{54}$$

$$I = 55\,mA \tag{55}$$

$$\tau = 80\,g - cm \tag{56}$$

$$J = 1x10^{-7} \tag{57}$$

$$B = 2.8x10^{-6} \tag{58}$$

$$\omega = 1047\,rad/s \tag{59}$$

- Con Carga al máximo

$$V = 10000\,RPM \tag{60}$$

$$I = 5.6\,A \tag{61}$$

$$\omega = 1047\,rad/s \tag{62}$$

$$\tau = 191\,g - cm \tag{63}$$

- Máxima Eficiencia

$$Ef(\%) = 49.6\% \tag{64}$$

$$W = 5.96\ w \tag{65}$$

$$V = \ 10000\ RPM \tag{66}$$

$$I = \ 2\ A \tag{67}$$

$$\tau \ = \ 60\ g - cm \tag{68}$$

- Máxima Energía

$$W = 6.44\ w \tag{69}$$

$$V = \ 7845\ RPM \tag{70}$$

$$I = \ 2.6\ A \tag{71}$$

$$\tau \ = \ 80\ g - cm \tag{72}$$

Para poder similar, necesitamos conocer los parámetros del motor, con la carga a la cual se lo pondrá a trabajar, para ello necesitamos completar los valores que necesitamos así que procedemos a calcular, primero calcularemos la constante eléctrica, para lo cual nos guiamos de los datos obtenidos de las características (29):

$$te = \frac{La}{Ra} = \frac{0.5}{2.4} = 0.21 \tag{56}$$

Luego calculamos la V_{fem}, por medio de la siguiente ecuación, donde V es el voltaje inducido en la entrada:

$$V_{fem} \ = \ V \ - \ (\ La \ \times \ Ra) \tag{56}$$

$$V_{fem} \ = \ 6 \ - \ (\ 0.5 \ \times \ 2.4) = 4.8 \tag{57}$$

$$V_{fem} \ = 4.8 \tag{58}$$

La constante contra electromotriz la obtenemos de:

$$K_E = \frac{V_{fem}}{n} = \frac{V}{\omega} \tag{59}$$

$$K_E = 0.005 \tag{60}$$

Por dato la constante de torque corresponde a:

$$K_T = 4.4\ x\ 10^{-3} \tag{61}$$

Para facilitar los cálculos la aproximamos a:

$$K_T = 0.005 \tag{62}$$

De igual manera usando el tiempo mecánico podemos calcular el momento de inercia

$$Jm\ = \frac{tm \cdot K_T \cdot K_T}{Ra} \tag{63}$$

$$Jm\ = \frac{3.1 \cdot 0.005 \cdot 0.005}{2.4} \tag{64}$$

$$Jm\ = 3.22 \times 10^{-4} \tag{65}$$

Aproximando para facilitar los cálculos tenemos:

$$Jm\ = 0.0003 \tag{66}$$

Luego calculamos el torque de fricción usando:

$$Tf\ =\ K_T\ *\ Iar \tag{67}$$

$$Tf\ =\ 0.005\ *\ 5.6 \tag{68}$$

$$Tf = 0.028 \tag{69}$$

Finalmente calculamos la Constante de Fricción de Coulomb (β) por medio de:

$$\beta\ = (K_T\ *\ Ia) -\ Tf \tag{70}$$

$$\beta\ = (0.005\ *\ 5.9) -\ 0.028 \tag{71}$$

$$\beta\ = 0.0015 \tag{72}$$

4.1.5. Simulación

La primera simulación realizada en MATLAB engloba el modelo cinemático realizado asi como tambien transformaciones de posicion para poder conocer el movimiento del vehiculo partiendo desde sus acciones fisicas. Complementariamente se ha modelado el motor dc que usa para impulsar al vehiculo que es el que dará los parametros de velocidad al mismo. Finalmente se realiza los diagramas en SIMULINK con los respectivos calculos y se muestra los analisis teoricos para el vehiculo.

Tomando en cuenta los parámetros de dirección, velocidad del motor y modelamiento cinemático se ha realizado el diagrama de bloques en SIMULINK mostrado en la Figura 87 en el cual se alberga de manera modular lo anteriormente mencionado.

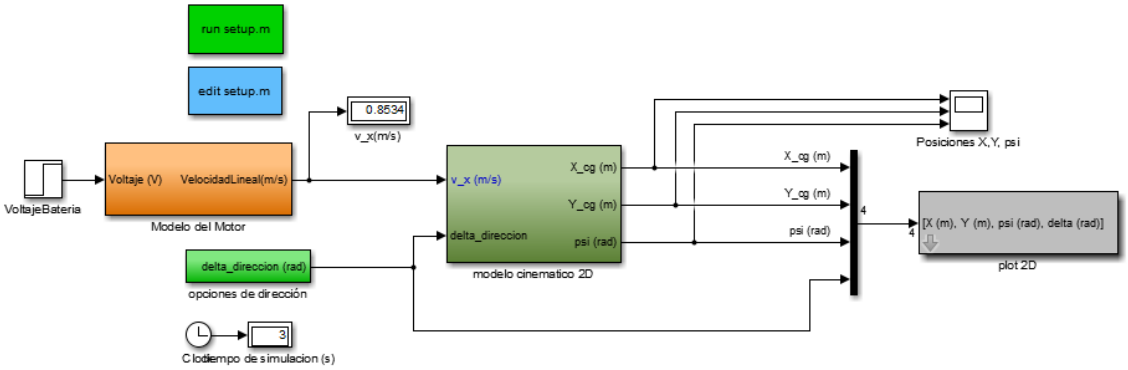


Figura 87. Diagrama de bloques de simulación completo (SIMULINK).

4.1.5.1. Simulación del motor DC

Primero procedemos a explicar la parte de la simulación del motor, en la cual se integra tanto la parte eléctrica como mecánica calculada en el apartado anterior, y representada en la (Figura 88). Por el bloque de color naranja. Primero procedemos a calcular la corriente y luego la velocidad angular. Partiendo de un voltaje de entrada, mismo que será el obtenido de la batería, con ello simularemos la velocidad a la que girará el motor a su máxima velocidad, finalmente se

transformará la velocidad angular a velocidad lineal al multiplicar el resultado con el radio de las llantas, cabe recordar que las llantas de atrás se encargan de realizar ese movimiento.

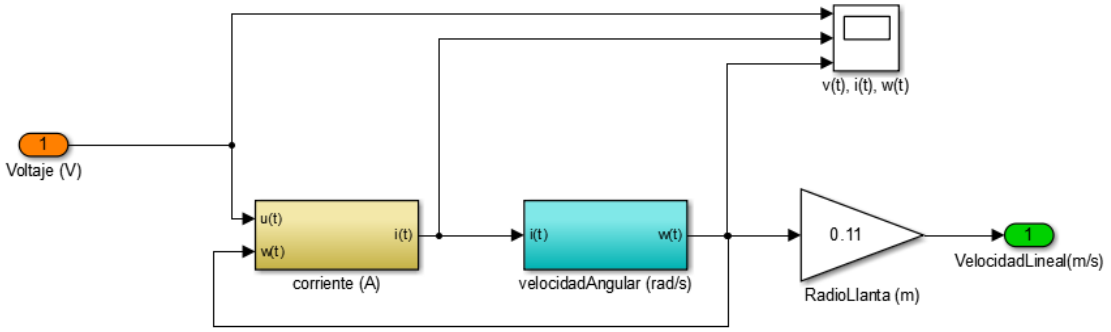


Figura 88. Simulación Modelo del Motor DC SIMULINK

Dentro del bloque amarillo mostrado en la (Figura 88). Se encuentra el diagrama correspondiente al cálculo de corriente mostrado en la (Figura 89).

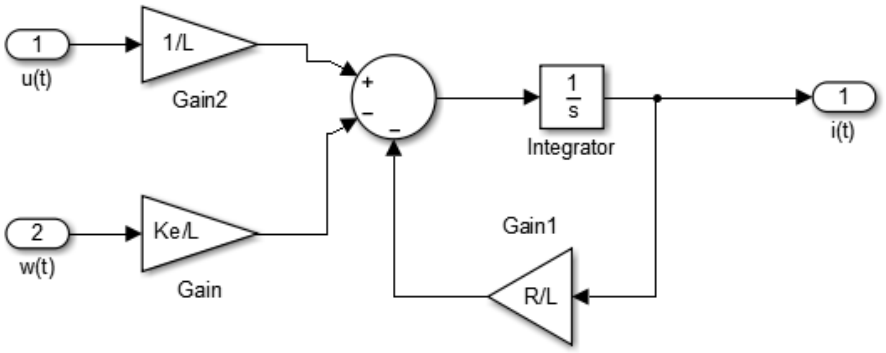


Figura 89. Diagrama de bloques cálculo de corriente SIMULINK

De la misma manera una vez obtenida la corriente se procede a calcular la velocidad angular por lo que la variable obtenida del cálculo anterior entra en el siguiente diagrama pintado de color cian como se muestra en la (Figura 90). Y representado en la (Figura 90).

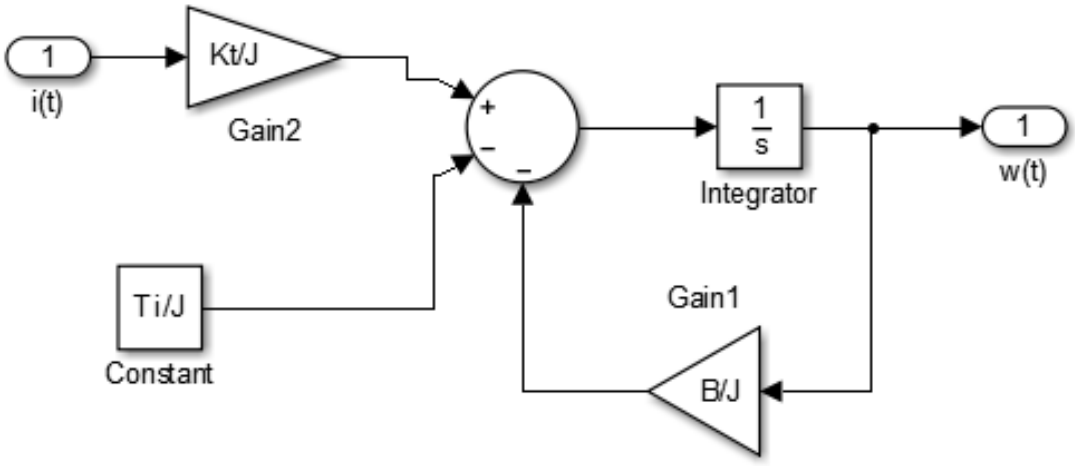


Figura 90. Diagrama de bloques cálculo de velocidad angular SIMULINK

Usando las ecuaciones anteriores se procede a simular en MATLAB, para lo cual primero se simula el motor en sus condiciones iniciales es decir sin carga. Usando los valores que nos dan por defecto los datos del motor, usando la función escalón para la entrada de voltaje, tomando en cuenta el voltaje que entregan las baterías que es 6v.

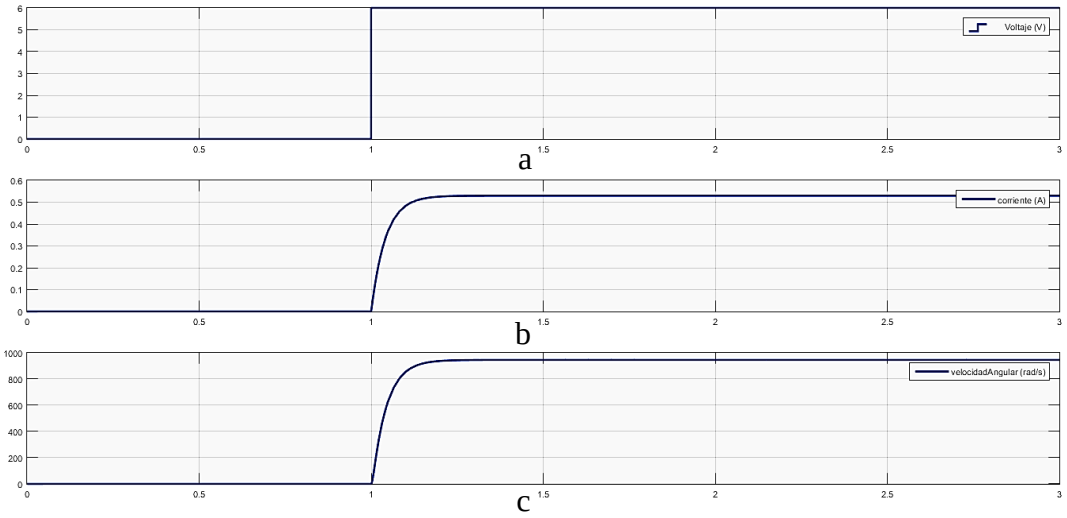


Figura 91. Respuesta de simulación del motor sin carga
a) Voltaje de entrada b) corriente c) velocidad angular

Como se muestra en la (Figura 91), frente a una entrada de 6 voltios se alcanza una corriente de aproximadamente 0.52 A, y una velocidad angular cercana a 1000 rad/s que equivale a

9549,29 y es un valor cercano al valor dado por el fabricante del motor. Nuestro fin es utilizar este motor con nuestra carga y conocer a qué velocidad se moverá el vehículo, por lo cual procedemos a simular con los datos utilizados con carga.

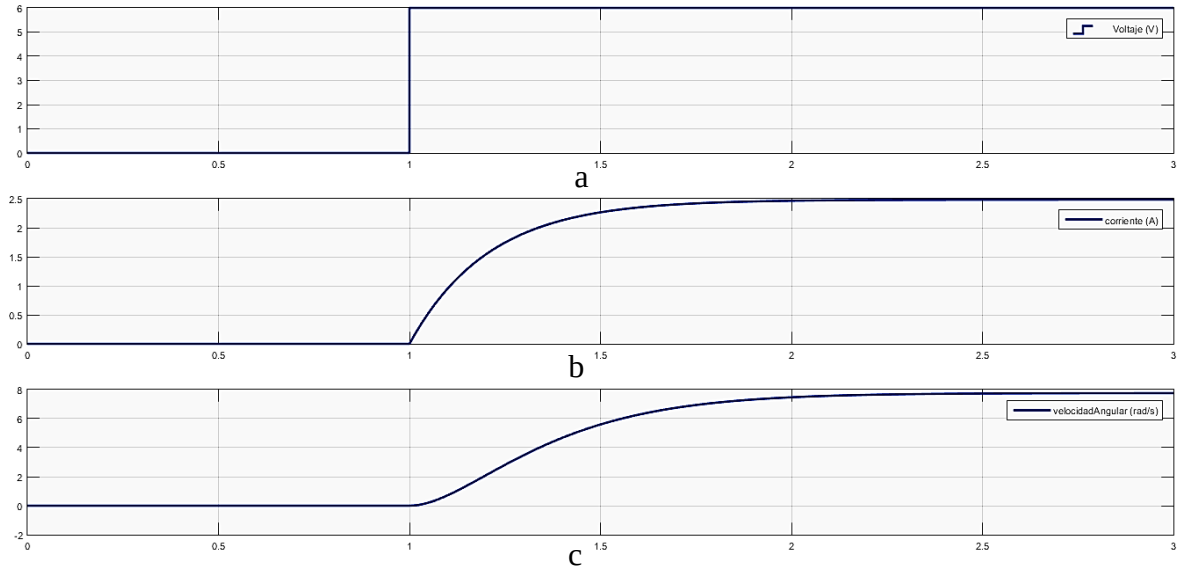


Figura 92. Respuesta de simulación del motor sin carga
a) Voltaje de entrada b) corriente c) velocidad angular.

En la (Figura 92) se muestra que ante la misma entrada de 6 voltios la corriente alcanza un valor de 2.6 A, curiosamente es similar al criterio de máxima energía del motor especificado por el fabricante, la velocidad angular resultante es de aproximadamente 8 rad/s. que al multiplicarla por el radio de la llanta se entrega como resultado 0.85 m/s, previamente se midió experimentalmente la velocidad y como resultado se obtuvo 0.82 m/s lo que indica que está muy bien y estos valores quizá cambien por cuestiones de rozamiento u otros factores externos.

4.1.5.2. Simulación de la dirección

Puesto a que la dirección se realiza por medio de los ángulos generados por el movimiento del volante, que serán realizados por cálculos computacionales resultantes de la planificación de trayectorias, en esta simulación se envían ángulos predefinidos únicamente con el fin de mostrar

la respuesta del sistema, para realizar este trabajo se utilizó una función predefinida como se muestra en la (Figura 93).

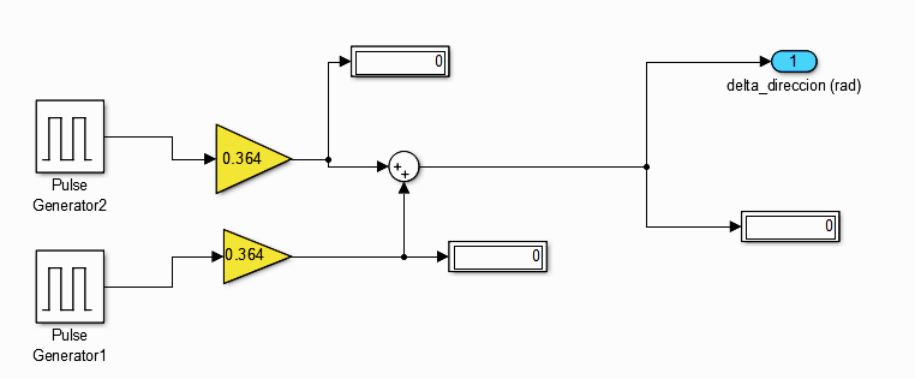


Figura 93. Diagrama de bloques de dirección.

Usando los generadores de pulsos con tiempos diferentes, se puede simular los giros, para el ejemplo se utilizó el valor de 0.364 que es la resultante a obtener la tangente de 20 grados, se dio un tiempo para girar a la izquierda y luego se iguala el valor para ir recto y finalmente se activa únicamente el generador 1 para girar a la derecha, el generador 2 arroja únicamente valores negativos y por otro lado el generador 1 únicamente valores positivos que es como realmente funciona el planificador.

4.1.5.3. Simulación del modelo cinemático del vehículo.

La simulación del modelo dinámico representado por el bloque verde mostrado en la (Figura 94). Contiene las ecuaciones que se obtuvieron del modelo cinemático, basado en la longitud y el ancho del vehículo.

$$v_X = \left[v_x \cdot \cos(\Psi) \ - \left(\frac{L}{2}\right) \cdot \omega \cdot \sin(\Psi)\right] \tag{73}$$

$$v_Y = \left[v_x \cdot \sin(\Psi) + \left(\frac{L}{2}\right) \cdot \omega \cdot \cos(\Psi)\right] \tag{74}$$

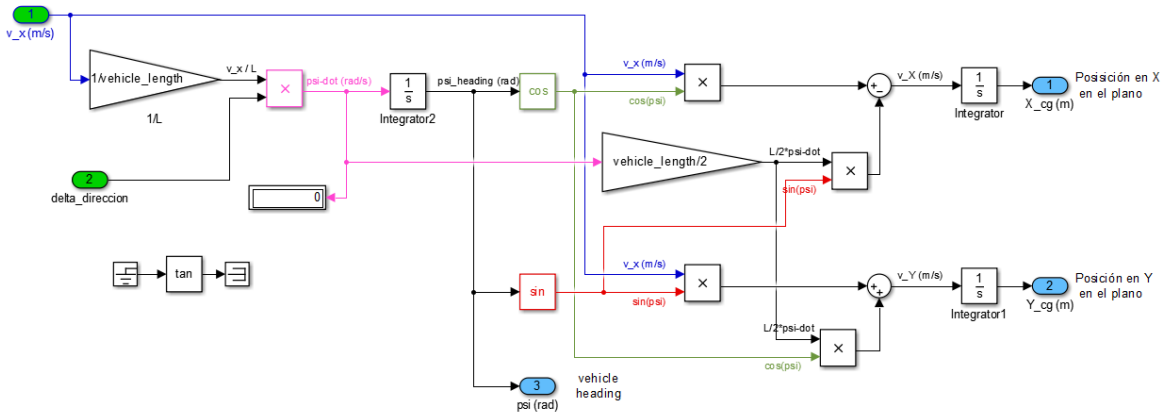


Figura 94. Diagrama de bloques del modelo cinemático del vehículo

Como se había indicado previamente v_X y v_Y son las velocidades sobre el sistema de coordenadas y v_x es la velocidad global del vehículo. A este bloque ingresa la velocidad lineal del motor, y la dirección del volante y se obtiene como resultado la posición sobre el sistema de coordenadas y la dirección del vehículo.

Finalmente se envía a un bloque de graficado, en el cual el vehículo prediseñado en un script muestra no solo la animación de la dirección, sino que también se grafica un boceto del vehículo. Adicional se muestra los resultados de la posición y dirección en función del tiempo de manera independiente.

A manera de ejemplo se planteó mover el vehículo a velocidad constante con 6v de entrada recto, luego girar hacia la izquierda, nuevamente recto y finalmente girar a la derecha, y este es el resultado final de la simulación.

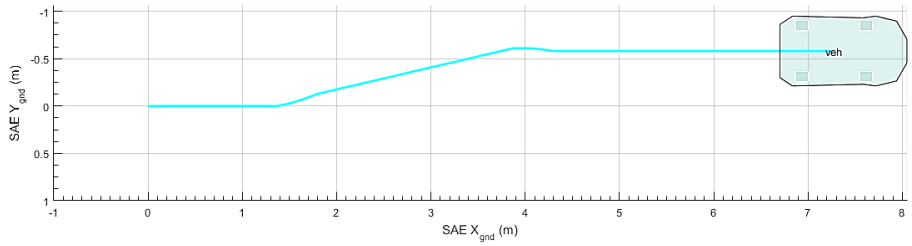


Figura 95. Simulación de modelo cinemático

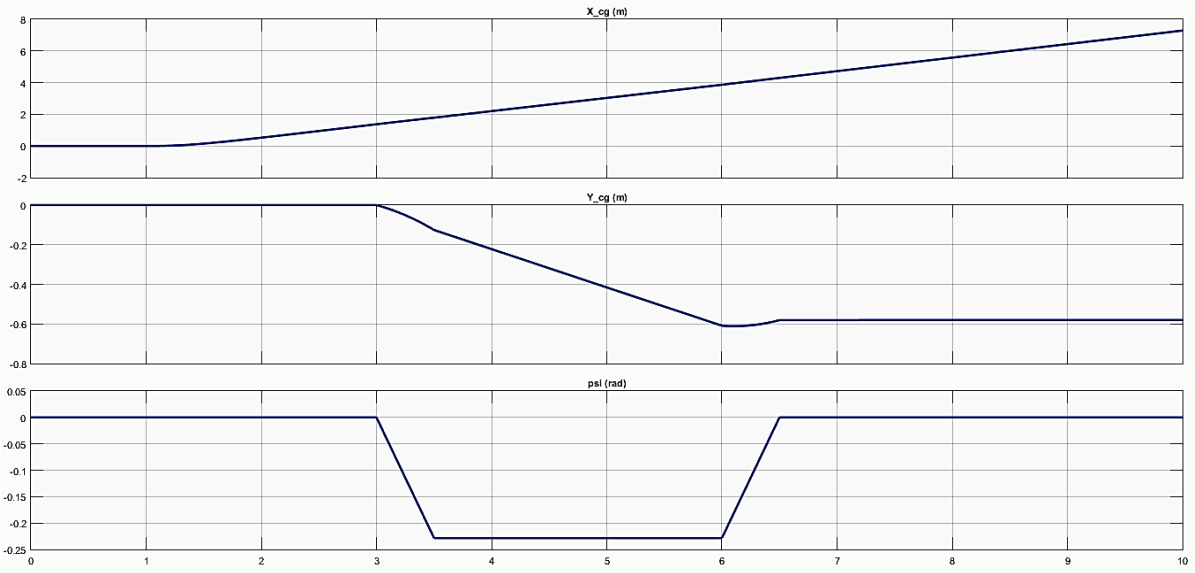


Figura 96. Respuesta de posición y orientación.
a) Posición X. b) Posición Y. c) Orientación Ψ

En la (Figura 95) se muestra el resultado final de la simulación en la cual el vehículo realiza los movimientos establecidos. En la (Figura 96) se muestra la respuesta de la respuesta al diagrama de bloques para establecer la posición y orientación. Una vez que se conoce como actúa el vehículo, se abre la posibilidad de realizar la planificación enfocada a un análisis dinámico, como una partícula en el plano para conocer de manera simplificada la pose del robot y la trayectoria. Por lo tanto, lo que sigue es obtener el plano o en este caso el mapa.

4.2. Localización y mapeo

La estimación de la posición o localización basadas en la percepción del entorno utiliza sensores de video o sensores activos (Jiménez & Baturone, 1996). Previamente se estableció al sensor activo LIDAR como fuente de percepción 2D, con ello se establece la localización y mapeo. Esto gracias a la técnica SLAM definida previamente en el Capítulo II. Consecuentemente la implementación del algoritmo Core SLAM, está sujeta a parámetros

simples y un código de no más de doscientas líneas de código. Allí, se aprovecha la percepción de profundidad 2D en el desarrollo de mapas parciales, que definen el espacio conocido X.

Parte inicial al momento de implementar el algoritmo es la definición de parámetros que permiten el uso de LIDAR y la generación de los mapas. Como se muestra a continuación los parámetros son:

- *MAP_SIZE,* es el tamaño máximo al cual puede almacenarse un mapa en pixeles, para este caso 2048.
- *MAP_SCALE,* es el factor de conversión entre los datos recibidos y los datos a almacenar en el mapa, este valor es de 0.1.
- *SCANSIZE,* este parámetro es la cantidad de puntos otorgados por el LIDAR y por lo visto en el Capítulo anterior este valor es de 1081.
- *MAXSCANS,* este valor es la cantidad de perfiles de profundidad 2D, utilizados para la elaboración del mapa, se estableció inicialmente con un valor de 50.
- *offset,* es el ángulo desplazado para el cual el sensor toma su punto central según las especificaciones vistas este valor es de 90 grados.
- *angle_min y angle_max,* el sensor tiene un barrido de 270 grados, por lo que desplazar 90 grados sus ángulos máximo y mínimo son 135 y -135 grados.
- *detection_margin,* este parámetro consiste en la menor distancia que puede detectar el sensor LIDAR, gracias a las pruebas realizadas en el Capítulo anterior, se establece que tiene un valor de 60mm
- *distance_no_detection,* es el rango máximo de detección que posee el sensor, de igual manera previamente se lo estableció en 20000mm.

Una vez establecido los parámetros y la pose inicial se desarrolla el algoritmo de la (Figura 97), algoritmo *slam(),* en el cual se procede a recibir los datos pertenecientes al perfil de profundidad 2D por medio de la función ReadScans (sensor.data). Posterior es necesario inicializar el mapa en vacío. En el paso 3 se dibuja el perfil de profundidad por medio de funciones gráficas para cpp llamadas SDL, un ejemplo de aquello es el perfil mostrado en la (Figura 98). El perfil junto con la pose inicial y los parámetros establecen un estado inicial, del cual parte el algoritmo Core SLAM.

Algoritmo 1: coreSLAM - slam()

Input : parametros, pose.inicial

```
1 ReadScans(sensor.data);
2 MapInit(mapa);
3 SDL.DrawScan();
4 StateInit(mapa, pose.inicial);
5 for l = 0 to nbscans do
6 |   DistanceScanToMap(sensor.data[nbscans],mapa);
7 |   MapUpdate(mapa,pose.inicial);
8 end
```

Output: mapa

Figura 97. Algoritmo para generación de mapa coreSLAM

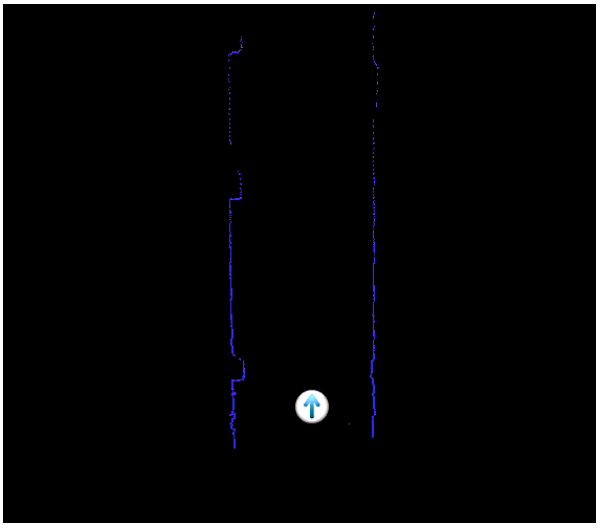


Figura 98. Perfil de profundidad para estado inicial

Resulta necesario detallar el algoritmo perteneciente a la función ReadScans(), por el cual parte de la base de CoreSLAM ha sido modificada. En la Figura 99 se describe esta función, de forma inicial se realiza un bucle de iteraciones pertenecientes al número máximo de escaneos. El sensor se comunica por medio de ethernet por lo que se establece la conexión como muestra la línea 4. A continuación se establece el vector donde se almacenan los valores del sensor, lo que da pie a urg.start.mesurement().

Algoritmo 2: Lectura del perfil 2D - ReadScans()

```
   Input  : *sensor.data
 1 long *length.data, time.stamp;
 2 for l = 0 to TEST.MAX.SCANS do
 3 |   sd = sensordata[l];
 4 |   urg.open(ETHERNET, "192.168.0.10", 10940);
 5 |   length.data = urg.max.data.size();
 6 |   urg.start.measurement();
 7 |   length.data.size = urg.get.distance(lengthdata);
 8 |   *time.stamp = urgtimestamp;
 9 |   urg.close();
10 |   urg.delay(100);
11 |   for i = 0 to TEST.SCAN.SIZE do
12 |   |   sd.d[i]=length.data [i];
13 |   end
14 end
```

Figura 99. Algoritmo para la toma del perfil 2D del sensor laser

Es allí donde se almacenan los valores de distancias de forma consecutiva, adicional se retiene el tiempo actual y se genera una latencia de 100 milisegundos para que el sensor responda con nuevos datos en la siguiente iteración. Finalmente, los datos son asignados al puntero perteneciente a sensor.data, lo cual permite continuar con el algoritmo Core SLAM.

Como se describe en la metodología del algoritmo consta de dos funciones esenciales, el cálculo del perfil de distancias al cual le hemos llamado perfil 2D y la actualización del mapa la primera función apila los perfiles y compara entre perfil nuevo y el mapa que se va generando

consecutivamente. Esto se lleva a cabo mediante un algoritmo simplificado de Montecarlo, el cual hace coincidir los recursivos perfiles en un mapa integrado.

Se aplica un filtro de partículas el cual utiliza la probabilidad de que cada partícula replique su posición. En este algoritmo la odometría se puede ignorar, sólo se utiliza para establecer el punto de inicio. Cabe señalar que el filtro de partículas establece una incertidumbre del 10% esto quiere decir que el 10% de las partículas son ruido. El mapa se crea con una resolución de un pixel equivalente a un centímetro. Se toma en cuenta el concepto de latencia para medir deslizamientos relativos, permitiendo que puntos se centren en la escala establecida.

Por otro lado, la función de actualización de mapa, lo que realiza es sintetizar el mapa por medio del algoritmo de Breseham modificado, el cual establece que un área definida como obstáculo se represente con una función agujero en el cual cada punto, está disperso, lo cual recae en que el centro de este agujero define al punto real del obstáculo. Esta función sirve como guía y estabilización del mapa tanto en posición como en orientación. Estas dos funciones se repiten conforme al número de escaneos locales.

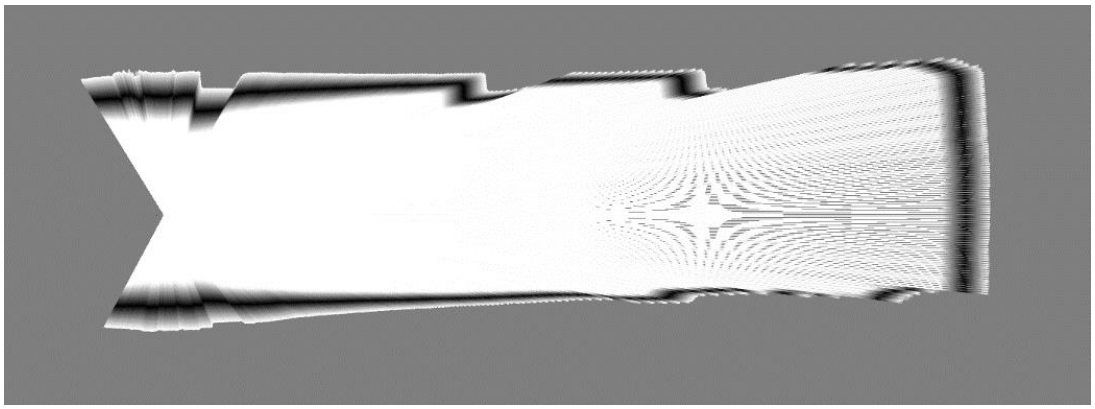


Figura 100. Mapa bidimensional PGM del entorno mapeado

Una vez completo el algoritmo slam(), se regresa como resultado el vector equivalente al mapa de pixeles, adicionalmente se genera un archivo de imagen a escala de grises PGM como el

de la Figura 100, el cual se atribuye al perfil previamente mostrado. Es necesario que la orientación del mapa generado tenga un desfase de 90° atribuido al funcionamiento del sensor por lo cual se corrige la pose inicial desfasándola este valor. En la imagen se puede apreciar los contornos como funciones agujero y un 10% de partículas, esta forma de mapa de grilla es beneficioso ya que agrega un grado de seguridad para evitar futuras colisiones. Si el caso fuese reconstruir fielmente el mapa 2D sería adecuado montar un filtro de partículas más desarrollado.

4.3. Planificación de trayectorias

La investigación y desarrollo en el campo de la robótica móvil para solventar el problema de planificación de trayectorias en entornos desconocidos y basadas en sensores activos implica el uso de planificadores globales y locales. En (Local and Global Planning) se establece que para la navegación de un UGV es necesario un planificador local para garantizar la convergencia al objetivo en función de las restricciones cinemáticas. Por otro lado, los planificadores globales construyen el modelo global de la trayectoria, para este caso se constituye el árbol de RRT.

4.3.1. Planificador Local

Este planificador basa su funcionamiento en la generación de trayectorias específicas de acuerdo con la cinemática definida. La implicación directa consiste en conectar un punto inicial con un punto final y alcanzable. El vehículo funcionará de acuerdo con la velocidad y ángulo que se le asigne, y realizará este movimiento por un determinado tiempo, luego tendrá que desde su pose final anterior realizar otro movimiento, pero para que la siguiente posición sea correcta es necesario que desde el punto final el vehículo conozca su pose actual y a partir de allí con ese nuevo sistema de referencia genere la siguiente trayectoria.

Primero procedemos a asumir que el robot siempre realizará movimientos circulares, por lo que su trayectoria siempre será circular, entonces tenemos que definir la ecuación del círculo. Previamente se ha definido que el algoritmo funcione con centímetros, por lo tanto, las ecuaciones estarán dadas en centímetros.

Por lo tanto, la longitud y ancho del vehículo serán 68 y 54 cm respectivamente, adicional como condiciones iniciales se colocará al vehículo alineado con el eje x como se lo planteo en el modelamiento cinemático.

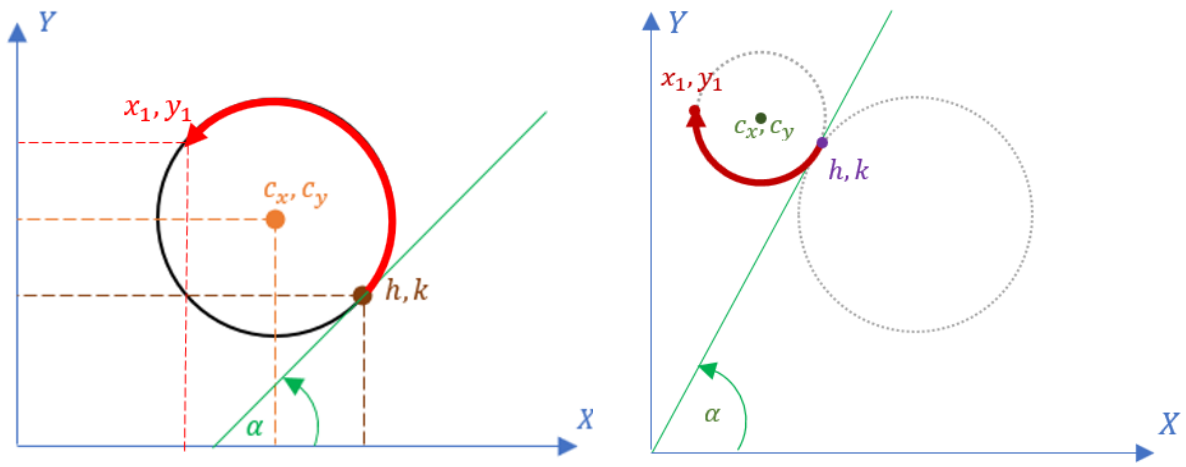
Partimos de la definición de la ecuación del círculo.

$$x_1 = (c_x - r \cdot \cos(\theta + \alpha)) + h \tag{75}$$

$$y_1 = (c_y + r \cdot \sin(\theta + \alpha)) + k \tag{76}$$

Estas ecuaciones definen al círculo en cualquier punto del plano, para lo cual tenemos que:

- (x_1, y_1): Punto en x en el sistema de coordenadas principal (X, Y).
- (c_x, c_y): Valor del centro del circulo en (X, Y).
- R: Radio del círculo.
- (h, k): Punto de inicio de la gráfica del circulo
- θ : Angulo de recorrido, definido por el tiempo
- α: Angulo de inicio a graficarse



(a) (b)

Figura 101. Variables que definen al círculo de trayectoria
a) trayectoria circulo inicial b) trayectoria de continuación

En la (Figura 101) se ilustra cómo funcionará, pues se asume que el vehículo parte desde la posición (h, k) y con el ángulo α, por lo tanto su próximo radio de giro será perpendicular a la recta formada por el ángulo y el punto de inicio, que indirectamente forman el sistema de coordenadas local o del vehículo en esa posición, el movimiento viene dado según el tiempo, por lo cual la variable θ se grafica en función del tiempo, cuando la partícula llegue a su posición final de llegada, el punto final pasa a ser el punto inicial de la siguiente trayectoria, se calcula el ángulo que recorrió el vehículo en el tiempo y luego este ángulo resultante pasa a ser α mientras que se está a la espera del nuevo ángulo de desplazamiento, así como la velocidad lineal del vehículo, que pasa a transformarse en velocidad angular para su respectivo análisis.

El radio de giro se obtiene a partir del ángulo de Ackerman definido anteriormente por la ecuación (8), luego se obtiene la velocidad angular descrita por

$$\omega = \frac{v_x}{R} = \frac{V}{R} \tag{77}$$

En este punto del análisis se tiene en consideración el tiempo, por lo cual, de acuerdo con las ecuaciones básicas de la dinámica, el ángulo recorrido se calcula por medio de:

$$\theta = \omega \cdot t \tag{78}$$

Se procede a calcular el punto centro para el giro.

$$cx = r \cdot \cos(\alpha) \tag{79}$$

$$cx = -r \cdot \sin(\alpha) \tag{80}$$

Para hacer que el ángulo dependa del tiempo las ecuaciones (75), (76), se parametriza según el tiempo a la variable θ al pasar a ser función del tiempo. Adicionalmente se tiene que validar en el caso de que los ángulos mayores a 0 sean positivos, y en cuanto a los que vayan al sentido contrario sean negativos y no valores mayores a 270 grados que es su equivalente, finalmente como se ha explicado en el apartado anterior se realiza al final del movimiento una actualización en las variables para que los parámetros finales de la trayectoria anterior pasen a ser los parámetros iniciales de la siguiente.

$$\alpha = \theta + \alpha \tag{81}$$

$$h = x \tag{82}$$

$$k = y \tag{83}$$

De manera que, si tomamos el ejemplo de la (Figura 101), la trayectoria que siguió realmente el vehículo fueron dos curvas, de la cual se partió de una pose inicial que se ha definido como (h, k, α), que en la ecuación representa el inicio del círculo, y el ángulo en el que se inicia, es decir su sistema de coordenadas local, que se muestra representado en la (Figura 102). Una vez que se conoce de donde parte en robot espera las instrucciones de ángulo, velocidad y tiempo que

se desplazará, estas variables definirán el próximo circulo a realizarse de las cuales usando la ecuación (25) para obtener el radio de giro R a partir del ángulo ingresado, se obtiene de le ecuación (77) la velocidad angular ω a partir de la velocidad linear V y finalmente usando la ecuación (78) se obtiene el ángulo final usando el parámetro del tiempo ingresado.

Despejando ω de (77) y (78):

$$\theta = \frac{V}{R}t \tag{84}$$

Una vez se ha realizado el modelo dinámico y se tiene las ecuaciones correspondientes se procede a estimar el movimiento, para obtener cual será la posición final del vehículo bajo los parámetros que sean establecidos, que como se ha explicado antes luego pasará a ser el nuevo punto de inicio para la siguiente estimación del movimiento mientras las coordenadas locales se siguen alineando a la posición actual del vehículo teniendo una trayectoria similar a la mostrada en la Figura 102.

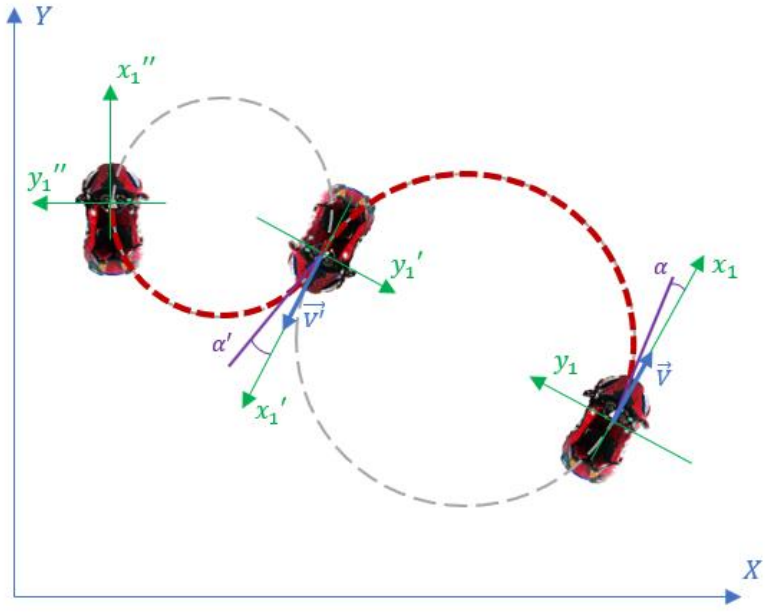


Figura 102. Estimación de movimiento a partir de la pose actual, la velocidad, tiempo y ángulo entregado

Las diferentes posibilidades de movimientos vienen dadas por los valores que se entrega de velocidad tiempo y ángulo, si consideramos el ejemplo mostrado en la (Figura 102)Figura 102, se observa que un circulo es más pequeño que el otro y va en otra dirección, por lo que se puede afirmar que $\alpha < \alpha'$, mientras menor sea el ángulo mayor será el radio de giro y por ende el circulo, si se da la siguiente condición.

$$R = \lim_{\delta_{ack} \to 0} \frac{1}{\tan \delta_{ack}} = \lim_{\alpha \to 0} \frac{1}{\tan \alpha} = 0 \tag{85}$$

El desplazamiento vendría a ser una línea recta, además en el sistema no solo se puede dar valores de velocidad positivos, también existe la posibilidad de dar valores negativos, lo que indicaría un movimiento en retroceso, debido a que cambia el sentido del vector y con esto se tiene muchas más opciones para realizar nuevas trayectorias, en la (Figura 103). Se ilustra un ejemplo de las posibles rutas que puede tomar un vehículo de acuerdo con los parámetros que sean ingresados. Para el caso de la (Figura 103 a)) se ilustra los círculos posibles si únicamente se cambia el ángulo del volante o de giro total del vehículo, antes denominado también como ángulo de Ackerman. En la (Figura 103 b)) se muestra las trayectorias que puede realizar y como se continuará a partir de las mismas, primero toma un ángulo negativo y cierta velocidad y se desplaza a un tiempo llegando al punto 1, donde su ángulo pasa a ser positivo para dar un giro hacia la derecha y con un menor ángulo y menor velocidad hasta llegar al punto 2, donde pese a que mantiene su dirección, la curva se hace más cerrada por lo que se presume que pudo haberse incrementado la inclinación o reducido la velocidad, en el punto 3 existe un cambio del vector velocidad a negativo por lo que procede a dar un movimiento hacia atrás y el ángulo pronuncia incluso aún más, desde el punto 4 finalmente se mueve en línea recta hasta llegar a 5 por lo cual el ángulo entregado debió ser cero a una velocidad nuevamente con dirección positiva.

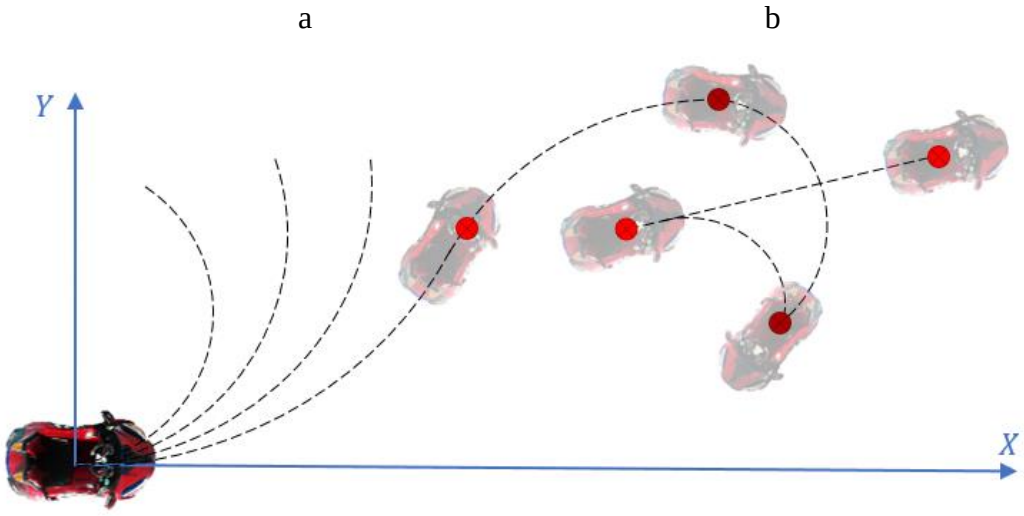


Figura 103. Posibles rutas que seguir del vehículo
a) Cambiando el ángulo b) trayectorias posibles cambiando todos los parámetros

4.3.1.1. Simulación del planificado local

Por medio d un script de MATLAB se procede a realizar una simulación de las trayectorias que sigue el vehículo, en la cual se ingresa por teclado los parámetros de inicio del vehículo como se muestra en la (Figura 104 a)) y luego se procede a realizar la simulación entregando el ángulo, la velocidad y el tiempo de la misma manera cómo funcionará el planificador de trayectorias, pero este se realiza de manera manual como se muestra en la (Figura 104 b)), únicamente con el fin de conocer previamente la dinámica del sistema, luego este mismo algoritmo de estimación es transcrito y parte del algoritmo de planificador de trayectorias. El resultado de esta simulación nos entrega como resultados los valores de la pose final y variables importantes para definir el movimiento como se muestra en la (Figura 104), así como la gráfica animada de los movimientos a realizarse en función del tiempo, la velocidad está en cm/s, el tiempo en segundos y el ángulo en radianes.

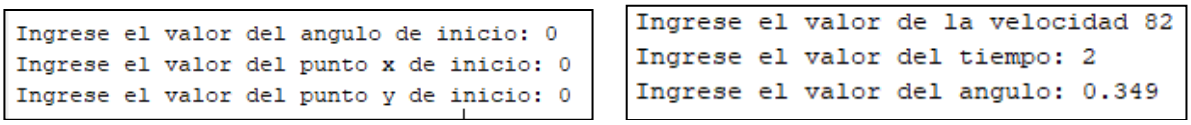


Figura 104. Ingreso por teclado de datos
a) datos iniciales b) parámetros de movimiento

Al finalizar la primera ejecución se obtienen los siguientes datos.

- R = 186.8668
- anguloRecorrido = 0.8776
- distanciaEstimada = 1.6186e+03
- xEstimado = 67.4635
- yEstimado = 143.7430
- anguloVehiculo = 0.8776
- anguloVehiculoGrados = 50.2845
- Ingrese 1 para continuar 0 para finalizar ruta:

Cabe recordar que estos datos se encuentran expresados en centímetro, para lo cual como respuesta resultante tenemos que llegará el vehículo al punto $(x = 67.46\,, y = 143.743)$cm con un ángulo de 0.8776 radianes que equivale a 50.28 grados. Por medio de la simulación de movimiento podemos observar de manera gráfica este comportamiento en el tiempo hasta llegar al punto final, la respuesta se ilustra en la (Figura 105).

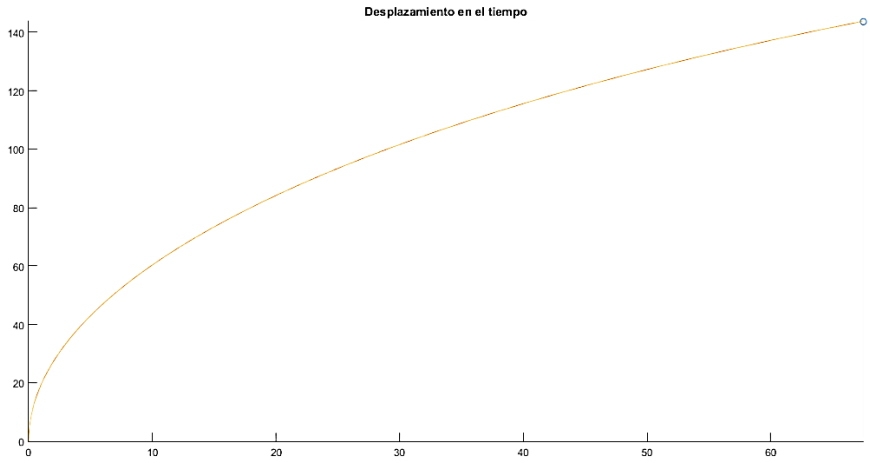


Figura 105. Animación del desplazamiento en el tiempo

Tambien existe otro grafico, que es el grafico resultante de la trayectoria recorrida, este tambien almacena todo el recorrido realizado en funcion del tiempo, este por ser el primer movimiento que realizamos será similar a la (Figura 105), pero en los posteriores movimientos será de gran utilidad, se puede observar este resultado en la (Figura 106). Tambien se muestra de manera independiente la variación en el tiempo de la posición, en la (Figura 107 a)) se muestra la variación de la posicion en X en el tiempo y en la (Figura 107 b)) se muestra la variacion en Y en el mismo tiempo, esta gráfica tambien acumula los datos, de manera que permita visualizar los cambios que existen entre los diferentes movimientos dinámicos.

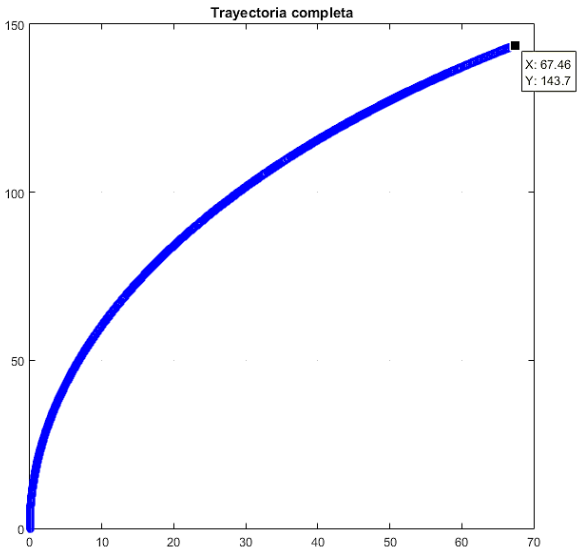


Figura 106. Trayectoria completa
de la simulación

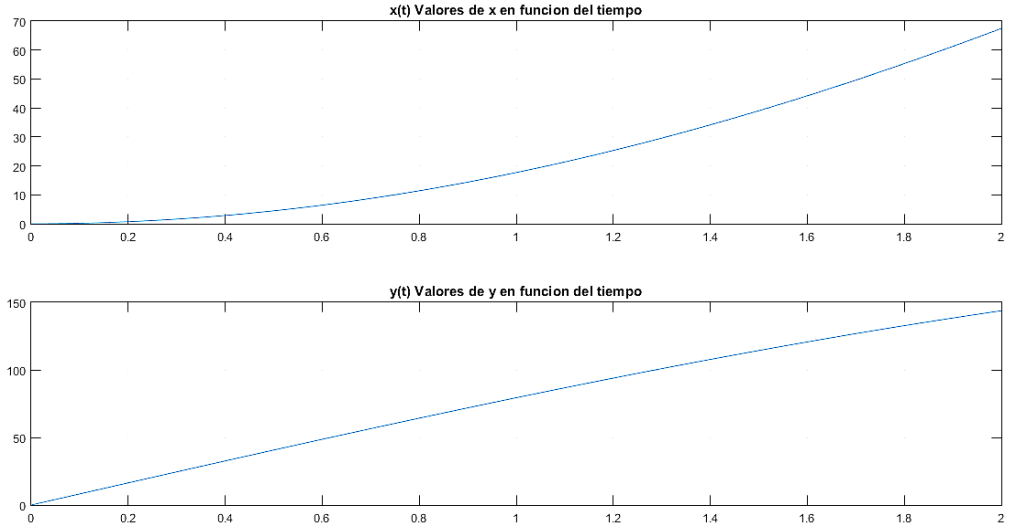


Figura 107. Respuesta individual de la posicion en el tiempo
a) Valores de x b) Valores de y

Luego procederemos a realizar otro movimiento, esta vez usando menor velocidad y en el sentido contrario para lo cual ingresamos en el programa los siguientes valores:

- Ingrese el valor de la velocidad: 54
- Ingrese el valor del tiempo: 2

- Ingrese el valor del angulo: -0.17

  Y se obitene como resultado de la pose final:

- xEstimado = 140.1657
- yEstimado = 223.1553
- anguloVehiculo = -0.2726

Y como resultado de la simulación tenemos los resultados mostrados en: (Figura 108, Figura 109, Figura 110).

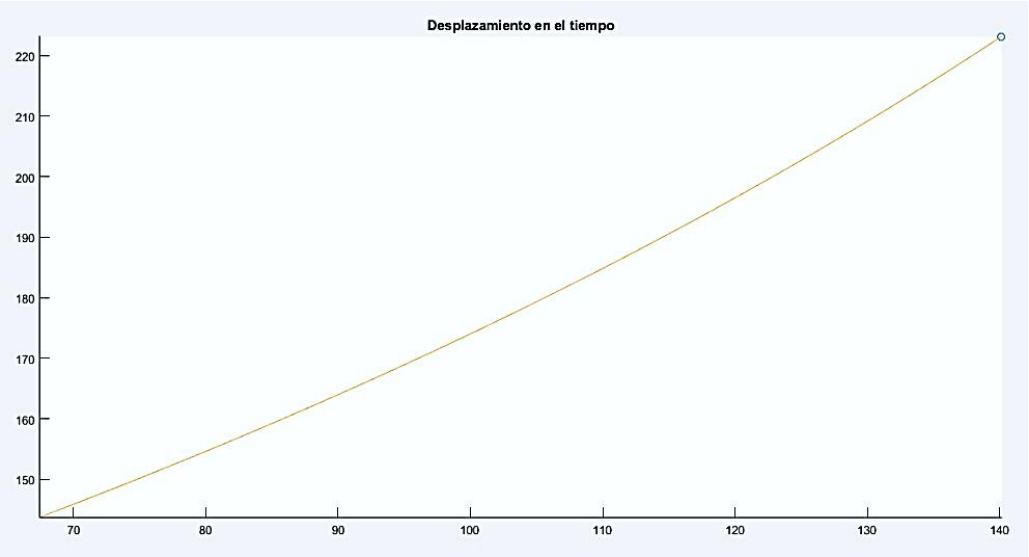


Figura 108. Respuesta individual de la posicion en el tiempo
a) Valores de x b) Valores de y

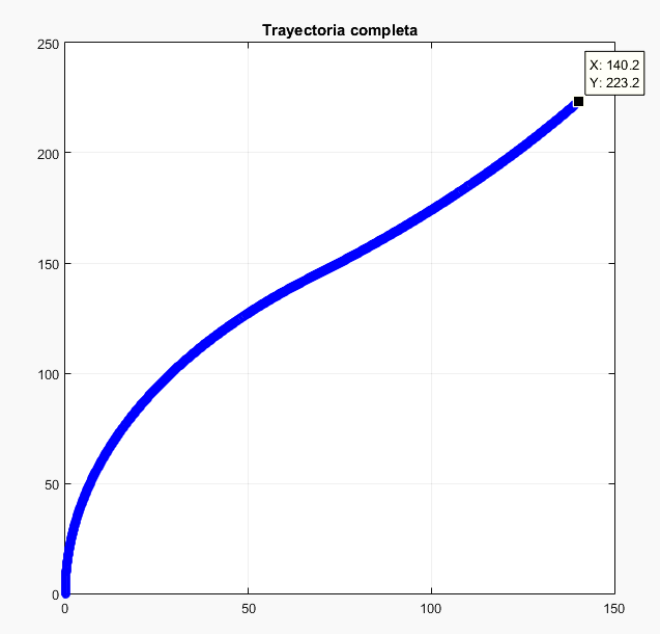


Figura 109. Trayectoria completa de la simulación con dos movimientos

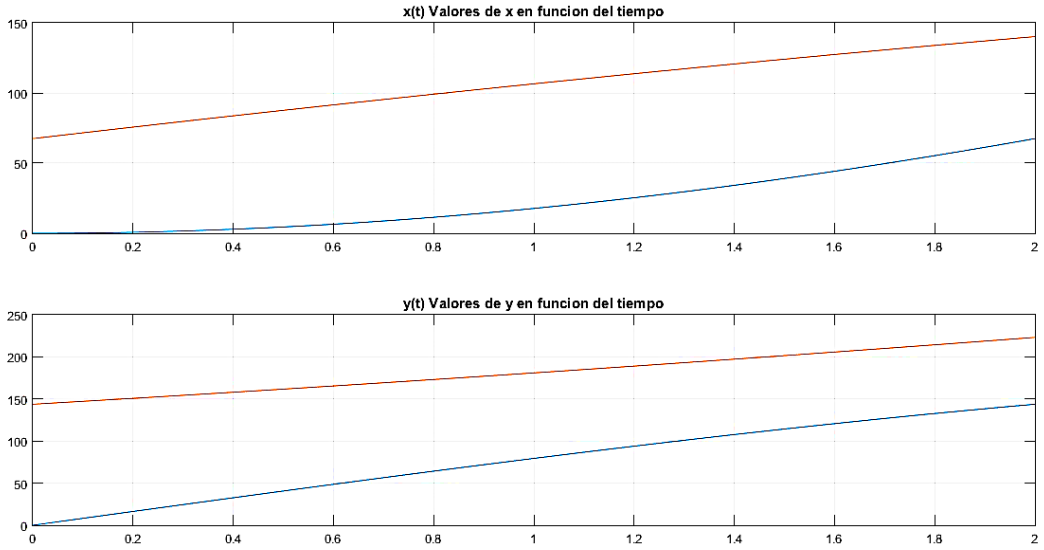


Figura 110. Respuesta individual de la posicion en el tiempo a) Valores de x b) Valores de y

De esta manera se puede estimar los movimientos y simular para comprobar el funcionamiento teórico con el funcionamiento real, tan solo se necesita colocar mas parámetros y se lo puede realizar las veces que se quiera.

4.3.2. Planificador Global

Como se exploró anteriormente existen dos algoritmos destacables RRT y PRM, estos algoritmos permiten la búsqueda rápido para entornos grandes. Sin embargo, se descarta una combinación de estos algoritmos ya que PRM requiere verificar que requiere el total conocimiento del ambiente y para este caso los mapas 2D se forman dinámicamente. RRT se puede aplicar en ambientes combinados entre conocidos y desconocidos lo cual se ajusta explícitamente a este caso. De las variantes existentes y descritas se ha decidido tomar como base el algoritmo Risk-RRT (Fulgenzi, Spalanzani, Laugier, & Tay, 2010) y modificarlo para vehículos no holonómicos en entornos parcialmente desconocidos.

El algoritmo posee dos tareas principales una la percepción de obstáculos fijos y móviles al fin de prevenir colisión, y la otra de planificar una trayectoria segura. Para la detección de obstáculos se realiza un enfoque conservador en el cual se arman rutas parciales ya que no se conoce a ciencia cierta que hay más allá de lo que el LIDAR no ve o de lo que puede llegar a ocurrir. Es así que se realiza un proceso Gaussiano para identificar el riesgo de la posición final, esto se puede representar matemáticamente como una función promedio y una función de covarianza:

$$f(x) \sim G(m(x), k(x, x')) \tag{86}$$

$$m(x) = E[f(x)] \tag{87}$$

$$k(x, x') = E[(f(x) - m(x))(f(x') - m(x'))] \tag{88}$$

Donde $G(\mu, \Sigma)$ representa un proceso Gaussiano con media μ y covarianza Σ funciones del dominio del espacio conocido E. Este principio sirve como base para desarrollar la evaluación de riesgo. Es así que la selección de las mejores trayectorias se realiza calculando la probabilidad de colisión para el UGV que sigue las trayectorias y evite los obstáculos. En la (Figura 111) se

muestra el algoritmo de Risk-RRT el cual posee aspectos iniciales como; inicialización, búsqueda hasta alcanzar meta, seguimiento pasa a paso de trayectoria, identificación del entorno hasta la Línea 12.

Risk-RRT

```
1:  procedure RISK-RRT
2:      trajectory = empty
3:      Tree = empty
4:      Goal = read()
5:      t= clock()
6:      while Goal not reached do
7:          if trajectory is empty then
8:              brake
9:          else
10:             move along trajectory for one step
11:         end if
12:         observe (X);
13:         delete unreachable trajectories(T, X)
14:         observe(Map, movingobstacles)
15:         t= clock()
16:         predict moving obstacles at time t, ..., t + Nτ
17:         if environment different then
18:             update trajectories(T,Map,moving obstacles)
19:         end if
20:         while clock()< t + τ do
21:             grow trajectories with depth<= N in T
22:         end while
23:         trajectory = Choose best trajectory in T
24:         t = clock()
25:     end while
26:     brake
27: end procedure
```

Figura 111. Algoritmo de planificación Risk-RRT
Fuente: (Fulgenzi, Spalanzani, Laugier, & Tay, 2010)

El momento que se alcance nuevas posiciones y en sí que el vehículo se desplaza es necesario descartar trayectorias inútiles o inalcanzables como se describe en la línea 13, en consecuencia, se observa el entorno definiendo los obstáculos y prediciendo obstáculos en movimiento. Si el ambiente consigue ser muy diferente se procede a actualizar las trayectorias. Lo que sigue es algo propio del RRT lo cual les hacer crecer el árbol mientras transcurre un tiempo lo cual se lo hace para evitar situaciones dinámicas.

Cabe destacar que al crecer la trayectoria (línea 21) es necesario escoger los nodos más prometedores. Es así que se pesa a todos los nodos tomando en cuenta su riesgo de colisión y estimando la distancia del mismo hasta la meta. Esto se lo conoce como heurística y está definida por:

$$\overline{w}(q_N) = \frac{\sqrt[n]{L_\pi(q_N)}}{dist(q_0, q_N, P)} \tag{89}$$

$$w(q_N) = \frac{\overline{w}(q_N)}{\sum_q \overline{w}_q} \tag{90}$$

En el primer numerador se normaliza la probabilidad del riesgo de $\pi(q_N)$, el nodo conocido. Se divide para la distancia entre el nodo, la raíz del árbol y la distancia aproximada de la trayectoria P. Este es para un nodo, luego se normaliza todos los pesos con el conjunto de nodos del árbol. El nodo que se escoge del conjunto de nodos es el nodo con menor peso. Es así que se llega a resultados como los mostrados en la (Figura 112), en la cual se aprecia que el algoritmo de riesgo y la función de búsqueda o crecimiento de trayectoria busca acercarse a la meta durante un tiempo hasta que se reinicia el árbol y se genera una nueva trayectoria llegando de forma paulatina a la meta.

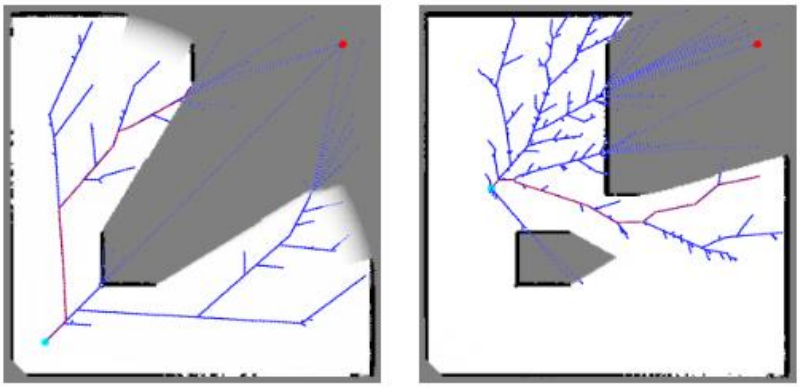


Figura 112. Crecimiento de la trayectoria en función del espacio explorado
Fuente: (Fulgenzi, Spalanzani, Laugier, & Tay, 2010)

Modificaciones fueron realizadas para optimizar este algoritmo, una de las principales es tornar en cuenta la respuesta no holonómica planteada en el planificador local. Es así que se ha desarrollado el algoritmo llamado Non-Holonómic Risk RRT o NHR-RRT, este algoritmo se complementa con la evasión de obstáculos dinámicos e integra en el algoritmo de navegación y percepción estéreo del UGV.

4.3.3. Algoritmo de Planificación

Non-Holonómic Risk RRT es un algoritmo que trae como esencia a un árbol, el cual posee una raíz, crece por ramas hacia puntos aleatorios en el entorno, con la búsqueda de alcanzar un punto específico y generar un fruto. Este fruto puede ser o no el destino final, en el caso de no ser el destino final este fruto se convierte en una nueva raíz a fin de crecer hacia el destino. Un árbol crece por donde posee espacio, evita caminos oscuros en los que colisiona con obstáculos. Además, el árbol posee un tiempo de vida tras el cual su mejor fruto busca alcanzar el objetivo final.

Esta analogía da pie al desarrollo del Algoritmo 4 mostrado en la (Figura 113). Este algoritmo fue desarrollado como una función en la cual como se tiene claro es necesario tener como parámetro inicial el mapa instantáneo del entorno. Es allí donde el árbol de trayectorias crece. Consecuentemente la función retorna como parámetro de salida la trayectoria o el vector de nodos llamados RUTA, por la cual el UGV seguirá su camino.

Algoritmo 4: Real Time Risk RRT Estrella - RT-Risk-RRT*()

```
   Input  : mapa
   Output: 𝒯
1  while tiempo-gastado() < tiempo-maximo do
2      puntaje-random = random();
3      if puntaje-random > stepSize then
4          while n_aux .riesgo <=umbral − riesgo do
5              n_aux .pose = RealRandomPose();
6              n_aux .riesgo = RiesgoNodo(n_aux) ;
7          end
8          q_rand = n_aux .pose ;
9      else
10         q_rand = q_final;
11     end
12     q_best = EscogerMejorNodo(q_rand);
13     Extender(q_best,q_rand);
14     tiempo-gastado(milisegundos)
15 end
16 𝒯 : EnconrtarRuta()
17 return 𝒯
```

Figura 113. Algoritmo Non-Holonómic Risk RRT

Es común que en los algoritmos de RRT se permita crecer al árbol por cierto número de iteraciones definidas, más eso se relaciona directamente con la complejidad del entorno. Resulta óptimo lo realizado en este algoritmo, al considerar que el árbol crezca durante un tiempo máximo. Esto se debe a que el entorno es parcialmente desconocido en tanto el vehículo explora la trayectoria conoce su próximo destino.

El algoritmo RRT básico busca un punto aleatorio que pertenece al espacio conocido, $q_{rand} \in X$. Luego busca el nodo más cercano y genera un nuevo nodo entre aquel nodo q_{near} y q_{rand} llamándolo q_{new}. En este algoritmo q_{rand} posee la probabilidad del 10% de ser escogida el punto final o meta como el lugar hacia donde crece el árbol. Esto permite mejorar la heurística o desempeño de la ruta, haciendo que su búsqueda sea voraz y asertiva como muestra en las líneas 2 a 12.

Uno de los aportes esenciales en este algoritmo radica en la función RealRandomPose() la cual obtiene el punto aleatorio donde se va a extender y colocar el nuevo nodo q_{new}. En (Naderi, Rajamaki, & Hamalainen, 2015) plantea el uso de una función de muestreo aleatorio que divide la probabilidad del muestreo aleatorio en tres formas; Lineal, Elíptico y Uniforme. Se utiliza esa base con ciertas modificaciones, el muestreo Uniforme es un muestreo totalmente aleatorio rr en el que cualquier punto del espacio puede ser empleado como q_{rand}. El muestreo lineal LineTo, resulta en entregar solamente puntos aleatorios entre el inicio y la meta, como si fuese una línea recta. Elíptico se optimizo en Romboide, creando un romboide aguijón que genera puntos en un área específica pero cercana a la meta deseada. En (91) se consideran dos parámetros para realizar la distribución aleatoria. Por un lado α es el parámetro entre 0 y 1 que define la probabilidad de que se escoja un punto en línea. Luego β define la probabilidad de que sea un muestreo Uniforme. Para el porcentaje restante se aplica la función Romboide.

$$q_{rand} = \begin{cases} LineTo(q_{final}) & si\ \ rr > 1 - \alpha \\ Uniform(X) & si\ rr \leq \frac{1-\alpha}{\beta} \\ Romboide(q_o, q_{final}) & otros \end{cases} \tag{91}$$

Es necesario entender que está definida el tipo de variable *pose*, misma que consiste en una pose de tres grados de libertad; $x, y \wedge \theta$. Posición y Orientación en dos dimensiones. Esta función esta descrita en la (Figura 114). Rand es una función de cpp que genera valores entre 0 y 100 por lo que resulta necesario normalizarla.

Algoritmo 5: RealRandomPose()

Output: *pose*

```
pose puntaje-random;
puntaje-random = rand() ;
real-random = puntaje-random /100;
if real-random > (1 − α) then
    real-random = LineTo(p_final);
else
    if real-random < (1 − α)/β then
        real-random = Uniform(p_final);
    else
        real-random = Romboide(p_inicial,p_final);
    end
end
return real-random
```

Figura 114. Algoritmo para obtener poses aleatórias

Como parte los algoritmos RRT siempre se encuentra la función que permite detectar colisión. En este caso particular como se ha mostrado la opción es realizar una función de riesgo que permite identificar el factor de riesgo que un nodo puede tener. En otras palabras, el UGV debe desplazarse por los nodos que le ofertan menor riesgo de colisión. Como se muestra en la (Figura 115) se considera al UGV como un rectángulo con un centro y cuatro vértices. Estos vértices poseen una transformación geométrica lo cual permite obtener las posiciones de cada vértice y posteriormente realizar una lista de puntos a los que se les cotejara junto con el mapa de grilla, evaluando cada tono gris.

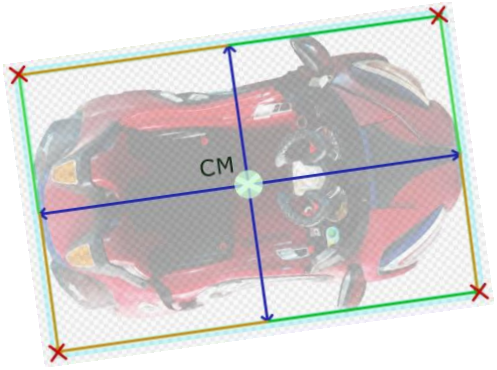


Figura 115. Geometría del vehículo adoptada en el algoritmo

El riesgo del nodo se evalúa punto por punto, tomando en cuenta que el riesgo se va acumulando. El riesgo es un valor normalizado por el valor llamado riesgo-neto y promediado entre todos sus valores. Esta función esta detallada en la (Figura 116), adicionalmente hay que considerar que si el promedio de riesgo es mayor a 1 se limita el valor para poseer control sobre el resultado.

Algoritmo 6: RiesgoNodo()

```
   Input  : nodo
   Output: riesgo
 1 largo = robotLength/2.0;
 2 ancho = robotWidth/2.0;
 3 celdas-riesgo =ListaDeRiesgo(IntensidadPose(esquinas));
 4 for i = 0 to celdas-riesgo.size() do
 5 |   riesgo = riesgo + celdas-riesgo [i]/riesgo − neto;
 6 |   if celdas-riesgo [i]/riesgo − neto < umbral − riesgo then
 7 |   |   riesgo = 1;
 8 |   |   return riesgo;
 9 |   end
10 end
11 riesgo = riesgo /i;
12 if riesgo > 1 then
13 |   riesgo = 1
14 end
15 return riesgo;
```

Figura 116. Algoritmo para determinar el riesgo de un nodo determinado

Como se muestra en la línea 12 del Algoritmo 4 de la (Figura 113) es indispensable no solo seleccionar el nodo más cercano q_{near}, sino más bien el mejor nodo o el nodo con más cercanía al destino q_{best}. En la (Figura 117), observamos que esta función tiene como atributo de entrada la meta o el nodo al cual se desea extender el árbol. En la línea 1 se hace uso de una función llamada PesoNodo() la cual evalúa el peso que posee la meta en función de la raíz y el destino. En (92) se representa el peso y con entre menor peso posee el nodo actual mejor es el nodo.

$$W = riesgo * \frac{\sqrt{(meta.x - nodo.x)^2 + (meta.y - nodo.y)^2}}{10} \quad (92)$$

Algoritmo 7: EscojerMejorNodo()

```
Input  : meta
Output: nodo
mejor-peso = PesoNodo(n_root, meta);
for i = 0 to candidatos.size() do
    n_actual = candidatos [i];
    peso-acual = PesoNodo(n_actual, meta) ;
    if peso-acual <= mejor-peso then
        n_best = n_actual;
        mejor-peso = peso-acual;
    end
end
return n_best;
```

Figura 117. Función que determina cual es el mejor nodo de entre los candidatos

El vector llamado candidatos contiene todos los nodos ya establecido, para el primer bucle el único nodo asignado es la raíz y es así que desde allí parten los demás nodos a expandirse. El nodo con menos peso es el nodo escogido, si existe mismo peso de todas formas se escoge el último nodo así se prevé que se escoja los nodos más recientes y el árbol tenga mayor profundidad, es decir las ramas contengan más nodos.

La estructura llamada *nodo* es una estructura de un nodo que contiene las siguientes variables internas:

Tabla 14
Variables internas de un nodo

Tiempo	El momento en el que el nodo fue creado
Pose	La posición y orientación
Control previo	Velocidad y ángulo previos que se realizaron
Padre	El nodo anterior desde el cual se extendió este.
Hijos	Todos los nodos que se extendieron a partir de este.
Controles	Todas las posibles posiciones alcanzables con el planificador local.
ID	Identificador de profundidad a partir de la raíz
Riesgo	El valor de riesgo entre 0 y 1.

Una vez que se posee el mejor nodo se procede a extender es decir a obtener el q_{new} a la lista de candidatos. Es decir, se genera un vector de nodos que serán base para escoger la trayectoria óptima. En la (Figura 118) podemos ver a detalle esta función, en la cual como parámetro de

entrada se encuentra el nodo q_{best} y la meta parcial llamada q_{rand}. Un nodo cuando es creado automáticamente se crea con la combinación de todas sus acciones de control, es decir las combinaciones de ángulo de giro y velocidad. Por ejemplo, de lo previamente definido el ángulo podría ser 30° y la velocidad puede ser 50cm/s.

Algoritmo 8: Extender()

```
   Input  : nodo, meta-random
1  for i = 0 to nodo.controles.size() do
2      valor-x = rand(0,1);
3      mejor-puntaje = 0;
4      p_esperada = CinematicaRobot(nodo, nodo.controles[i], delta − t);
5      p_inesperada = CinematicaRobot(nodo,nodo.controles[i],
         delta − t∗valor-x);
6      nodo-aux .pose = p_inesperada;
7      riesgo-inesperado = RiesgoNodo(nodo-aux);
8      puntaje-control = CalcularPuntajeControl(p_esperada,riesgo-
         inesperado,meta-random);

9      if puntaje-control >mejor-puntaje then
10         mejor-puntaje = puntaje-control;
11         mejor-indice-control = i;
12     end
13 end
14 candidatos .pushback(nodo);
```

Figura 118. Algoritmo que permite el crecimiento del árbol RRT

En las líneas 4 y 5 se observa la función principal que sigue a pie lo desarrollado en el planificador local, anteponiendo (75), (76) y retornando el punto al cual llegaría con esa acción de control. Para lo cual este punto es el nuevo nodo q_{new}. Como la cinemática guarda relación con el tiempo se establece un tiempo fijo de 2 segundos a los cuales se genera la pose esperada. Una pose inesperada también se genera de forma aleatoria lo cual permite verificar que en el trayecto no exista colisión.

De todas las combinaciones de control se calcula un puntaje con la función CalcularPuntajeControl(), la cual considera el riesgo de una pose inesperada, la pose esperada y la meta aleatoria para considerar cual es el nodo que llegue más cerca y que llegue con menor

riesgo. Es así que el nodo con mayor puntaje sobresale y se asigna como el nuevo nodo q_{new}. Este nodo es agregado al vector de candidatos.

Una vez terminado el tiempo y con una buena cantidad de nodos candidatos, es decir con varias ramas y varios nodos se procede a encontrar la ruta conforme al algoritmos mostrado en la (Figura 119). De ser el caso que ya se realizó una trayectoria previa, esta se borra con la función clear(). Se reutiliza la función EscogerMejorNodo() en la línea 2, pero en este caso se escoge el nodo más cercano hacia la meta final. Como se mostró en la anterior tabla el ID es el identificador y la raíz es el número cero. De esta forma por medio de un bucle se empieza a retroceder en los nodos por medio de los padres. Cada padre lleva su trayectoria, su acción de control y su riesgo. En la línea 7 se ve insertada la trayectoria cada nodo extraído se coloca al inicio, para de esta forma tener una trayectoria ordenada. Resulta una búsqueda del padre en los candidatos para convertirlo en hijo y así buscar nuevamente el pare hasta dar con la raíz.

Algoritmo 9: EcontrarRuta()

```
1  trayectoria.clear;
2  n_mejor = EscogerMejorNodo(n_meta.pose);
3  n_actual = n_mejor;
4  trayectoria .pushback(n_actual);
5  while n_actual .id > 0 do
6  |   n_actual = n_actual .padre;
7  |   trayectoria.insert(trayectoria.begin(),n_actual);
8  |   for i = 0 to candidatos.size() do
9  |   |   if candidatos [i].pose ==n_actual .pose then
10 |   |   |   n_actual = candidatos [i]
11 |   |   end
12 |   end
13 end
```

Figura 119. Algoritmo para recuperar la ruta alcanzada

El algoritmo NHR-RRT no necesariamente alcanza la meta propuesta desde un inicio, eso permite que sea un algoritmo dinámico y adaptable a las circunstancias del mapa. Siempre se busca obtener una trayectoria que acerque el UGV hacia la meta.

4.4. Navegación Autónoma

La navegación autónoma se conjuga con la percepción tridimensional en un solo algoritmo en el cual existen aspectos que se desarrollaron haciendo uso de herramientas para paralización de procesos como el multithreading. Esto permite que mientras se ejecuta un proceso como la planificación de trayectorias se aproveche en otro como la percepción tridimensional. Sin embargo, esta acción de procesos paralelos limita su uso al no poder compartir variables de forma libre.

Para iniciar el programa es necesario poseer la pose inicial del UGV, así como también la pose de destino o meta como se detalla en el algoritmo completo de la (Figura 120). Luego el primer paso es inicializar las variables en lo que se destaca que se coloca en el vector de candidatos al nodo raíz, además se asigna la pose final y el riego de la misma. Como se anticipó se ejecuta el algoritmo SLAM() dando cuenta que se genera el mapa desde el punto inicial.

Algoritmo 3: Navegación y Percepción Estéreo en UGV

```
   Input : p_inicial, p_final
 1 Inicializar();
 2 mapa = SLAM(p_inicial);
 3 IniciarPercepcion3D();
 4 repeat
 5 |  T = RT-Risk-RRT*(mapa)
 6 |  for z = 0 to T.size() do
 7 |  |  codigo[z] = Codificar(T[z].velocidad, T[z].angulo);
 8 |  end
 9 |  SerialArduino(z, codigo);
10 |  n_fruto = MejorNodo(p_final);
11 |  Reiniciar(n_fruto);
12 |  ActualizarNodos();
13 |  if FueraDeTrayectoria() then
14 |  |  n_root .pose = p_robot;
15 |  end
16 |  nuevo-mapa = SLAM(p_robot);
17 |  mapa = UnirMapas(mapa, nuevo-mapa);
18 until Meta-Alcanzada();
19 TerminarPercepcion3D();
```

Figura 120. Algoritmo para la navegación autónoma

La percepción tridimensional es uno de los procesos que se realiza de forma paralela, es decir en la línea 3 se inicia y continúa a la par del resto del algoritmo. El algoritmo de navegación se ejecuta hasta que la meta sea alcanzada. La función Meta-Alcanzada() revisa constantemente si el último nodo fruto llegó o está muy cerca de la meta, existe un umbral del cual se toma como si fuese una situación de meta alcanzada. Esto se realiza para optimizar ya que al ser un UGV no holonómico requiere mayor trabajo para alcanzar una pose específica fielmente.

La consulta entre procesos paralelos se logra por medio de la llamada a funciones, gracias a ello es que se llama a la función FueraDeTrayectoria() en las líneas 13 a 15. Esta función permite obtener la pose del robot y verificar si el UGV se salió de la trayectoria estipulada. Esta función forma parte del proceso de percepción tridimensional. En la (Figura 121) se observa cómo funciona la pose del robot, esta se logra a partir de una fusión de datos entre el sensor inercial y la estéreo visión.

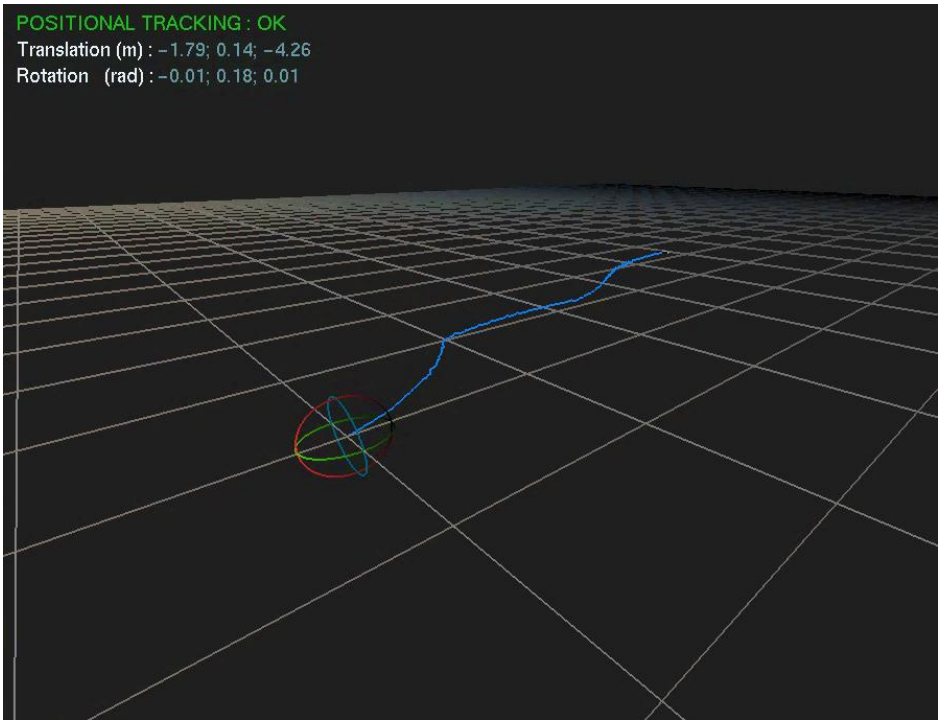


Figura 121. Trayectoria del UGV seguida por la cámara en tiempo real

Continuando con el algoritmo de la Figura 120, una vez obtenida la trayectoria es necesario codificar la trayectoria para enviarla al dispositivo controlador Arduino, este a su se comunica con el driver que acciona los motores. La codificación se realiza para que cada acción de control, velocidad y ángulo de giro, posean un código ASCII. Por ejemplo, una velocidad de -40cm/s y un ángulo de giro de -20° equivalen al carácter 'a'. Todos los códigos de la trayectoria se almacenan en un vector y posterior se envía por medio de comunicación serial con la función SerialArduino(). De la línea 10 a 12 del algoritmo lo que se realiza es actualizar el árbol, reiniciarlo en un nuevo fruto, esto significa que el fruto es la nueva raíz y que lo demás es olvidado. Finalmente, antes de retomar el ciclo del planificado de trayectorias se obtiene un nuevo mapa parcial el cual se lo sobrepone con el anterior mapa.

Si bien el entorno es un entorno controlado, se debe poseer una funcionalidad esencial en la navegación autónoma. Esta función es llamada Navegación Segura y lo que hace es permitir que, si existen obstáculos dinámicos, estos sean identificados a tiempo. Como se muestra en la Figura 122, este proceso se realiza por medio de otra paralización y el uso de una función llamada ProfundidadSegura(). La profundidad segura es un proceso que como se vio en el capítulo pasado la percepción de profundidad obtenida por la cámara estéreo permite establecer si un objeto es cercano, esta profundidad se compara con un valor umbral dando la capacidad de ejecutar el algoritmo principal o pararlo.

```
1 Thread Navegacion Segura
2 |  if ProfundidadSegura() then
3 |  |  run(Algoritmo1);
4 |  else
5 |  |  stop(Algoritmo1);
6 |  end
7 end
```

Figura 122. Función de navegación por profundidad segura en paralelo

4.5. Control de Movimientos

El control de movimientos (Aguilar, Manosalvas, Guillén, & Collaguazo, 2018), se realiza directamente sobre los motores, y es necesario poder controlar la activación, la velocidad y la dirección de giro que estos tienen (Aguilar & Angulo, Real-Time Model-Based Video Stabilization for Microaerial Vehicles, 2016), teniendo en cuenta que estos tienen que realizar cada actividad en función del tiempo, para llevarlo a cabo es necesario usar una tarjeta que tenga características de control y además que soporte trabajar con las características de potencia de los motores.

Para las necesidades de control de los motores es necesario comprender el funcionamiento de cada uno de ellos en el sistema de navegación completo. Luego de un previo análisis se ha elegido a la tarjeta Arduino Mega debido a su versatilidad y a la gran cantidad de información disponible en la actualidad además de su fácil uso.

Para poder actuar sobre los motores de forma sencilla se usa el Monster motor shield, debido a sus características encaja perfectamente sobre la mayoría de las tarjetas Arduino sin necesidad de realizar algún cableado extra lo que facilita aún más su uso en la (Figura 123) se ilustra el montaje de ambas tarjetas.

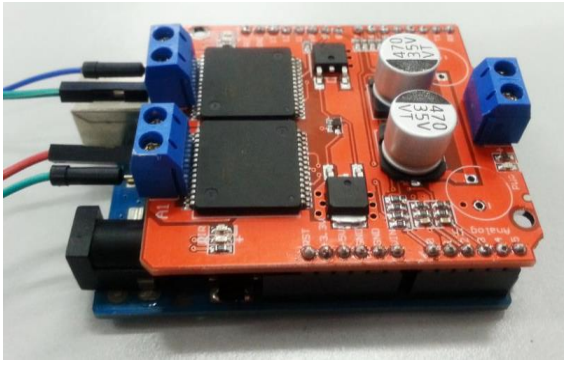


Figura 123. Montaje de Monster motor shield sobre Tarjeta Arduino UNO
Fuente: (Instructables, 2018)

Las características y las grandes ventajas de control para los motores que nos brinda este driver es excelente, además tiene la disponibilidad de ser usado con dos motores, lo que lo hace perfecto para realizar este trabajo debido a que se tienen que controlar el motor trasero que se encarga de la dirección del vehículo y también un motor delantero que controla el ángulo de las llantas posteriores, los detalles de uso de los pines de esta tarjeta se ilustran a continuación en. La (Figura 124) y la (

Tabla *15*).

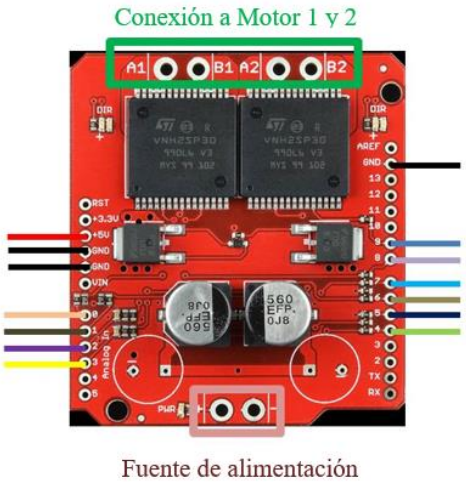


Figura 124. Configuración de pines de Monster motor shield

Fuente: (Instructables, 2018)

Tabla 15

Detalle de pines de Monster motor shield

Pin	Detalle
A0	Activar pin para el motor 1
A1	Habilitar el pin para el motor 2
A2	Sensor de corriente para el motor 1
A3	Sensor de corriente para el motor 2
D7	Girar en el sentido horario (CW) para el motor 1
D8	Girar sentido anti-horario (CCW) para el motor 1
D4	Girar en el sentido horario (CW) para el motor 2
D9	Girar sentido antihorario (CCW) para el motor 2
D5	PWM para el motor 1
D6	PWM para el motor 2

Fuente: (Instructables, 2018)

4.5.1. Algoritmos de control.

Conociendo los pines y la funcionalidad de cada uno de ellos, se realiza la programación, se parte por comprender las acciones de nuestra planta que en este caso viene a ser el vehículo (Orbea, y otros, 2017). Mediante pruebas experimentales se mide los parámetros máximos y mínimos del movimiento, para obtener estos valores de velocidad se varió el valor de salida de PWM en la rueda que realiza la tracción para el movimiento, se parte del valor más alto (255) y se mide la velocidad tanto en avance y retroceso, y se va reduciendo el valor de PWM hasta que el movimiento sea casi nulo o inútil para realizar algún movimiento.

En el caso de la dirección, al tratarse de un motor DC si se lo activa y este llega al límite de giro el motor seguiría trabajando, pero sin lograr nada más que esfuerzo y trabar al motor, cosa que pueda afectar e incluso dañar al mismo.

4.5.1.1. Control del ángulo

Para poder controlar el ángulo en el cual se colocará el motor delantero se usó un controlador difuso, debido a las características del sistema y la dificultad de encontrar la función de transferencia de la planta matemáticamente, además se toma en cuenta la versatilidad de este tipo de control y la necesidad de contar con control con retro alimentación.

El objetivo es usar la MPU-6050 para conocer el ángulo en el cual se coloca el volante que está conectado directamente con el motor y el sistema de giro, de esta manera usar este sensor como retro alimentación para actuar directamente sobre el actuador y entregar el ángulo deseado.

El planificador de trayectorias entregará los ángulos a los cuales se debe desplazar el vehículo, y el controlador tiene que accionar los motores en la dirección que sea necesaria hasta que el

error sea cero entre la salida del sensor y el ángulo deseado, y mantener este ángulo durante el trayecto, incluso pese a perturbaciones, el sistema de control planteado se ilustra en la (Figura 125).

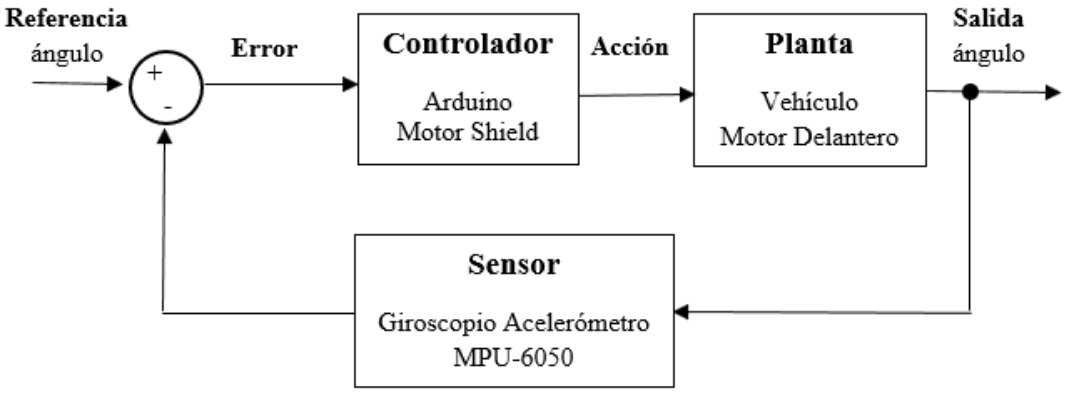


Figura 125. Diagrama del sistema de control del ángulo

Como se ha planteado un controlador difuso, es necesario tener los parámetros a los cuales va a funcionar y como va a actuar el controlador sobre nuestro sistema. Primero para lograr cada ángulo dependiendo de la posición actual del volante hacia la siguiente se tendrá que entregar un nivel de voltaje y de tiempo hasta que el error sea cero pero mientras más cercano sea al punto de set point se tendrá que hacer acciones cada vez menos bruscas además para que el control difuso funcione correctamente se toma dos entrada la entrada del error y la entrada de la variación del error y como salida del controlador difuso se tiene el valor de voltaje y el sentido de giro con el que actuarán los motores.

Los parámetros difusos tanto para las entradas y salidas son cinco los cuales efectuarán los cálculos de acuerdo con las funciones de membresía que se le ha otorgado, la simulación y creación de este controlador fue creada con la ayuda del software de MATLAB y su herramienta *Fuzzy Logic Designer* las reglas asignadas se muestran en la (Tabla *16*).

Se ha seleccionado:

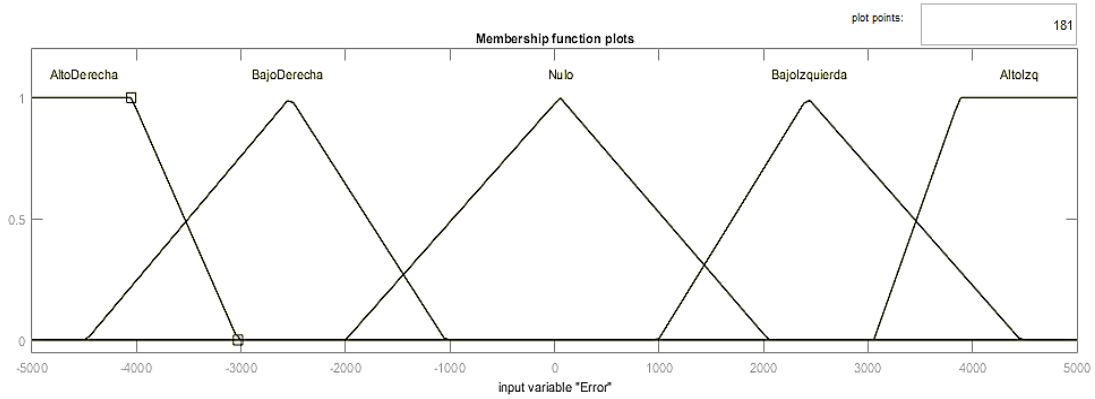


Figura 126. Función de membresía para la entrada del error

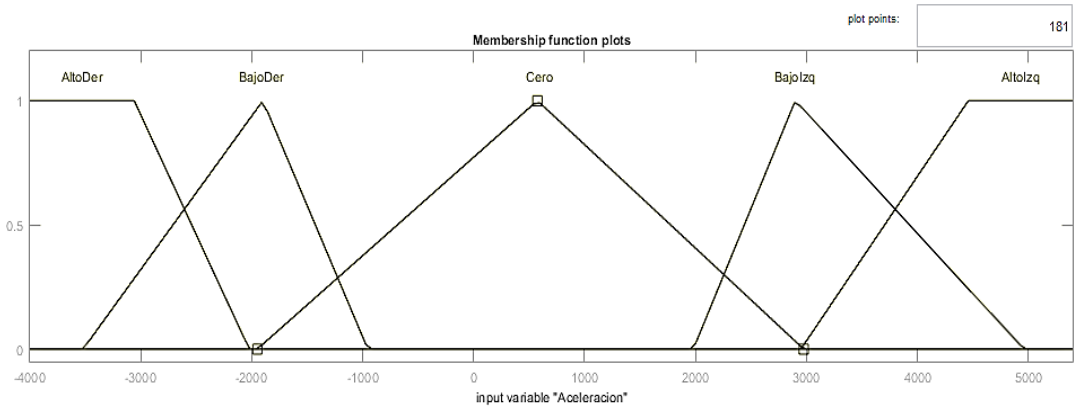


Figura 127. Función de membresía para la entrada de la aceleración

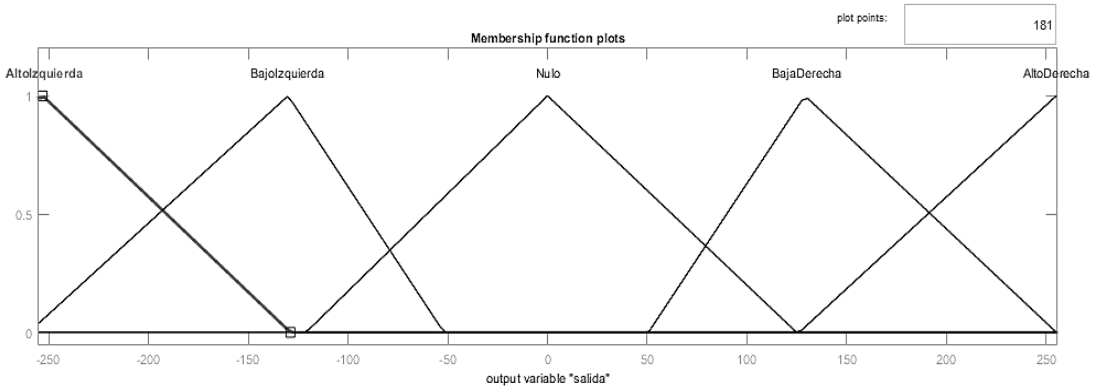


Figura 128. Función de membresía para la salida de voltaje en PWM

Con las funciones de membresía tanto para las entradas como las salidas se procede a realizar las reglas con las cuales a responderá nuestro controlador, se ha tomado 5 niveles tanto para las entradas como las salidas lo que permite tener un total de 25 reglas de control que se ilustran en la (Tabla *16*), y para comprender mejor el funcionamiento ante cada una de ellas se puede usar la ayuda del software en el que lo creamos, tanto para probar las reglas ante las diferentes entradas, así como también nos presenta el grafico de control mostrados en la (Figura 129).

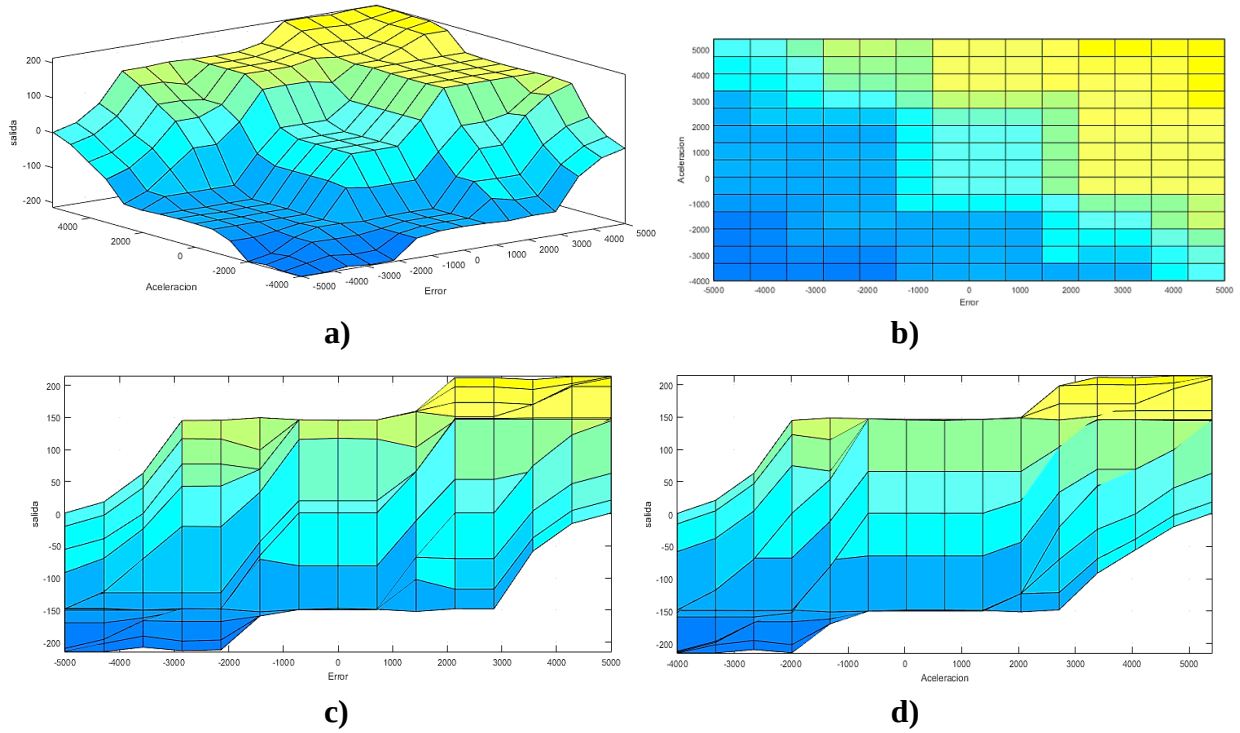


Figura 129. Superficie de control resultante de las reglas del control difuso propuesto
a) Vista en 3D, b) Vista superior (Aceleración, Error),
c) Vista lateral (Salida, Error), d) lateral (Salida, Aceleración)

Tabla 16
Reglas del controlador difuso

		Aceleración				
		AltoDer	**BajoDer**	**Nulo**	**BajoIzq**	**AltoIzq**
Error	**AltoDerecha**	AI	AI	BI	BI	N
	BajoDerecha	AI	BI	BI	N	BD
	Nulo	BI	BI	N	BD	BD
	BajoIzquierda	BI	N	BD	BD	AD
	AltoIzquierda	N	BD	BD	AD	AD

Como se ha especificado que por defecto y con el fin de reducir el procesamiento de planificación de trayectorias se usará únicamente valores de ángulos predefinidos, se calcula el tiempo que se tarda el vehículo en llegar al ángulo deseado, el cálculo no es más que un promedio tras varias pruebas ante los diferentes giros, para obtener el ángulo resultante, se usó la definición de ángulo de Ackerman presentada anteriormente. Los ángulos planteados son 5 en total, -20, -10, 0, 10 ,20. Los ángulos predefinidos fueron definidos luego de realizar un análisis de máximas curvaturas. Para conocer los valores limites se fue graficando las trayectorias a ángulos límites y luego se pudo conocer el radio de giro, y los ángulos máximos de curvatura.

Figura 130. Determinación de radio de giro
y de ángulos máximos de curvatura

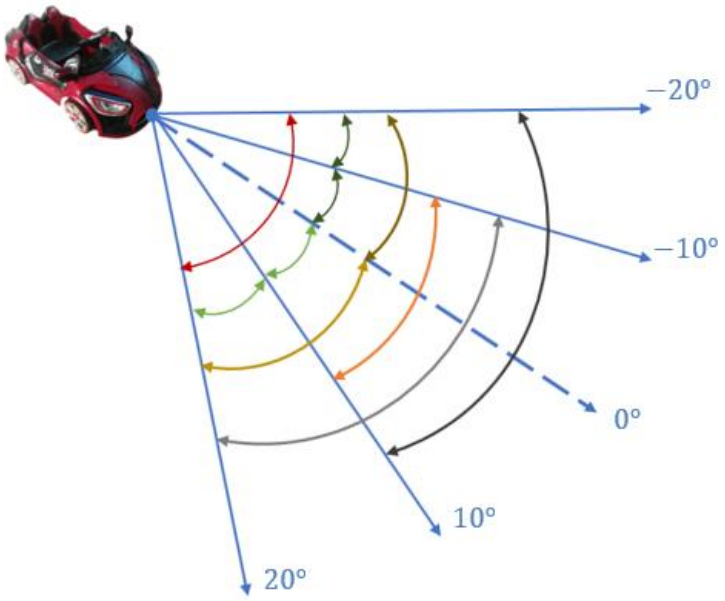


Figura 131. Desplazamientos angulares
entre ángulos fijos predefinidos

Debido a que no se cuenta con un sensor para conocer la posición del volante en ese momento, y hacia dónde va a girar el vehículo, se ha creado un algoritmo que permita realizar los giros necesarios, pero a la vez desde el giro actual poder ir hacia los demás giros, para ello es necesario etiquetar el valor actual donde se encuentra y por medio de condicionales para cada ángulo poder realizar el movimiento, mientras más lejano sea el siguiente ángulo al que se quiere llegar, se necesitará más tiempo de giro del motor, por lo cual se definen los valores de tiempo que se tarda en llegar al siguiente ángulo, finalmente se obtienen tres tipo de giros con valores diferentes que son los siguientes:

- Giro de lado a lado (40 grados|)
- Giro de lado a la mitad del otro lado (30 grados|)
- Giro medio (20 grados)
- Giro al ángulo adyacente (10 grados)

Con esto no hay que preocuparse de conocer la posición actual debido que al finalizar el giro se guarda en una variable el valor del ángulo actual en el que se encuentra y luego se procede a comparar el número actual para acceder al siguiente caso.

4.5.1.2. Control de velocidad

De manera similar a como se ha desarrollado control de ángulos, para reducir tiempos de cálculo y procesamientos en el planificador se ha limitado a tener tres niveles de velocidades constantes por dirección, es decir que tendremos un total de 6 velocidades posibles 3 positivas y 3 negativas, las negativas indican una dirección de retroceso del vehículo. Previamente se realizó pruebas para conocer cuál es la máxima velocidad a la que puede viajar el vehículo y las mínimas y a partir de estos datos más otras series de prueba de respuesta se llegó a considerar las siguientes velocidades.

- Velocidad de avance máxima: 50 cm/s
- Velocidad de avance media: 40 cm/s
- Velocidad de avance mínima: 30 cm/s
- Velocidad de retroceso máxima: -50 cm/s
- Velocidad de retroceso media: -40 cm/s
- Velocidad de retroceso mínima: -30 cm/s

Debido a que el Arduino trabaja con PWM para entregar velocidad, se tuvo que expresar estos valores en el Arduino en valores de ciclo (0 - 255), usando una transformación simple.

4.6. Comunicación entre controlador y planificador.

Para que el controlador sepa que acciones realizar, es decir que ángulo colocar y a qué velocidad desplazarse es necesario que obtenga los valores que estima previamente el

planificador de trayectorias, para que de esta forma el vehículo pueda llegar a la posición estimada usando únicamente los datos de ángulo y velocidad para realizar su objetivo, se usó un modo de comunicación serial, aprovechando que el Arduino tiene una comunicación directa con el computador por medio del puerto USB 2.0 de donde tiene su alimentación, entonces para poder comunicar ambos se usó un algoritmo conocido como Arduino-serial que se encuentra disponible en (Kurt, 2015), y del cual se realizó una serie de modificaciones para cumplir con los requerimientos del proceso.

Este algoritmo usa una librería denominada como “Arduino-serial-lib”, y una función externa denominada como “mongoose”, el tipo de comunicación que realiza es cliente -servidor. La estructura de comunicación es bastante simple, y su funcionamiento se muestra en la (Figura 132).

Debido a que es una comunicación seria, y no se puede enviar los datos específicamente con todo su contenido como se quisiera, se procede a realizar una codificación de datos, para que en un único envió se pueda enviar varias características para el control del vehículo y de esta manera optimizar la comunicación y el tiempo de ejecución.

Para realizar la codificación se toma en cuenta los datos que se van a enviar, son 6 valores diferentes de velocidad y 5 valores diferentes de ángulos, se utilizan caracteres comenzando por ‘a’ hasta el carácter ‘~’, para enviar los datos y se dividen de 6 en 6 para cambiar el ángulo, y el cambio de ángulo van desde el menor al mayor, de igual manera de los 6 caracteres se ordena la velocidad de menor a mayor, como se ilustra en la Tabla 16.

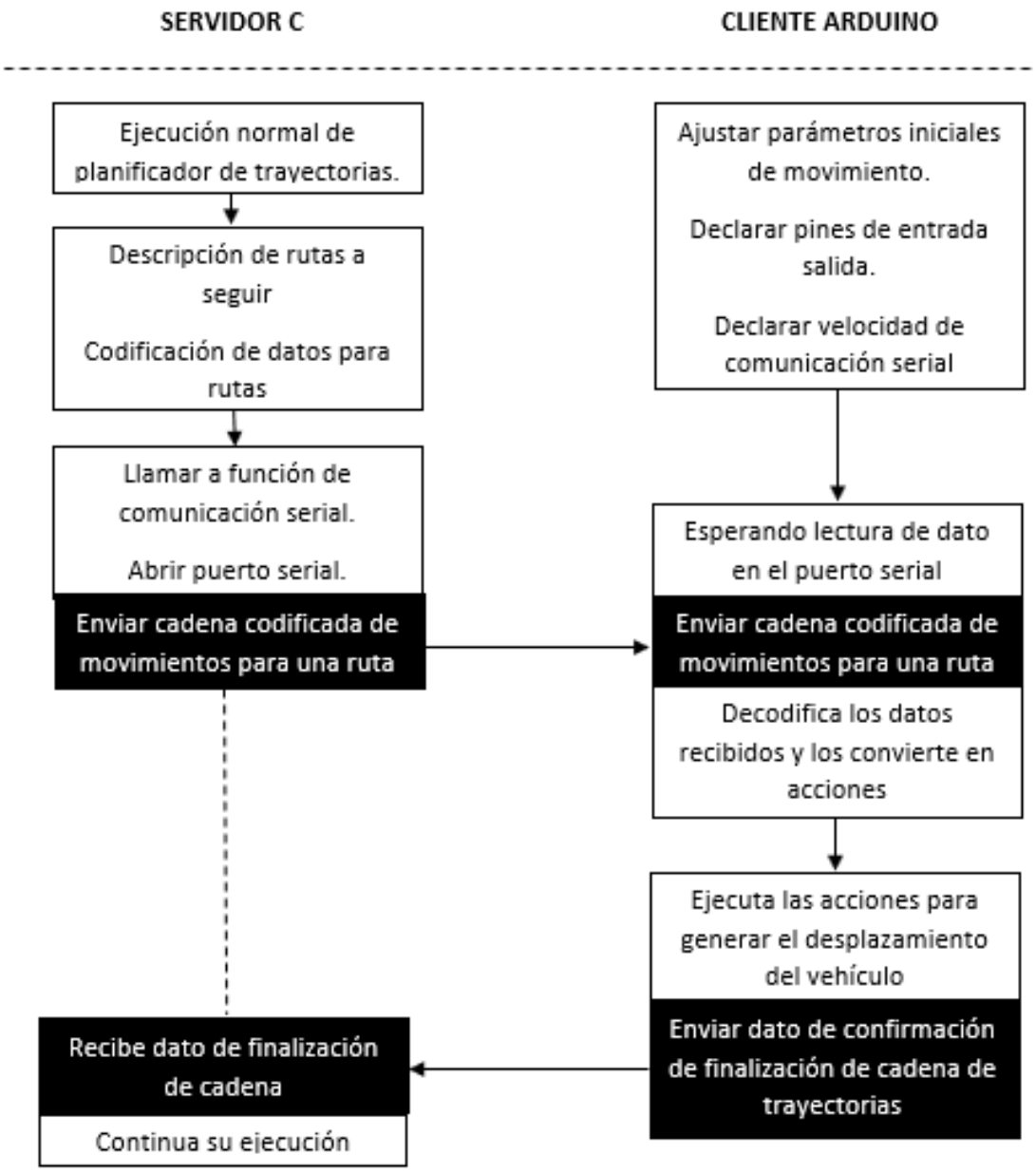


Figura 132. Diagrama de secuencias de proceso de transferencia de datos entre Planificador de trayectorias (Servidor) y el controlador (Arduino).

Tabla 17
Asignación de caracteres según Angulo y velocidad.

			Ángulos				
			Izquierda		Centro	Derecha	
			-20	-10	0	10	20
Velocidades	Atrás	-50 m/s	a	g	m	s	y
		-40 m/s	b	h	n	t	z
		-30 m/s	c	i	o	u	{
	Adelante	30 m/s	d	j	p	v	|
		40 m/s	e	k	q	w	}
		50 m/s	f	l	r	x	~

Para poder enviar la codificación anterior, se realiza una ecuación la cual permita calcular fácilmente el valor a ser enviado hacia Arduino, para poder usar una ecuación se procede a cuantificar cada situación según muestra la Tabla 18.

Tabla 18
Numeración para ecuación de codificación

Velocidades (m/s)	Valor numérico	Ángulos (°)	Valor numérico
-82	1	-20	1
-67	2	-10	2
-52	3	0	3
52	4	10	4
67	5	20	5
82	6		

$$Codigo = Codigo + 6(a-1) + v \quad (93)$$

A estos valores previos a ser enviados se les suma 96 para que su equivalencia en código ascii sea igual al de la letra a, de esta manera se puede enviar una cadena de trayectorias en un solo

dato de tipo String, por ejemplo “abc” enviamos al vehículo a realizar 3 acciones diferentes y a alcanzar 3 coordenadas durante su trayecto.

En el programa de Arduino es necesario decodificar este valor para conocer la velocidad y ángulo que se han asignado como tareas. Cuando se recibe el dato se recibe el carácter en código ascii, y primero hay que numerizarlo, para ello únicamente le restamos 97 y lo tendremos numerado desde 1, luego para obtener el valor numérico de ángulo dividimos el valor entre 6 y este será el ángulo, y el resto será el índice de velocidad.

$$valor = ak \tag{94}$$

$$IndAngulo = valor/6 \tag{95}$$

$$IndVelocidad = valor\ \%\ 6 \tag{96}$$

Tabla 19
Numeración para decodificación

Velocidades (m/s)	Valor numérico	Ángulos (°)	Valor numérico
-82	0	-20	0
-67	1	-10	1
-52	2	0	2
52	3	10	3
67	4	20	4
82	5		

Con los datos separados y decodificados se puede realizar las acciones de movimiento según como se haya indicado en el planificador de trayectorias, al finalizar toda la cadena, es decir toda la trayectoria que e ha enviado se devuelve el valor ‘1’ para que el programa pueda continuar, en su próxima trayectoria que se envíe no será necesario volver a abrir el puerto serial debido a que ya se abrió previamente.

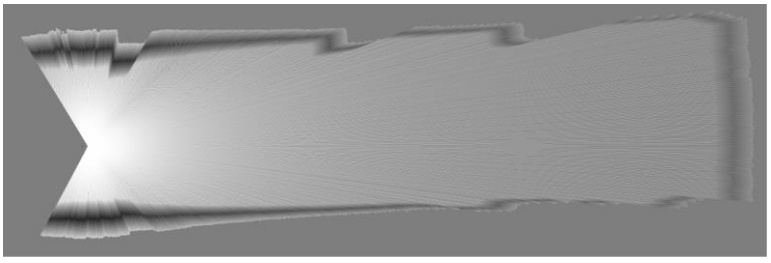
4.7. Optimización del Algoritmo

La planificación de trayectorias guarda ciertos parámetros variables como ya se lo ha dicho. Es así que resulta necesario identificar la mejor respuesta de cada parámetro a fin de tener la mejor trayectoria con menor tiempo y que cumpla las características de navegación segura.

4.7.1. Optimización del Mapa 2D

La generación del mapa 2D está directamente relacionada con el número de escaneos realizados, esto es atribuido al parámetro MAXSCANS por lo que resulta necesario establecer un valor a este parámetro. Es así que se realizan mapas de un entorno interior perteneciente a una cochera de 15x2.7 m, para lo cual se fija el parámetro en: 1, 5, 10, 15, 20. Como se muestra en la (Figura 133).

Por simple inspección se puede observar que entre 15 y 20 escaneos el mapa es igual. Por lo tanto, se establece que este parámetro debe ser establecido en 15 ya que más escaneos puede surtir efecto en el tiempo de ejecución. No se puede establecer por ningún motivo un mapa con menos de 10 escaneos, debido a su pobre respuesta.



(a)

CONTINÚA ⇨

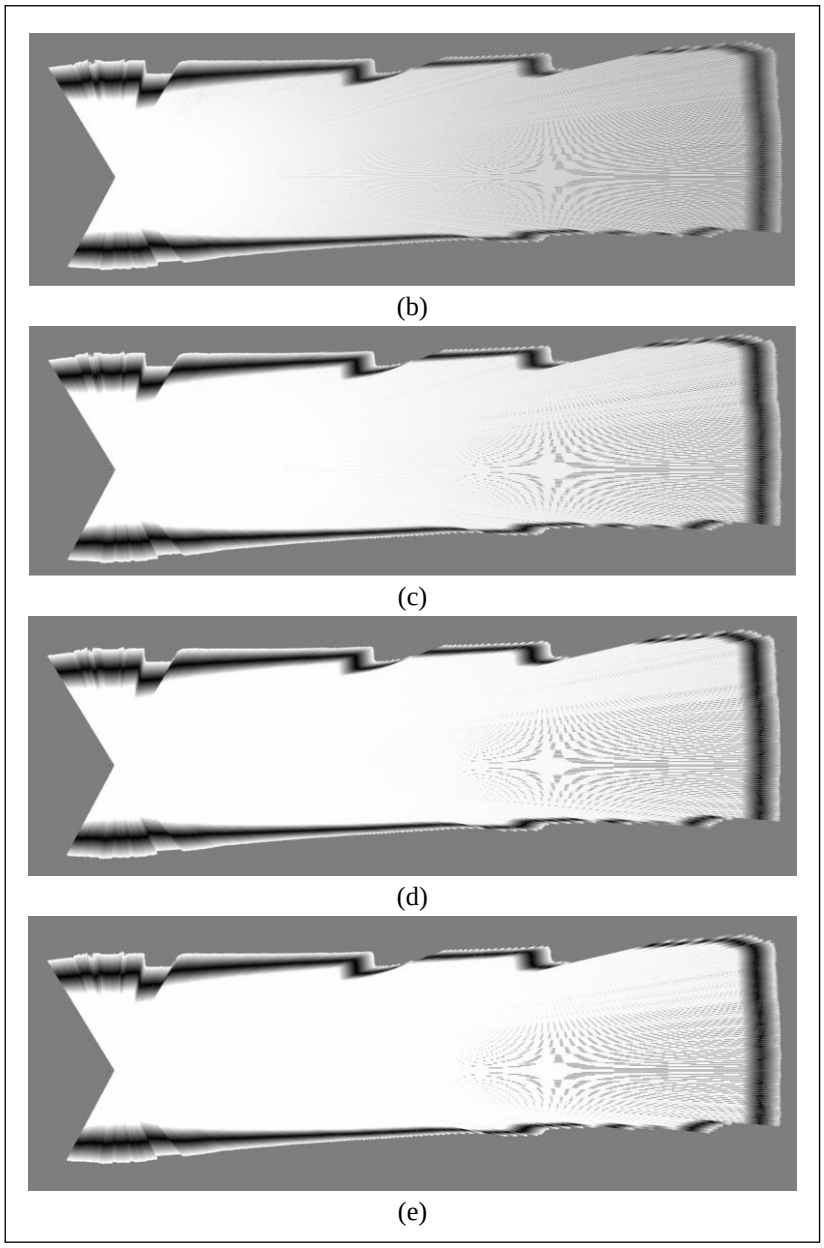


Figura 133. Respuesta del Mapa 2D en respuesta al número de escaneos.

4.7.2. Optimización de Pose Aleatoria

Previamente se ha definido a los parámetros α y β como variables que determinan las formas de la probabilidad del muestreo aleatorio. Muestreo Lineal, Uniforme y Romboide conforman

una probabilidad del 100% es así que se realizan pruebas con diferentes probabilidades en las tres formas dando paso a comportamientos diferentes de la planificación de trayectorias.

Para desarrollar las pruebas se definieron 8 casos los cuales se detallan a continuación:

Tabla 20
Casos para optimización de pose aleatoria

	Caso 1	**Caso 2**	**Caso 3**	**Caso 4**	**Caso 5**	**Caso 6**	**Caso 7**	**Caso 8**
Alfa	1	0.7	0.5	0.4	0.4	0.4	0.3	0
Beta	1	1000	500	2	4	8	100	200
Lineal	100%	70%	50%	40%	40%	40%	30%	0%
Romboide	0%	30%	50%	40%	50%	55%	70%	100%
Uniforme	0%	0%	0%	20%	10%	5%	0%	0%

Existen variables de las que dependen los parámetros, estas variables críticas inciden en la decisión final sobre cuál sea el caso para la distribución de poses aleatorias. Estas variables se definen para estos parámetros, y se utilizarán de igual manera en la evaluación de los siguientes parámetros. Una de las variables es el tiempo, a menor tiempo el algoritmo resulta más efectivo. Es así que se llegó a obtener las respuestas de la (Figura 134), los tiempos no resultan diferir mucho entre si y los dos mejores casos resultan ser Caso 4 y Caso 7 respectivamente. Cabe señalar que la meta del UGV se sitúa a 3 metros en línea recta y sin obstáculo alguno, para la evaluación de este y siguientes parámetros.

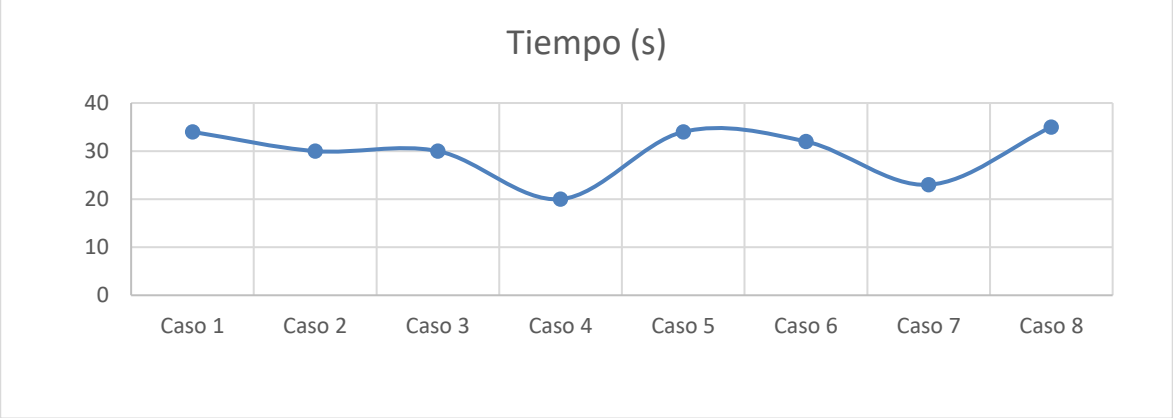


Figura 134. Variable de tiempo en pose aleatoria

Otra variable es el error de pose del UGV, esta pose tiene cuatro valores; Pose Meta (A donde debe llegar), Pose Calculada (A donde estima llegar), Pose Robot (Donde determina estar el UGV) y Pose Real (Donde en realidad se encuentra), por lo cual se consideró tres errores. El primer error es llamado Error de Cálculo mostrado en la (Figura 135), determina el error entre la pose calculada y la pose final o meta. El segundo error es llamado Error de Exactitud mostrado en la (Figura 136), determina el error entre la pose calculada y la pose real. El tercer error es llamado Error de Precisión mostrado en la (Figura 137), determina el error entre la pose real y la pose del robot.

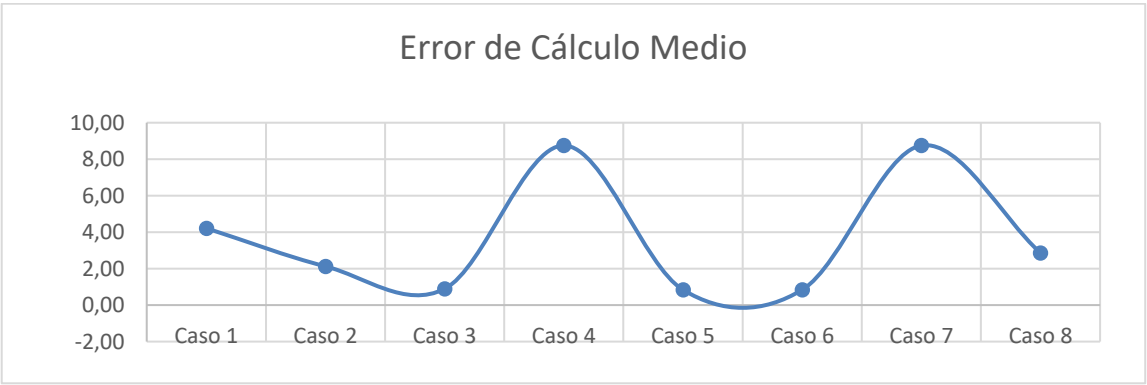


Figura 135. Error de cálculo para pose aleatoria

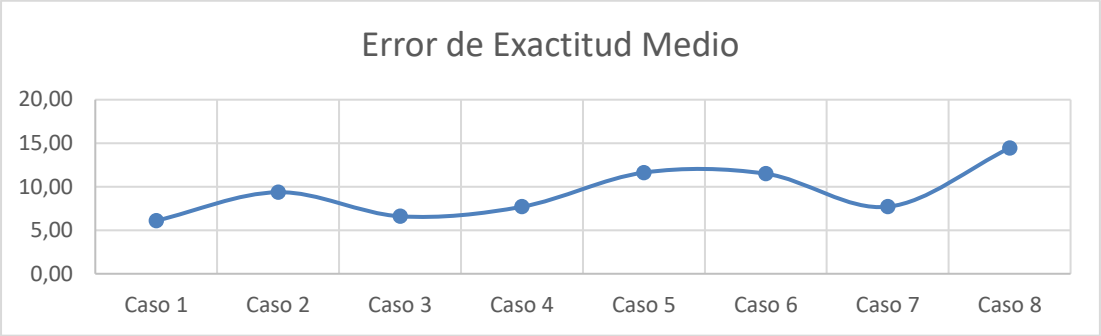


Figura 136. Error de exactitud para pose aleatoria

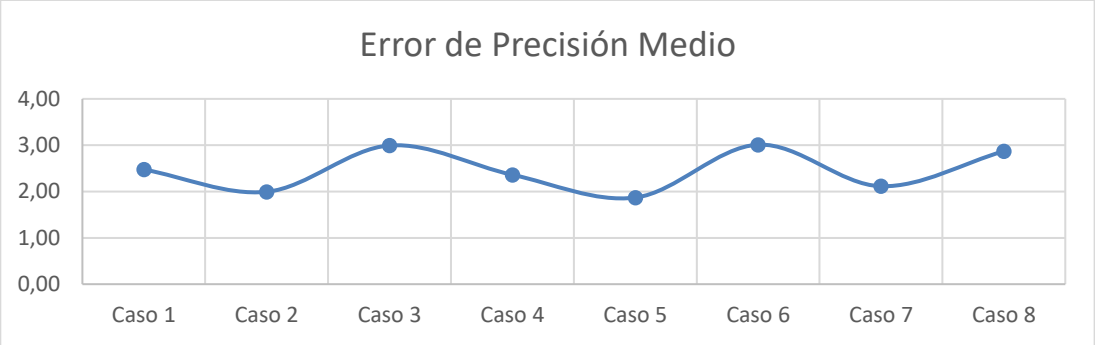


Figura 137. Error de precisión para pose aleatoria

Finalmente, otras variables como número de iteraciones realizadas y el número de segmentos locales, junto con los errores de pose y el tiempo se toman en cuenta para una ponderación de criterios. Esta ponderación permitirá escoger el caso con menor valor, lo cual lo atribuye como mejor caso. La siguiente ecuación define la función de ponderación y el porcentaje respectivo:

$$F_p = P_t + P_i + P_s + P_{ec} + P_{ee} + P_{ep} \quad (96)$$

Donde:

$P_{ec} = 25\%$: Ponderación error de cálculo

$P_{ee} = 20\%$: Ponderación error de exactitud

$P_s = 20\%$: Ponderación de segmentos locales

$P_{ep} = 15\%$: Ponderación error de precisión

$P_i = 10\%$: Ponderación de número de iteraciones

$P_t = 10\%$: Ponderación de tiempo

Las ponderaciones se colocan con mayor porcentaje al error de cálculo es decir el error entre el punto calculado y la meta que se requiere alcanzar. El número de iteraciones y el tiempo son considerados con menor porcentaje ya que su relevancia es menor. De igual manera la ponderación queda definida para los siguientes parámetros. Como se muestra en la Figura 138 el Caso 4 es el mejor caso, su bajo número de segmentos locales, bajo tiempo y bajo número de iteraciones lo pondera como el mejor caso. Para un siguiente parámetro se debe mejorar el error de cálculo, de esta manera optimizar más el desarrollo del algoritmo.

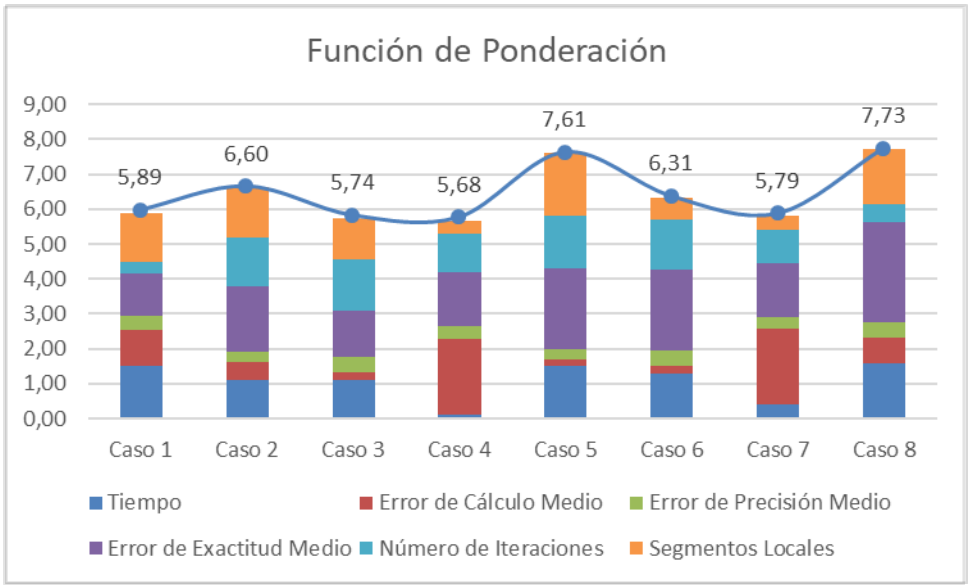


Figura 138. Resultados de la ponderación para la pose aleatoria

4.7.3. Optimización de Profundidad

La profundidad es un parámetro definido en el algoritmo como la cantidad de saltos o segmentos locales máximos permitidos a lo largo de una ramificación del árbol. De igual forma para desarrollar las pruebas se definieron 8 casos los cuales se detallan a continuación:

Tabla 21
Casos para optimización de profundidad

	Caso 1	Caso 2	Caso 3	Caso 4	Caso 5	Caso 6	Caso 7	Caso 8
Profundidad	2	4	6	8	10	12	14	16

Para el Caso 1 no se llegó a la meta por lo tanto se lo descarta directamente. Se examinan las mismas variables críticos que definen la decisión final sobre cuál sea el caso para la profundidad. Un aspecto destacable es el número de iteraciones, a menor iteraciones el algoritmo tiene menor gasto computacional. Es así que se llegó a obtener las respuestas de la (Figura 139), los dos mejores casos resultan ser Caso 6 y Caso 7 observando una disminución hasta un punto que vuelve a subir.

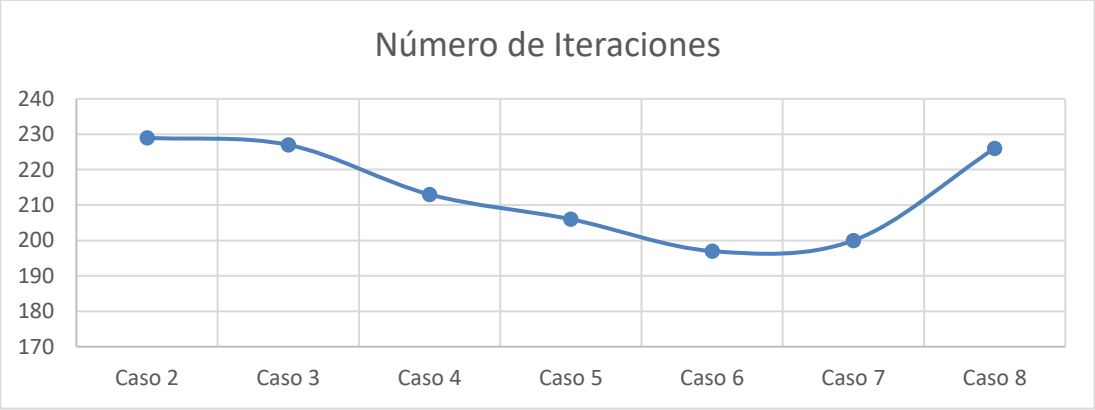


Figura 139. Número de iteraciones para profundidad

Se evaluó la variable de la pose final del UGV al igual que en anterior parámetro, dando lugar al Error de Exactitud en la (Figura 140), y el Error de Precisión en la (Figura 141) y Error de Cálculo en la (Figura 142).

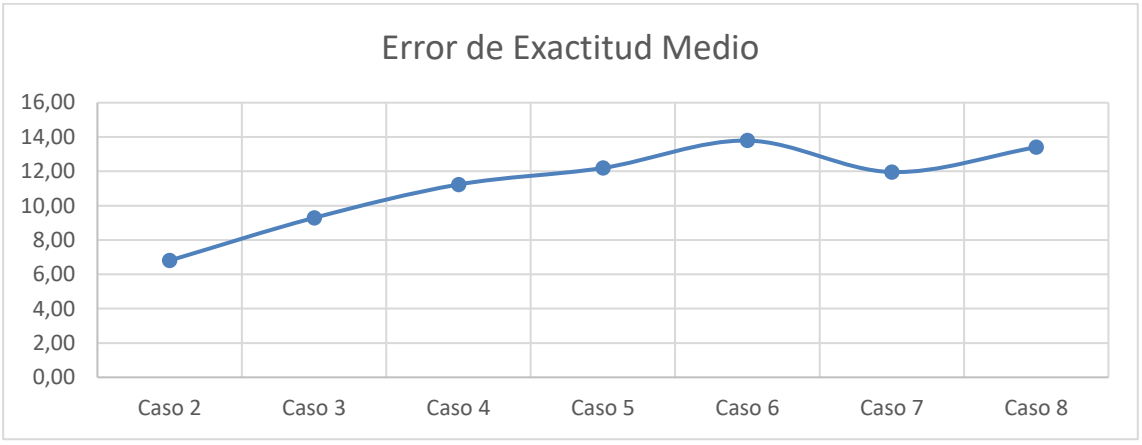


Figura 140. Error de exactitud para profundidad

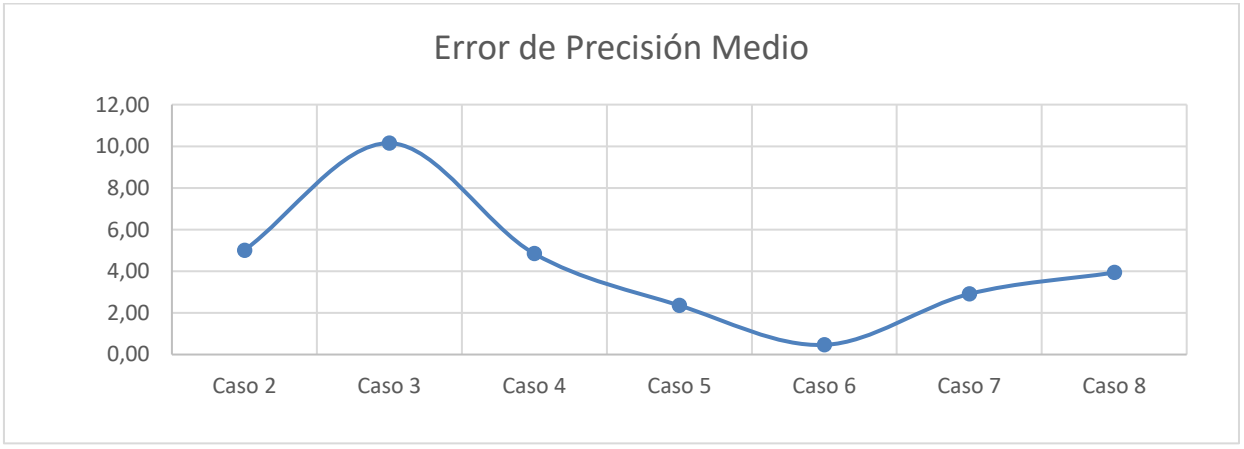


Figura 141. Error de precisión para profundidad

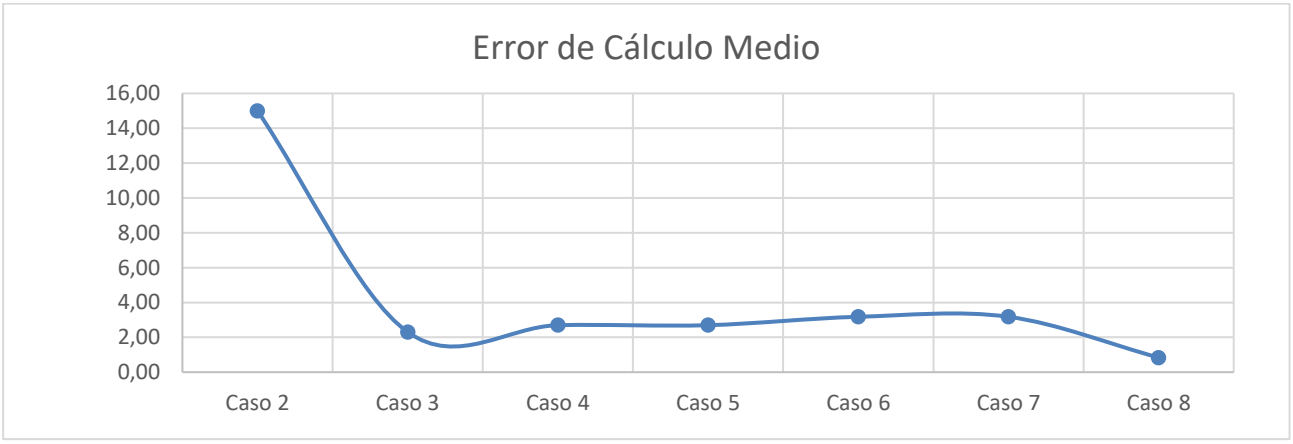


Figura 142. Error de cálculo para profundidad

Finalmente, todas las variables como tiempo, número de segmentos locales, errores de pose y número de iteraciones se toman en cuenta para la ponderación de criterios. Esta ponderación se realiza con los mismos pesos previamente definidos, para lo cual el caso con menor valor es escogido como mejor caso para este parámetro. En la (Figura 143) el Caso 6 es el mejor caso.

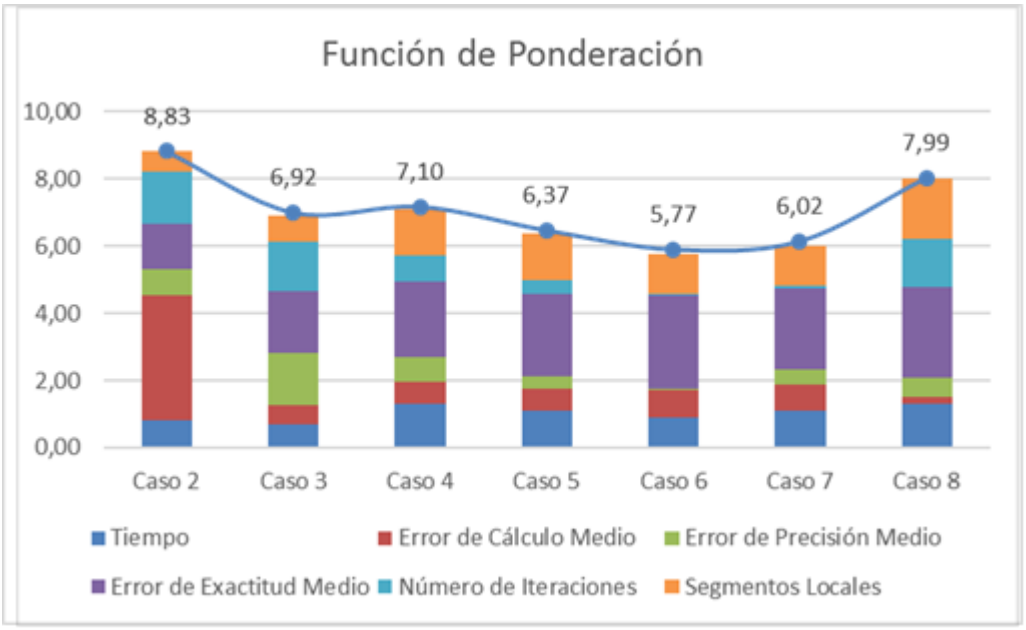


Figura 143. Resultados de la ponderación para profundidad

Su bajo número de iteraciones, bajo tiempo y bajo error de precisión lo pondera como el mejor caso. En este parámetro se consigue bajar el error de cálculo, optimizando el algoritmo. Se observa que a mayor profundidad este error de cálculo disminuye ya que posee mejores opciones a su disposición.

4.7.4. Optimización de Umbral de Riesgo

El umbral de riesgo es un parámetro definido en el algoritmo como el porcentaje del vehículo, considerado como rectangular, que debe de estar libre de colisión. Lo ideal fuese que el 100% del vehículo permanezca libre de colisión, pero al conocer que el mapa de dos dimensiones obtenido

posee un error del 10% es necesario optimizar este parámetro. De igual forma para desarrollar las pruebas se definieron 6 casos los cuales se detallan a continuación:

Tabla 22
Casos para optimización de umbral de riesgo

	Caso 1	**Caso 2**	**Caso 3**	**Caso 4**	**Caso 5**	**Caso 6**
Umbral de Riesgo	0.8	0.9	0.92	0.94	0.96	0.98

Se examinan las mismas variables críticos que definen la decisión final sobre cuál sea el caso para el umbral de riesgo. Un aspecto destacable es el error de cálculo, a partir del caso 2 el error de cálculo no supera el 2% lo cual indica una optimización alcanzada en el cálculo. El error de Exactitud se muestra en la (Figura 144), el Error de Precisión en la (Figura 145) y Error de Cálculo en la (Figura 146).

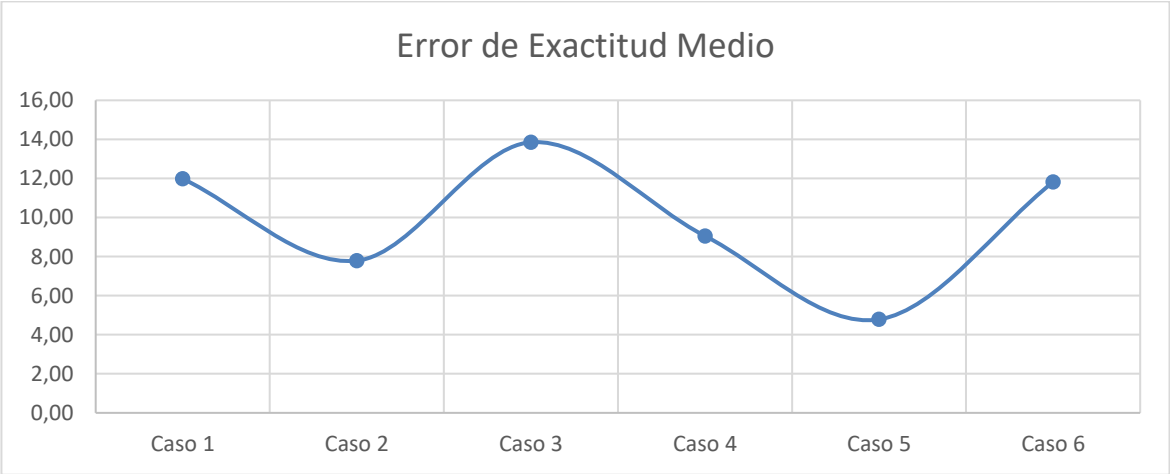


Figura 144. Error de exactitud para umbral e riesgo

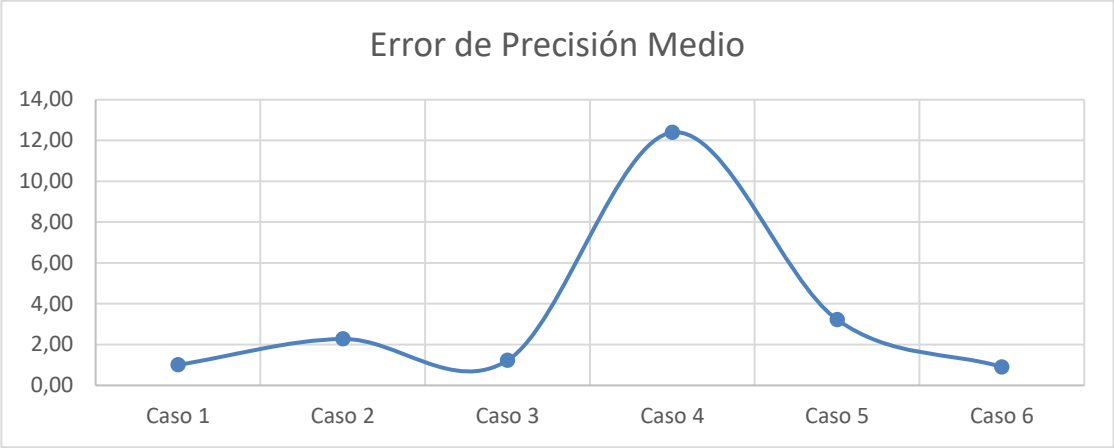


Figura 145. Error de precisión para umbral de riesgo

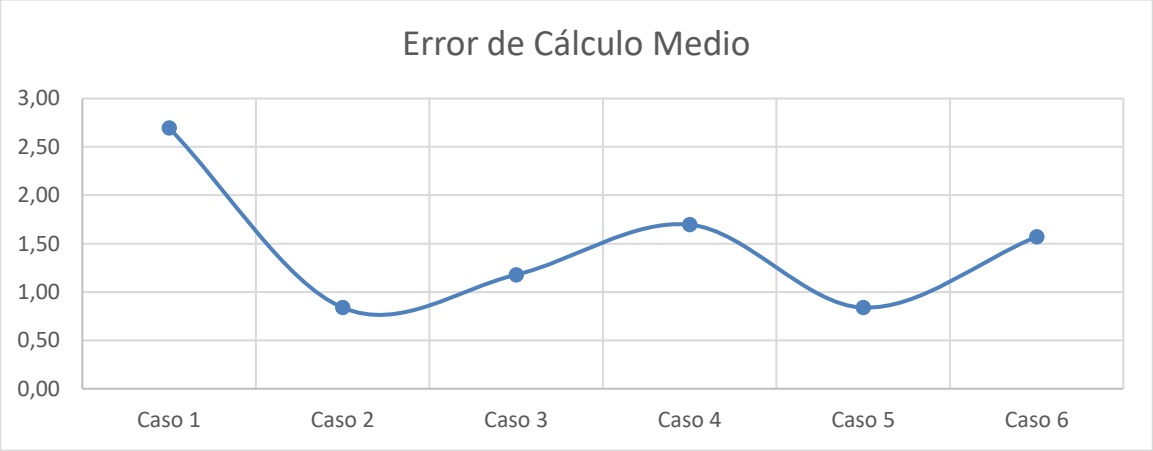


Figura 146. Error de cálculo para umbral de riesgo

Todas las variables como tiempo, número de segmentos locales, errores de pose y número de iteraciones se toman en cuenta para la ponderación de criterios. Esta ponderación se realiza con los mismos pesos previamente definidos, para lo cual el caso con menor valor es escogido como mejor caso para este parámetro. Esto es mostrado en la (Figura 147), para lo cual el Caso 6 es el mejor caso.

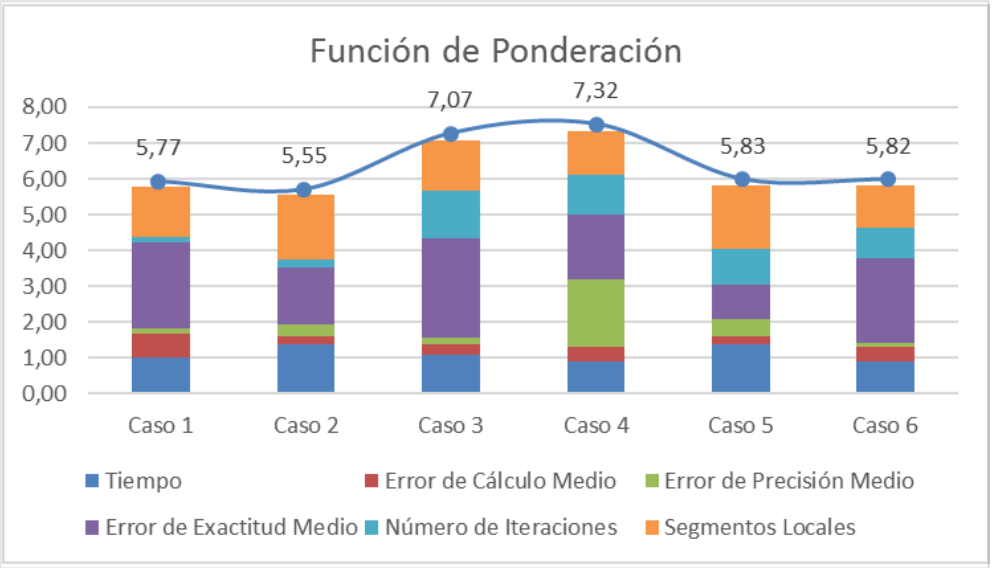


Figura 147. Resultados de ponderación para umbral de riesgo

Su bajo número de iteraciones, bajo error de cálculo y bajo error de precisión lo pondera como el mejor caso, alcanzando un valor de 5.55 en ponderación. En este parámetro se consigue bajar la ponderación total, logrando una optimización general.

4.7.5. Optimización de Meta Real

La meta real es un parámetro definido en el algoritmo como el porcentaje de probabilidad que posee la meta final en ser escogida como la meta parcial o real. Considerando pruebas previas no se puede colocar esta probabilidad con valores mayores al 50% ya que en esos casos la meta no es alcanzada de ninguna manera. Para desarrollar las pruebas se definieron 5 casos los cuales se detallan a continuación:

Tabla 23
Casos para optimización de meta real

	Caso 1	**Caso 2**	**Caso 3**	**Caso 4**	**Caso 5**
Meta Real	50%	40%	30%	20%	10%

Nuevamente se examinan las mismas variables críticos destacando los segmentos locales, es la primera ocasión que se realiza trayectoria con solo dos segmentos locales, en el peor caso se realizan 7 segmentos. Es así que se llegó a obtener la respuesta de la (Figura 148), el mejor caso resulta ser el Caso 2.

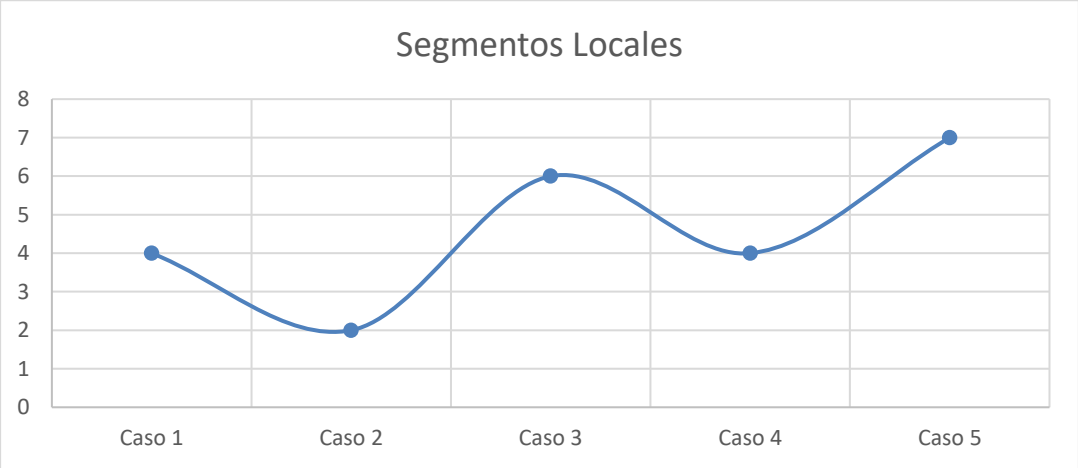


Figura 148. Segmentos del planificador local para meta real

Se evaluó la variable de la pose final del UGV al igual que en anterior parámetro, dando lugar al Error de Exactitud en la (Figura 149), y el Error de Precisión en la (Figura 150) y Error de Cálculo en la (Figura 151).

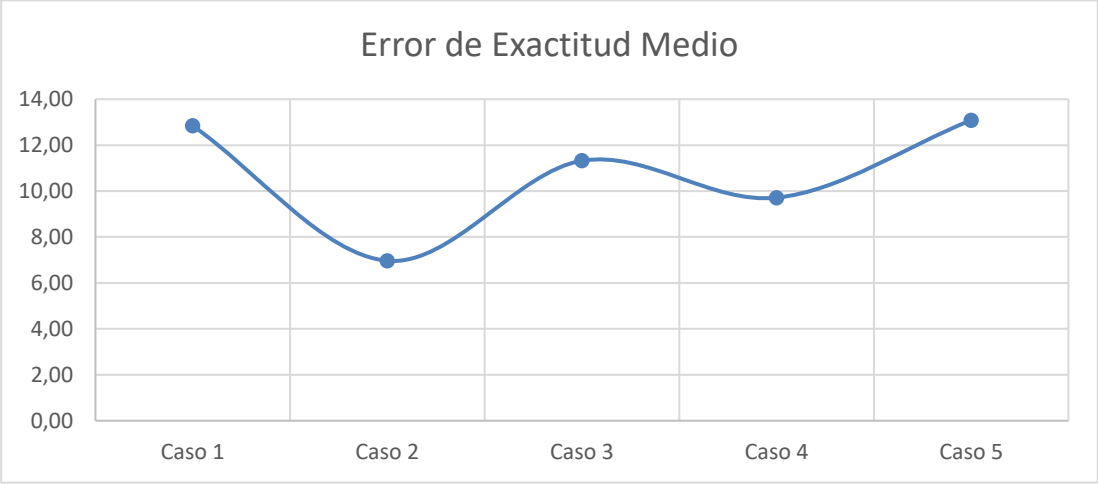


Figura 149. Error de exactitud para meta real

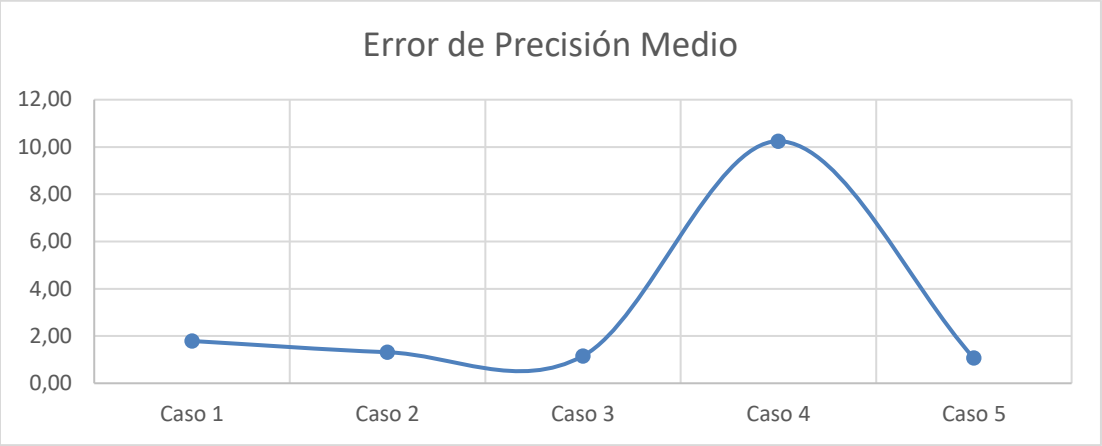


Figura 150. Error de precisión para meta real

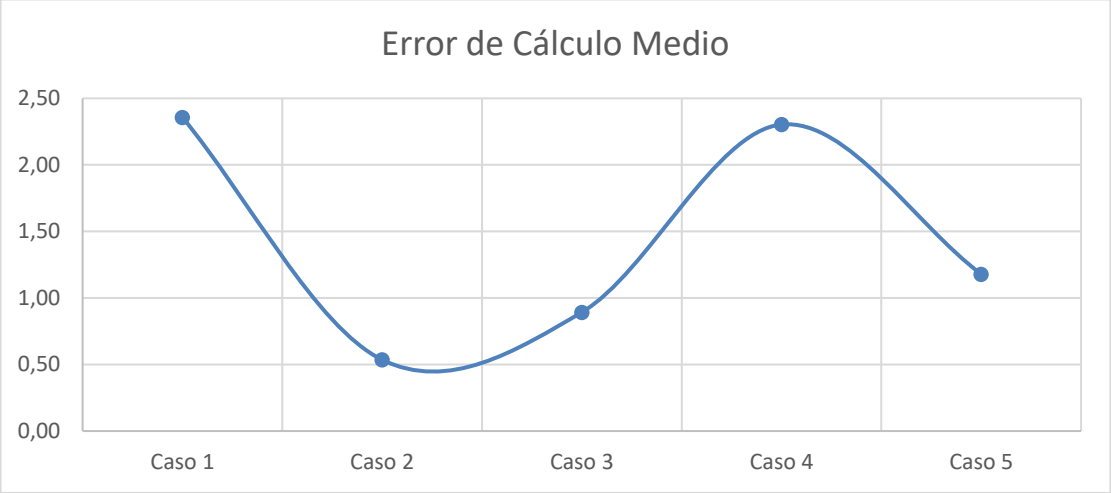


Figura 151. Error de cálculo para meta real

Finalmente, todas las variables como tiempo, número de segmentos locales, errores de pose y número de iteraciones se toman en cuenta para la ponderación de criterios previamente utilizada en los demás criterios. Es así como se llega a lo mostrado en la (Figura 151), en la cual se puede denotar como mejor caso al Caso 2.

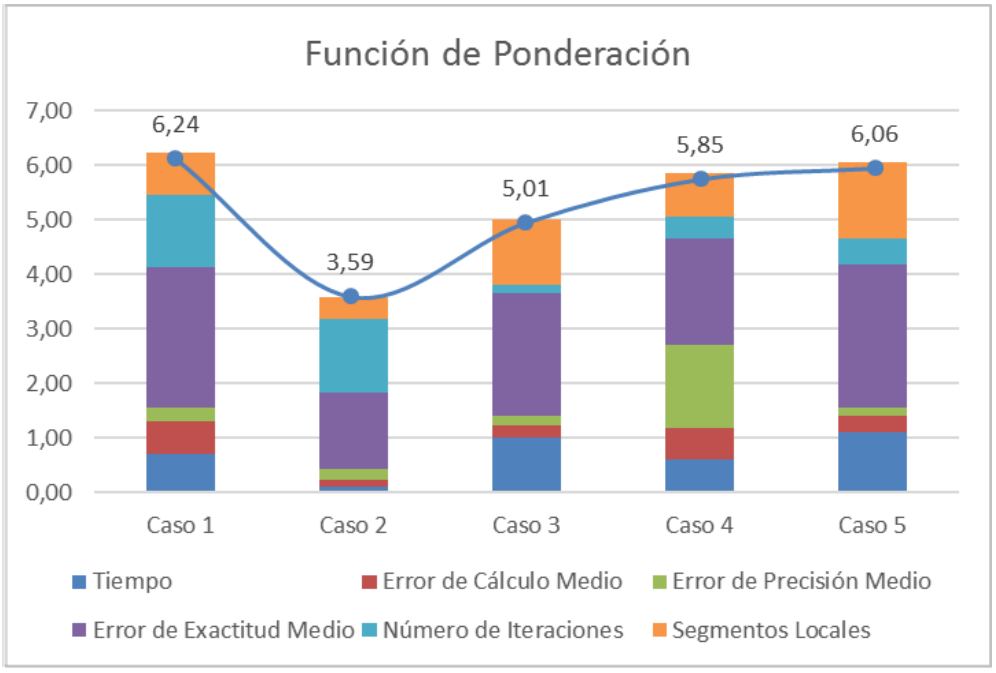


Figura 152. Resultados de ponderación para meta real

Su bajo tiempo, bajo error de cálculo, baja cantidad de segmentos locales y bajo error de precisión lo pondera como el mejor caso. En este parámetro se consigue bajar el error de cálculo, optimizando el algoritmo. Se observa que es necesario una probabilidad del 40% de que la meta final sea escogida como meta aleatoria.

4.7.6. Optimización de Paso Final

Como ya se había mencionado anteriormente el vehículo al ser no holonómico, se dificulta que se alcance la meta final fielmente. Es por ello que el paso final es un parámetro definido como la distancia euclidiana, en centímetros, de la cual el UGV puede estar de la meta final y así asumirla como meta alcanzada. De igual forma para desarrollar las pruebas se han definido 6 casos los cuales se detallan a continuación:

Tabla 24
Casos para optimización de paso final

	Caso 1	Caso 2	Caso 3	Caso 4	Caso 5	Caso 6
Paso Final	120	100	80	60	40	20

Se examinan las mismas variables críticos que definen la decisión final sobre cuál sea el caso para el umbral de riesgo. Un aspecto destacable es el tiempo, a menor paso final menor resulta ser el tiempo de ejecución. La respuesta del tiempo se muestra en la (Figura 153), junto con aquella está el error de Exactitud en la (Figura 154), el Error de Precisión en la (Figura 155) y Error de Cálculo en la (Figura 156).

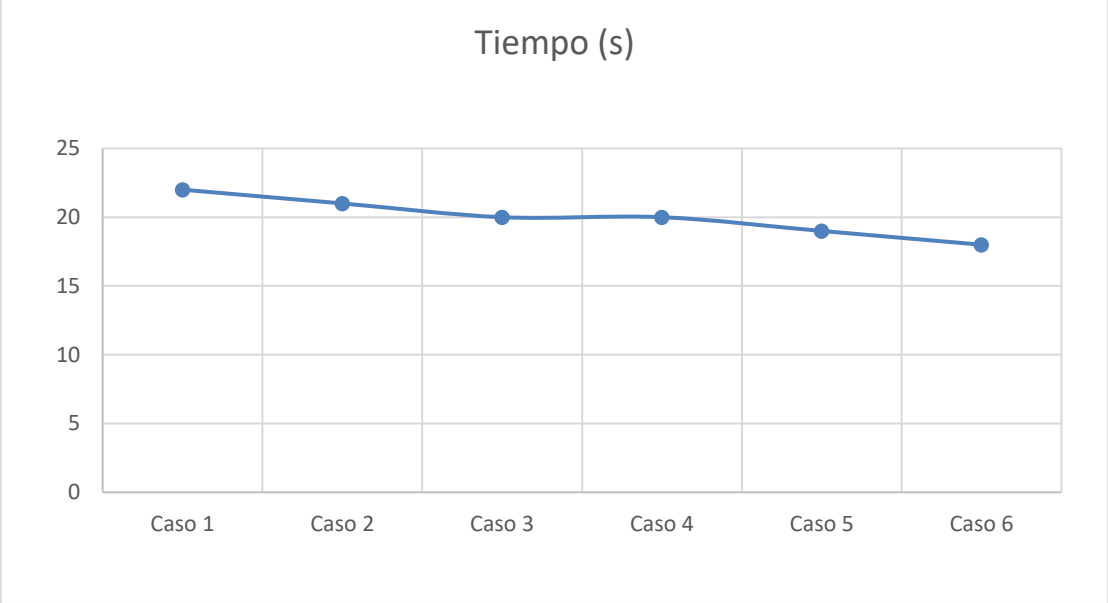


Figura 153. Tiempo para paso final

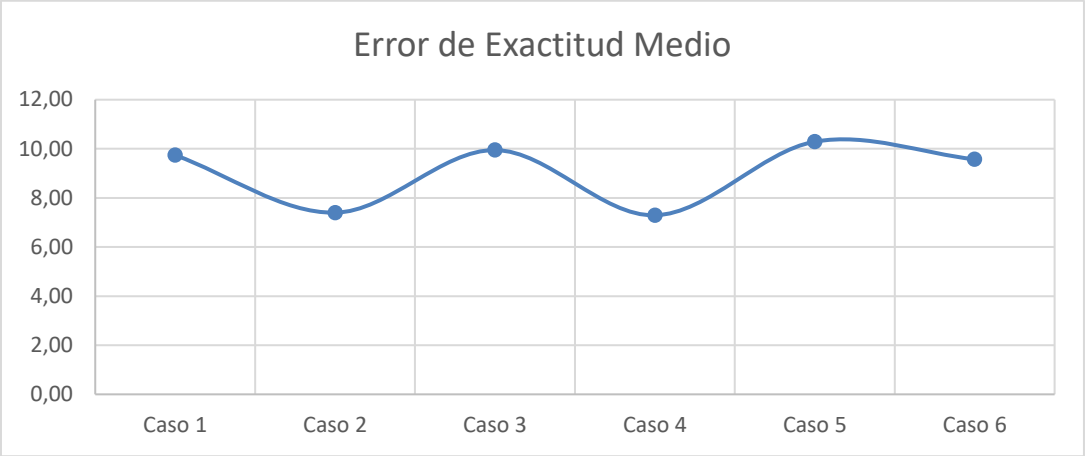


Figura 154. Error de exactitud para paso final

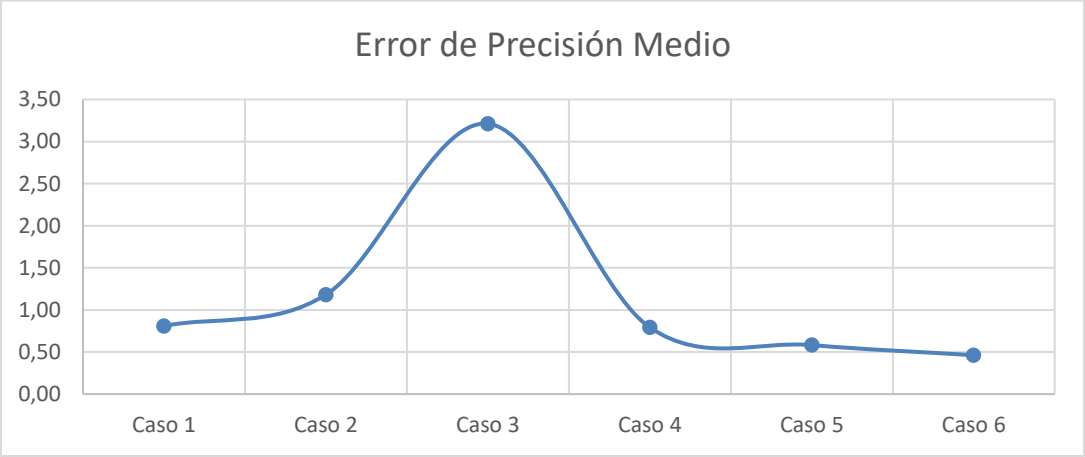


Figura 155. Error de precisión para paso final

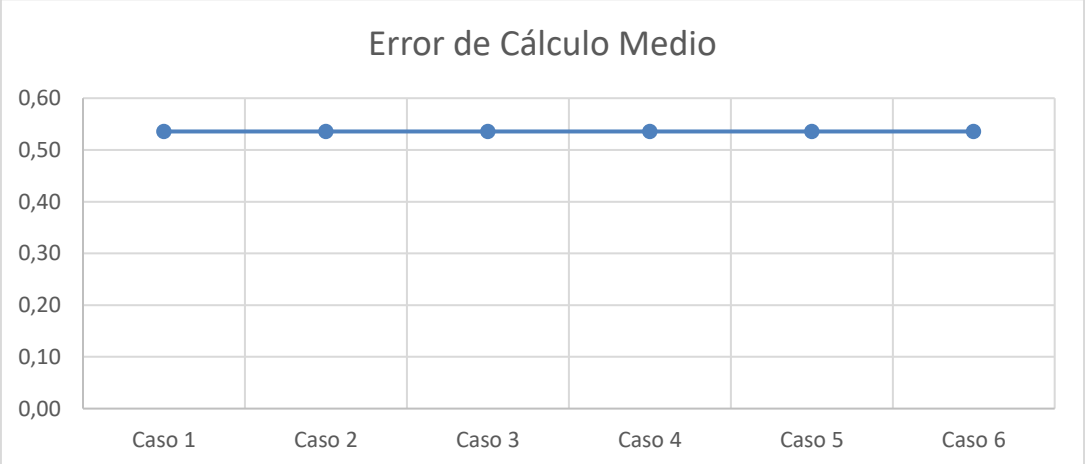


Figura 156. Error de cálculo para paso final

Complementariamente, todas las variables son consideradas en la ponderación de criterios, dando lugar a la (Figura 157). En esta figura podemos apreciar varias cosas; la primera es que el Caso 4 es el mejor caso y esto se debe a que alcanza la ponderación más baja que no se ha alcanzado. Este caso es el mejor debido a su baja cantidad de segmentos locales, su bajo número de iteraciones, su bajo tiempo, su baja cantidad de segmentos locales y bajos errores.

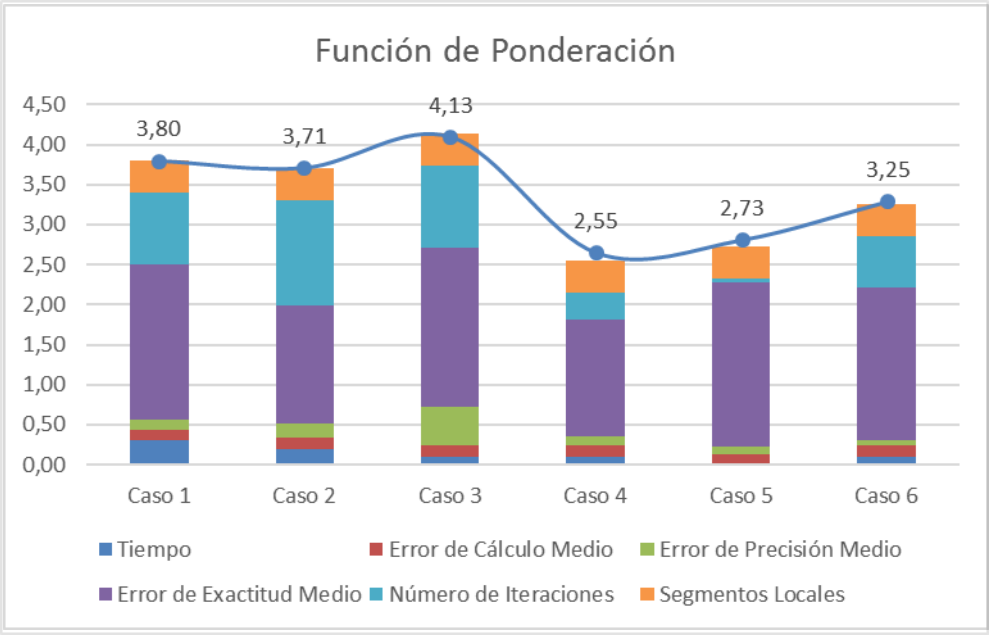


Figura 157. Resultados de ponderación para paso final

De esta manera se ha llegado a una optimización global de los parámetros definiéndolos para futuros casos y para la sección de experimentación. No quiere decir que estos parámetros queden fijos, la palabra es optimizada y que por supuesto pueden ser modificados conforme las circunstancias lo ameriten. Es así que se llega a la Tabla de optimización mostrada a continuación:

Tabla 25
Optimización de parámetros en el algoritmo de planificación

MAXSCANS	Alfa	Beta	Profundidad	Umbral Riesgo	Meta Real	Paso Final
15 escaneos	0.4	2	12 pasos	0.9	40%	60 cm

Cada caso definió un parámetro y en si una evolución del planificador. Como parte de esta es necesario mostrar la evolución de la trayectoria como resultado de la optimización de cada parámetro. La Figura a continuación describe lo ocurrido:

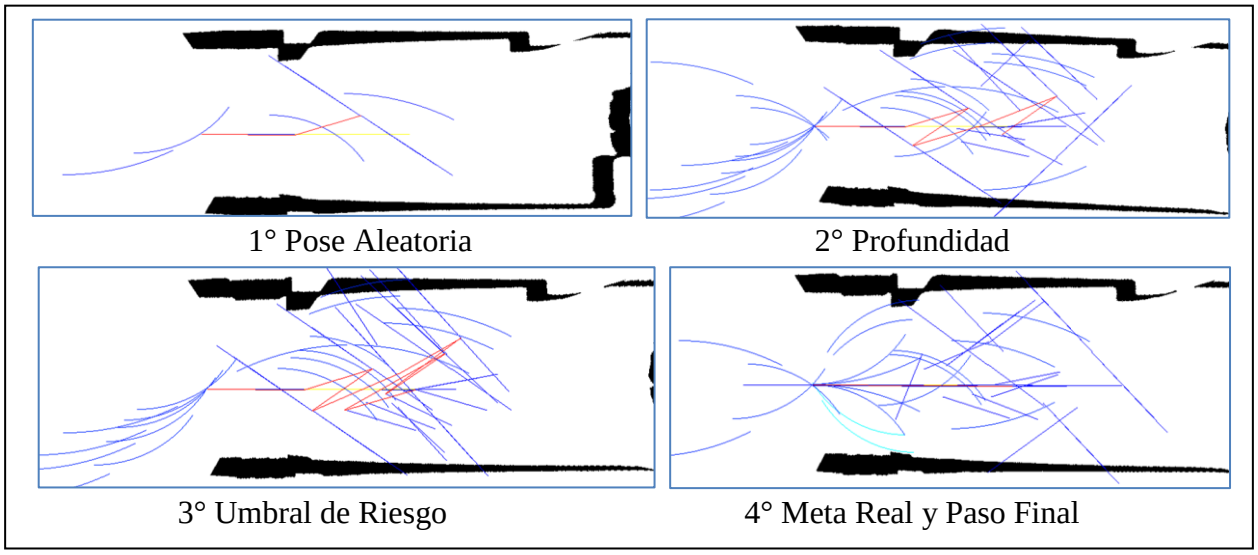


Figura 158. Evolución de la trayectoria realizara ante la optimización de parámetros

CAPITULO V

5. PERCEPCIÓN 3D

En este capítulo se explica el enfoque tomado para realizar la percepción 3D en el proceso de navegación del vehículo, explicando las fases de desarrollo en la cual se explica los algoritmos y metodologías utilizadas para alcanzar el objetivo de la percepción.

5.1. Adquisición de entrono 3D

Como se ha explicado en los capítulos anteriores, al trabajar con una cámara estereoscopia y su capacidad de obtener los datos de profundidad ya se tiene de percepción 3D, pues estamos estimando la profundidad de un plano visual, pero el objetivo es mostrar la percepción y los campos de visión adquiridos durante la navegación, ya se conoce el entorno según el mapeo en 2D, resulta muy conveniente también conocer su entorno en 3D, es muy común ver trabajos en los que se realizan percepciones en dos dimensiones, así como también usar sensores laser para recibir en 3D (Weingarten, 2003) , sin embargo con ayuda de la estéreo visión se puede realizar una imagen en 3D de un momento exacto, pero solo se tendría ciertas partes detectadas y mostradas en 3D, de acuerdo a los puntos capturados en un plano de visión.

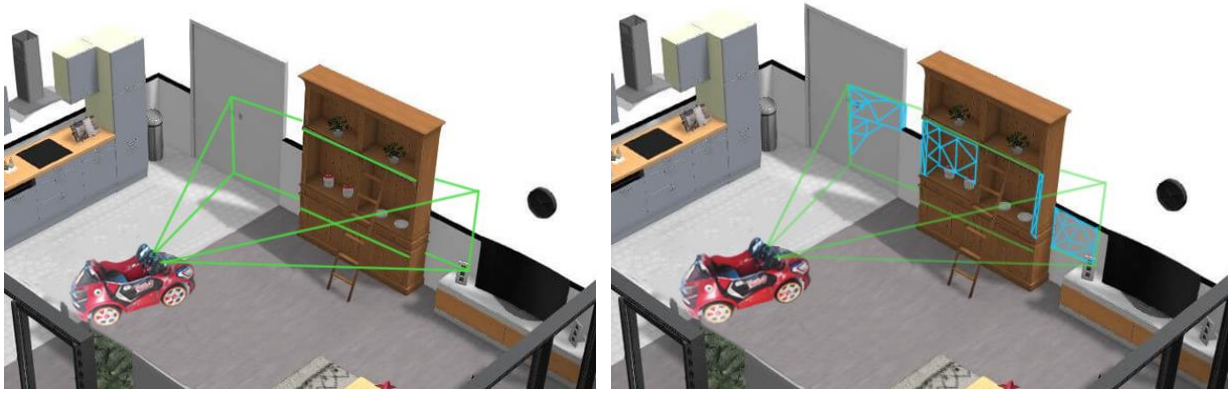


Figura 159. Representación de mapeo espacial desde una posición estática en un plano.

Para obtener la información más completa del entorno se tendría que ir relacionando un conjunto de imágenes en 3D obtenidas, a esto se lo conoce como mapeo espacial (Wulf, 2004). Para poder realizar este mapeo entonces es necesario conocer nuestra ubicación en todo momento y de esta manera correlacionar las imágenes receptadas cuando se realiza este proceso se lo conoce como “Simultaneous and Localization Mapping” (SLAM) (Nuchter, 2003).

El conocimiento del entorno trabaja de manera mutua, pues pese a que el carro parte de un punto estático, en el cual tiene que realizar la planificación de movimientos usando el conocimiento del LIDAR, también en ese mismo momento ya tiene información obtenida de la cámara como se ilustra en la Figura 160. El LIDAR puede tomar los datos que se encuentran en un plano como lo refleja con línea verde y cualquier objeto que se muestre en este plano es detectado por el sensor, mientras que la cámara otorga una visión que tiene cierto ángulo de apertura que crea un plano perpendicular mostrado en color azul.

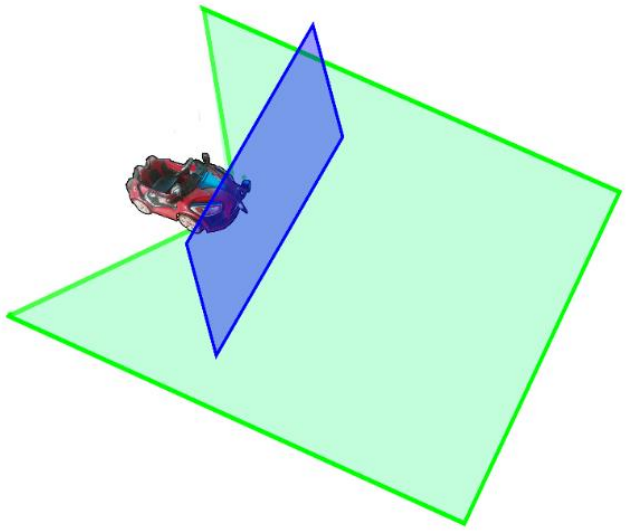


Figura 160. Representación planar de percepciones combinadas entre LIDAR 2D y visión

Si el vehículo se ejecutase en estas condiciones, podría funcionar perfectamente, y gracias al movimiento que realiza se tuviera los resultados del entorno frente a la trayectoria a seguir,

usando diferentes métodos de visión por computador, sin embargo, el trabajo que presentamos ocupa las grandes ventajas de usar una percepción de distancia que se explicó en el Capítulo II. Al poder estimar la profundidad es como si se representara múltiples planos en paralelo al plano de visión creado por la visión monocular, la distancia que sea capturada se almacenará en un punto de alejamiento diferente entre puntos, y de esta manera se puede alcanzar una percepción y representación en 3D basándonos en el funcionamiento de la disparidad estéreo, esta explicación de funcionamiento esta ilustrada en la (Figura 161), en donde se muestra a manera de representación una aproximación teórica de la captura del entorno 3D combinado en un instante.

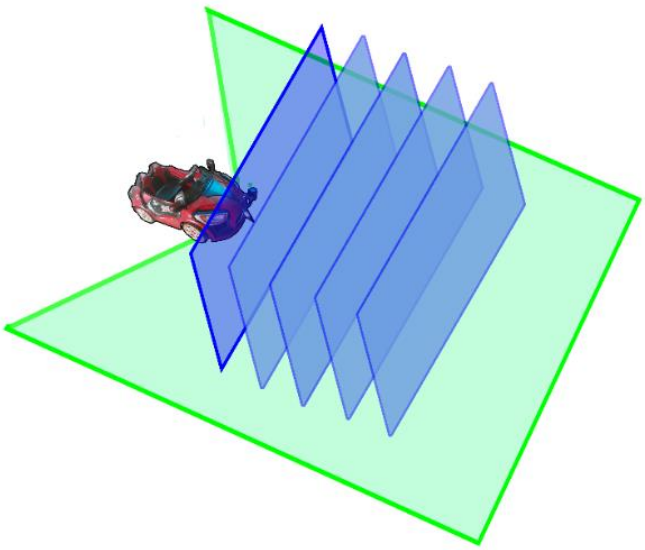


Figura 161. Representación planar de percepciones combinadas entre lidar 2D y estéreo visión.

La estimación de la disparidad estéreo ha sido un tema bien estudiado en la visión por computadora, con aplicaciones en varios ámbitos, incluida la reconstrucción tridimensional a gran escala, estimación de escenas y evitación de obstáculos para conducción autónoma y vuelo, etc. (Pillai, 2016). La mayoría de los estudios se han centrado en mejorar ciertos aspectos donde se toma muy en cuenta los resultados y los costos computacionales (Zbontar, 2015) (Szeliski,

2002) (A. Geiger, 2012), la necesidad de hacer aplicaciones estéreo en tiempo real ha llevado al uso de métodos basados en GPus y FPGAs u otros métodos en paralelo (S. K. Gehrig, 2009) (Tedrake, 2015) (Breiteneder, 2013), con esto se puede llegar a un rendimiento de alta velocidad que puede hacer que se cumpla las condiciones de funcionamiento de este trabajo, lograr fusionar las mediciones tomadas de manera automática y emitir resultados en los que se evidencie la percepción del entorno y permitirle al vehículo autónomo tomar decisiones rápidas (Pillai, 2016). Como se explicó en capítulos anteriores la cámara ZED, requiere necesariamente de GPU y es debido a estos factores. La cámara y el LIDAR, han sido colocados de tal manera que el LIDAR pueda obtener información del entorno a una altura media en la que el vehículo se encuentra y la cámara en una posición en la que se pueda obtener un plano amplio de manera vertical, la obtención de datos de ambos dispositivos se representa por medio de la Figura 162. Donde la línea de color verde es el resultado que entrega el sensor LIDAR y la de color azul representa al plano de captura de imagen que se tiene por medio de la cámara.

Figura 162. Representación de funcionamiento combinado montado sobre el vehículo

5.2. Caracterización del mapeo espacial

Luego de obtener los datos del entorno, se presenta como resultado un objeto en 3D, los puntos tomados se colocan en una posición específica en el espacio y la unión de ellos en conjunto con técnicas visuales permite presentar un objeto que se aproxima al entorno tridimensional real. La asignación espacial (también llamada reconstrucción 3D) es la capacidad de crear un mapa 3D del entorno. El mapeo espacial es útil para evitar colisiones, planear el movimiento y combinar de manera realista el mundo real y virtual (Stereolabs, 2018).

Para obtener resultados la cámara escanea continuamente su entorno creando un mapa que se actualiza a medida que el dispositivo se mueve y captura nuevos elementos en la escena. Dado que la cámara percibe distancias más allá del rango de los sensores RGB-D tradicionales, puede reconstruir rápidamente mapas 3D de grandes áreas interiores y exteriores (Stereolabs, 2018).

El software de Stereolabs para la cámara tiene incluidos una serie de programas con los cuales se puede realizar múltiples pruebas de la cámara, tienen también la opción de cambiar los parámetros predefinidos en el software, uno de ellos es el denominado ZEDfu, que permite capturar imágenes en vivo y desarrollar su mapa del entorno, pero además también se puede ingresar videos grabados previamente en formato SVO y abrir archivos OBJ para observar sus características en 3D. Para mostrar como ejemplo se realizó un escaneo usando este software de un entorno interior, probando con los parámetros de resolución VGA y a una distancia media de mapeado. Los resultados se ilustran en la Figura 163, para mostrar los objetos tridimensionales se usa el Software Meshlab.

El Meshlab es de código abierto y sirve para procesar y editar mallas triangulares 3D. Proporciona un conjunto de herramientas para editar, limpiar, curar, inspeccionar, renderizar,

texturizar y convertir mallas. Ofrece funciones para procesar datos brutos y para preparar modelos para impresión 3D (Visual Computing Lab , 2018).

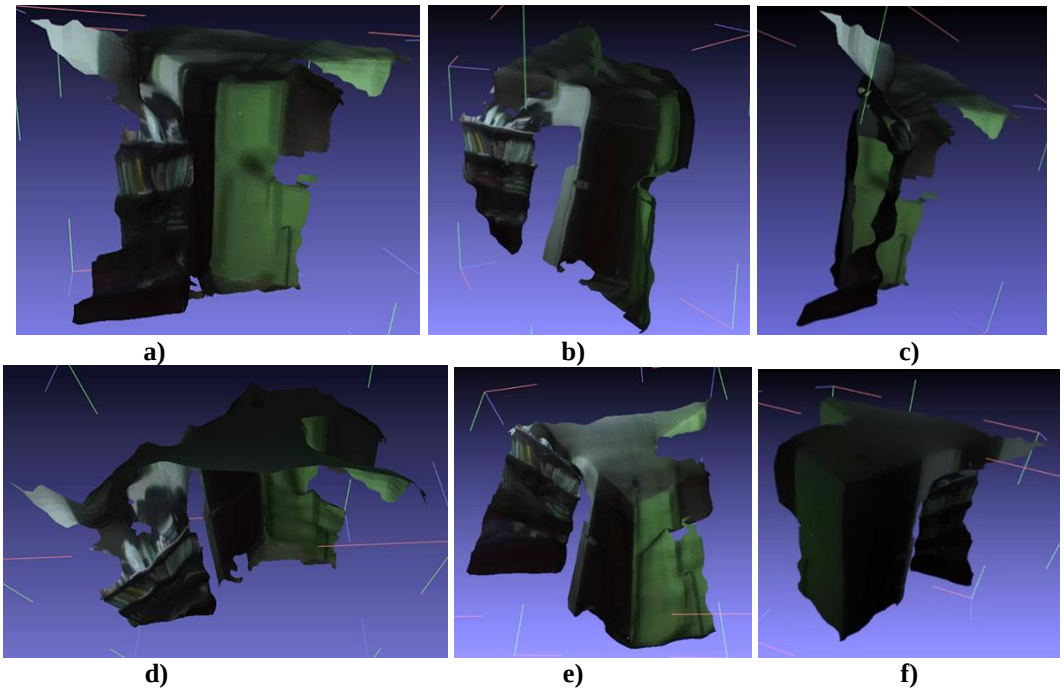


Figura 163. Escaneo de prueba del entorno 3D.
a) Vista frontal, b) vista lateral derecha, c) vista lateral izquierda,
d) vista superior, e) vista inferior, f) vista trasera.

El mapeo espacial representa la geometría del mundo real como una única malla triangular, el resultado esperado de la malla se muestra en la Figura 164. La malla se puede extraer continuamente a medida que se actualiza o solo una vez después de mapear un área completa. Se crea mallas triangulares con vértices, caras y normales unidas a cada vértice. La textura se puede registrar para colorear el modelo final. La malla final también puede ser diezmada usando un filtro de malla para reducir el conteo de polígonos (Stereolabs, 2018).

Figura 164. Malla triangular a partir de una malla triangular
Fuente: (Stereolabs, 2018)

La triangulación se realiza desde una nube de puntos que se realiza de la imagen tomada (Cousins, 2011), usando las características de percepción de distancia, estos puntos tomados se relacionan entre si formando la triangulación y creando la malla de la imagen, también a su vez se va tomando imágenes para obtener la textura, luego se reconstruye el modelo 3D y se aplica las características de iluminación obtenida de la triangulación y el color obtenido de la textura (Weingarten, 2003), finalmente el filtro mejora el resultado final de la imagen este proceso explicado brevemente se muestra también en la Figura 165.

Para poder obtener los datos del mapa es necesario crear un objeto que contenga los valores. Los parámetros para la reconstrucción de mapas 3D que permite cambiar la SDK de la cámara son:

- Resolución de mapeo
- Rango de mapeo
- Filtrado de malla

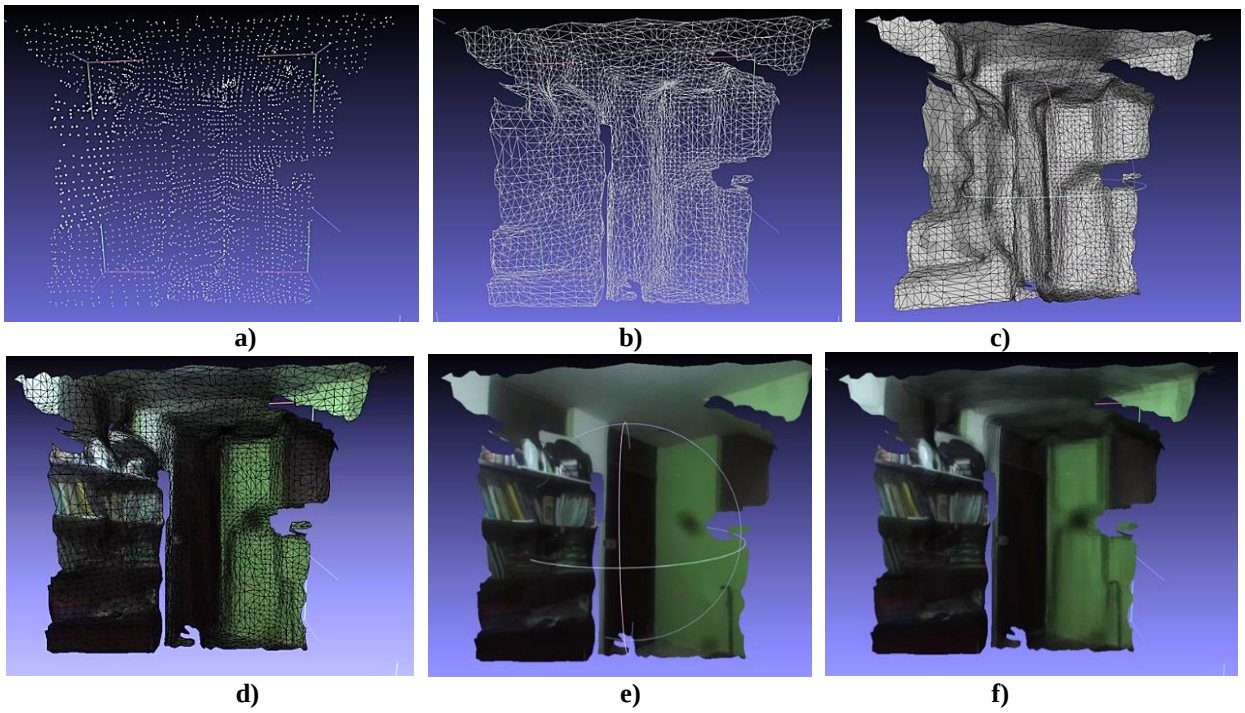


Figura 165. Proceso de reconstrucción del mapa.
a) Nube de puntos b) Creación de malla triangular c) Reconstrucción de objeto 3D
d) aplicación de texturizado e) resultado mapa 3D con texturas f) resultado mapa 3d mostrando efectos de iluminación.

Cuanto mayor sea el número de caras en un modelo, más fino será el detalle que puede capturar. La resolución se puede establecer entre 1 cm y 12 cm. La captura de mapas con alta densidad de triángulos requiere más memoria y recursos (Stereolabs, 2018). En el programa desarrollado, podemos cambiar tanto la resolución de mapeo como también la resolución de la cámara, esta resolución cambia el tamaño de la imagen, así como en el caso del mapeo afecta directamente a la definición de la textura, ya que a mayor resolución será más detallada. Se descartó cambiar la resolución de la malla debido a la baja capacidad de la computadora ya que el tiempo que demora en mapear y en renderizar es excesivo o no realiza la tarea, por lo cual se usa la configuración más baja, en la (Figura 166) se muestra el resultado de la triangulación utilizando una resolución de malla alta y resolución de cámara de HD1080.

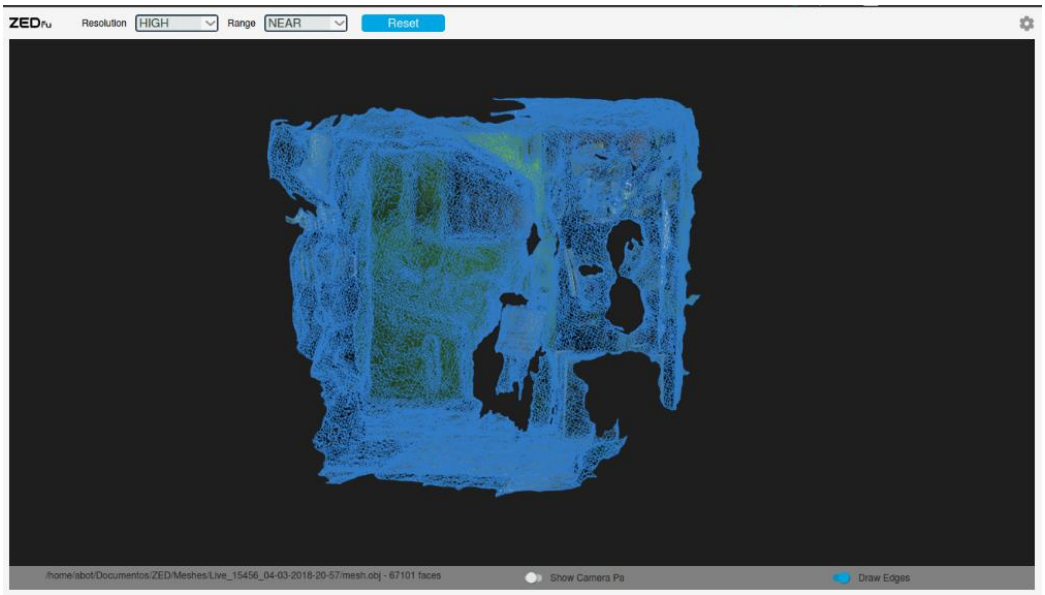


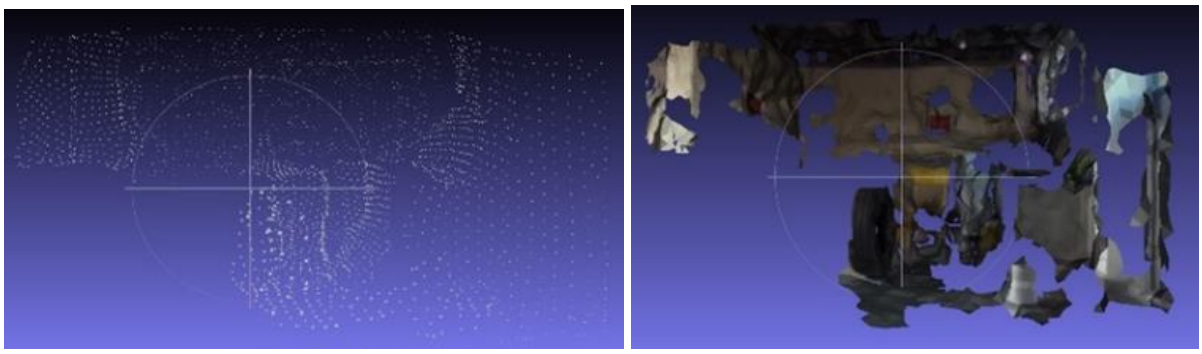
Figura 166. Resultado de mapeo con resolución alta de HD1080, mostrando la malla y por medio del software ZEDfu.

5.2.1. Resolución de imagen y calidad de percepción de la profundidad

Una característica muy importante es la resolución con la que se toma las imágenes, de esta dependerá la calidad de las texturas que son tomadas y por lo tanto el realismo que se le entregue a los objetos, sin embargo, usar las diferentes resoluciones afectará tanto en el tiempo de procesamiento, como en el resultado final, el vehículo se desplazará tanto en entornos interiores como exteriores por lo tanto para esta prueba de mapeo espacial se usaran los ambos entornos.

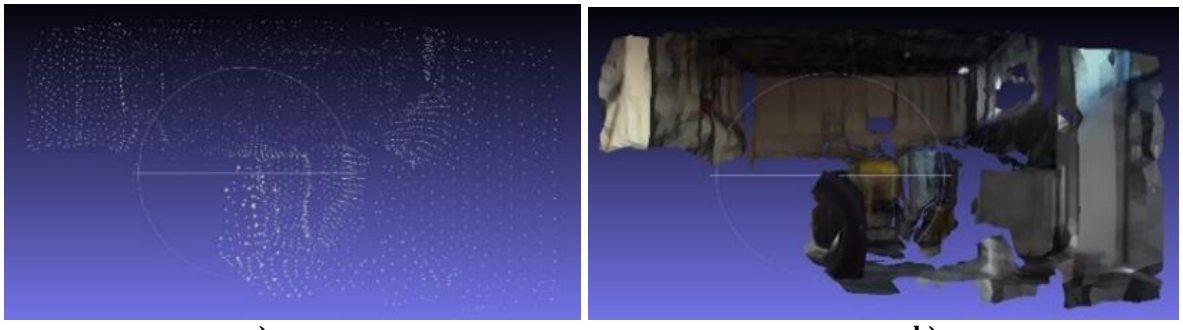
5.2.1.1. Caracterización en entorno interior

Desde la (Figura 167) hasta la (Figura 175) se muestra los resultados de un entorno interior, usando el Software Meshlab, para poder visualizar las imágenes y cambiar sus parámetros de visualización modificando la resolución y la calidad en la percepción.



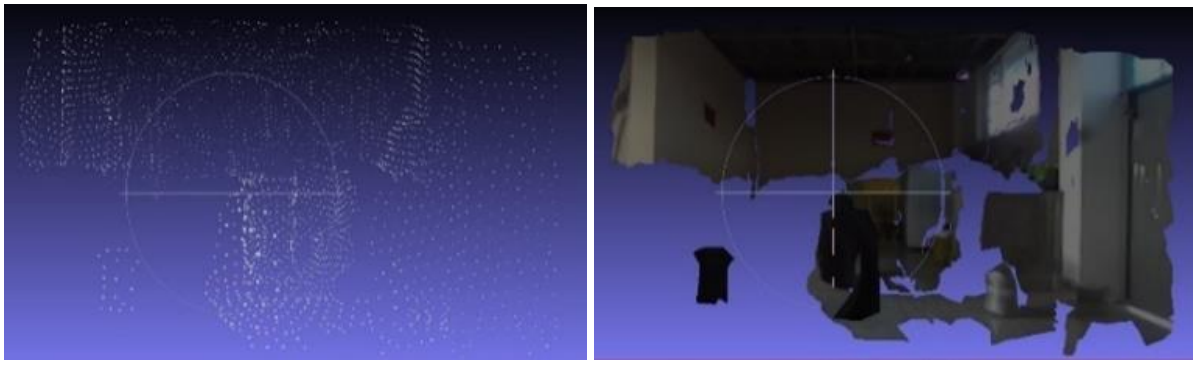
a) **b)**

Figura 167. Proceso de reconstrucción del mapa usando resolución VGA con calidad de percepción de profundidad PERFORMANCE y un filtro de tipo alto. a) Nube de puntos b) Imagen texturizada con efectos de iluminación.



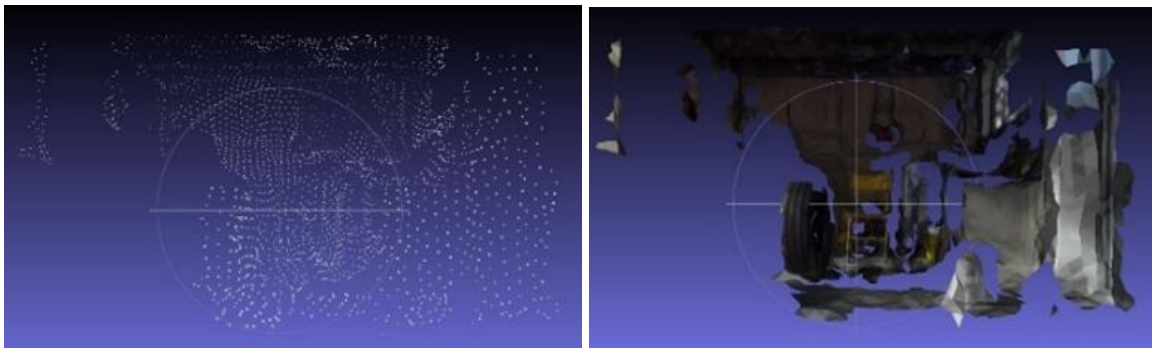
a) **b)**

Figura 168. Proceso de reconstrucción del mapa usando resolución VGA con calidad de percepción de profundidad MEDIUM y un filtro de tipo alto. a) Nube de puntos b) Imagen texturizada con efectos de iluminación.



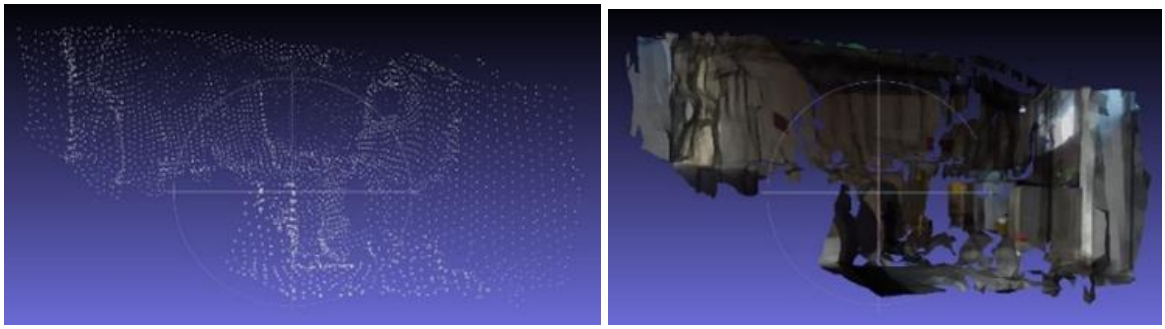
a) **b)**

Figura 169. Proceso de reconstrucción del mapa usando resolución VGA con calidad de percepción de profundidad QUALITY y un filtro de tipo alto. a) Nube de puntos b) Imagen texturizada con efectos de iluminación.



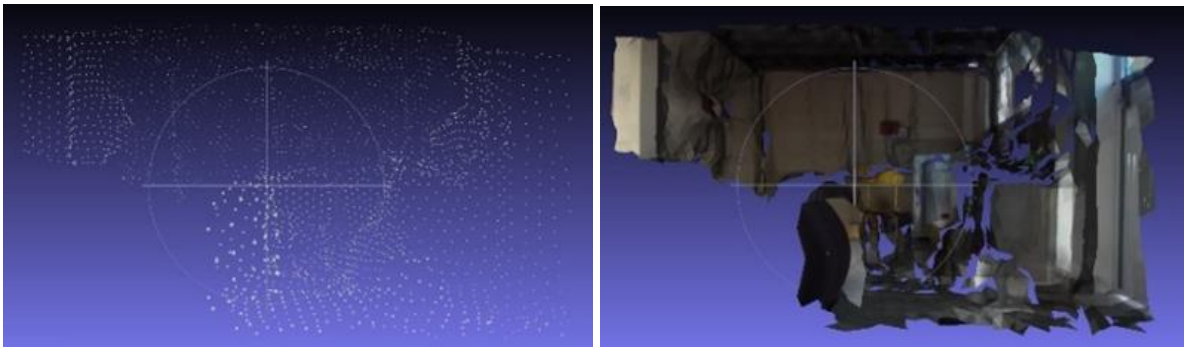
a) **b)**

Figura 170. Proceso de reconstrucción del mapa usando resolución HD720 con calidad de percepción de profundidad PERFORMANCE y un filtro de tipo alto a) Nube de puntos b) Imagen texturizada con efectos de iluminación.



a) **b)**

Figura 171. Proceso de reconstrucción del mapa usando resolución HD720 con calidad de percepción de profundidad MEDIUM y un filtro de tipo alto. a) Nube de puntos b) Imagen texturizada con efectos de iluminación.

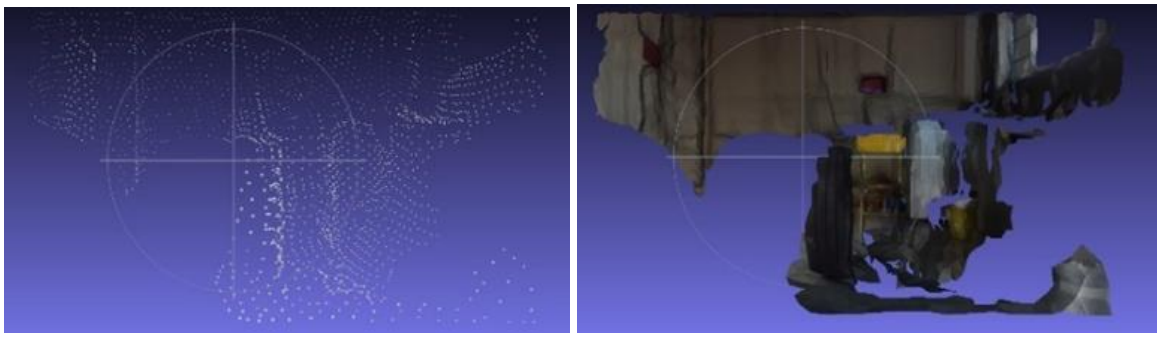


a) **b)**

Figura 172. Proceso de reconstrucción del mapa usando resolución HD720 con calidad de percepción de profundidad QUALITY y un filtro de tipo alto a) Nube de puntos b) Imagen texturizada con efectos de iluminación.

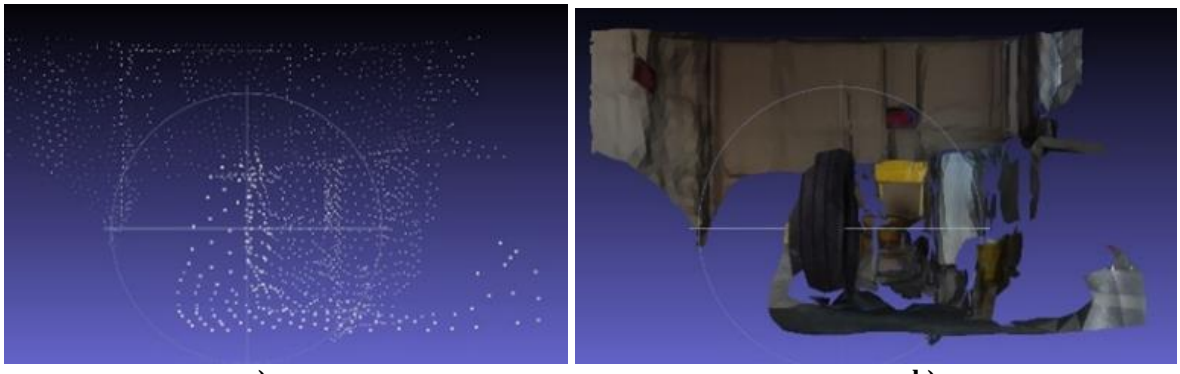
a) **b)**

Figura 173. Proceso de reconstrucción del mapa usando resolución HD1080 con calidad de percepción de profundidad PERFORMANCE y un filtro de tipo alto. a) Nube de puntos b) Imagen texturizada con efectos de iluminación.



a) **b)**

Figura 174. Proceso de reconstrucción del mapa usando resolución HD1080 con calidad de percepción de profundidad MEDIUM y un filtro de tipo alto a) Nube de puntos b) Imagen texturizada con efectos de iluminación.



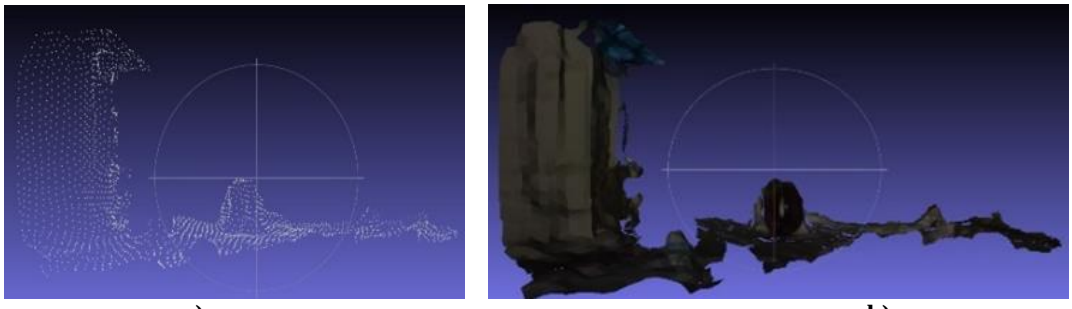
a) **b)**

Figura 175. Proceso de reconstrucción del mapa usando resolución HD1080 con calidad de percepción de profundidad QUALITY y un filtro de tipo alto. a) Nube de puntos b) Imagen texturizada con efectos de iluminación.

Mientras se aumenta la resolucion, mas nitida se muestra la imagen con un resultado cada vez mejor, en cuando a la calidad tambien se obtiene una diferencia notable, mientras mas se aumenta la calidad se obtiene un mejor mapa con aproximaciones cercanas a la verdadera forma del entorno.

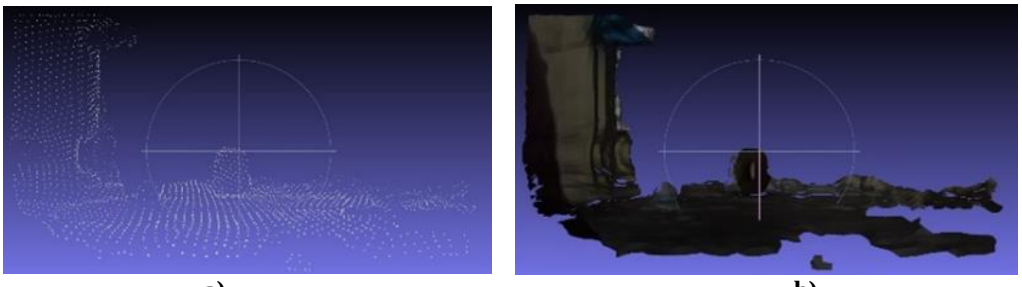
5.2.1.2. Caracterización en entorno exterior

Se realiza tambien pruebas en un ambiente exterior, donde las distancias a los objetos son mayores y se tiene como factor adicional la iluminacion natural, además que se tiende a tener una menor cantidad de objetos en una misma area determinada y tambien se determina por la ausencia de techos y paredes. Estos factores pueden hacer que la cámara funcione de diferente manera.



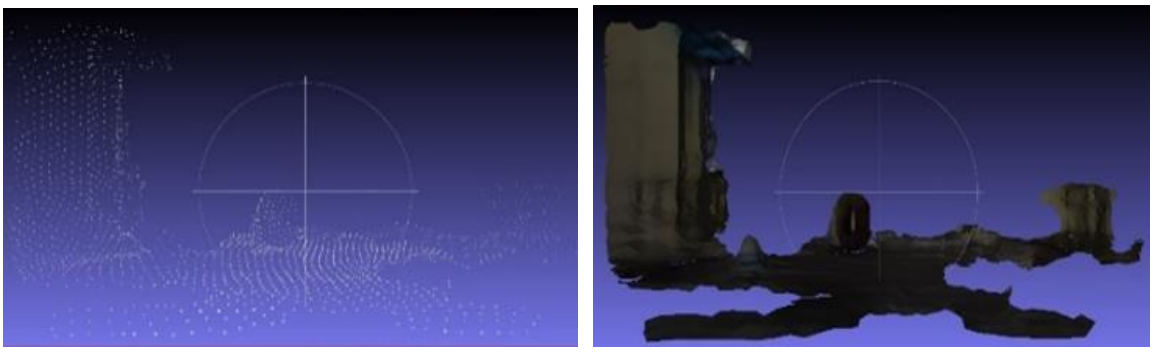
a) **b)**

Figura 176. Proceso de reconstrucción del mapa usando resolución VGA con calidad de percepción de profundidad PERFORMANCE y un filtro de tipo alto. a) Nube de puntos b) Imagen texturizada con efectos de iluminación.



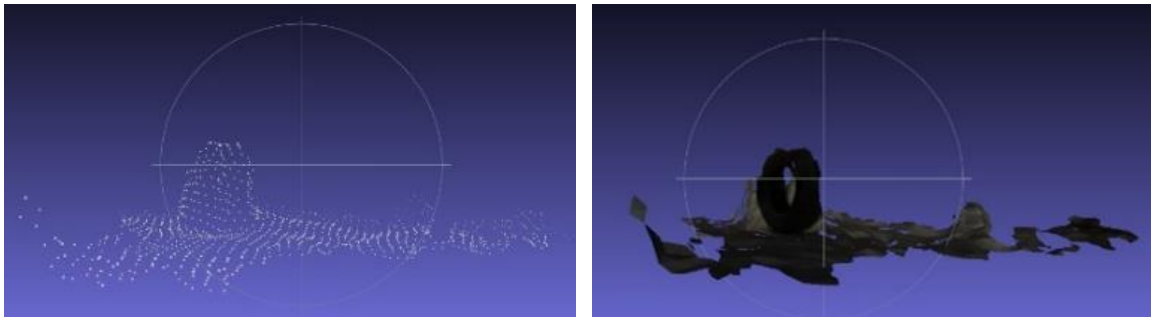
a) **b)**

Figura 177. Proceso de reconstrucción del mapa usando resolución VGA con calidad de percepción de profundidad MEDIUM y un filtro de tipo alto. a) Nube de puntos b) Imagen texturizada con efectos de iluminación.



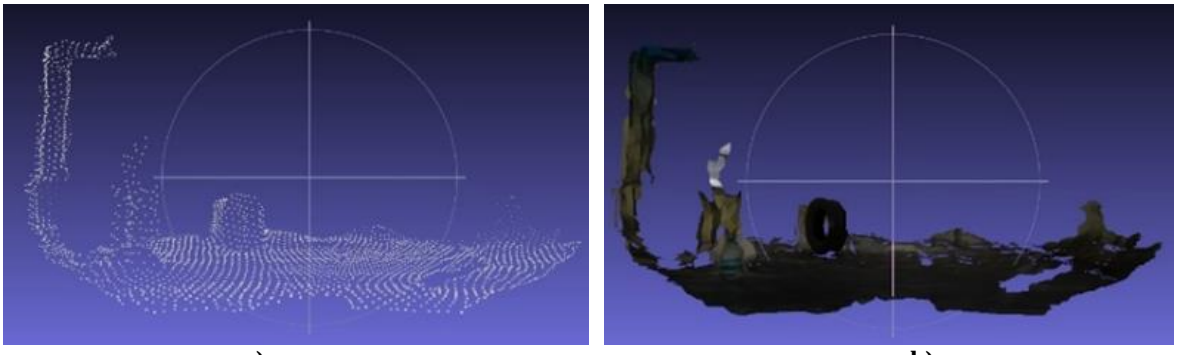
a) **b)**

Figura 178. Proceso de reconstrucción del mapa usando resolución VGA con calidad de percepción de profundidad QUALITY y un filtro de tipo alto. a) Nube de puntos b) Imagen texturizada con efectos de iluminación.



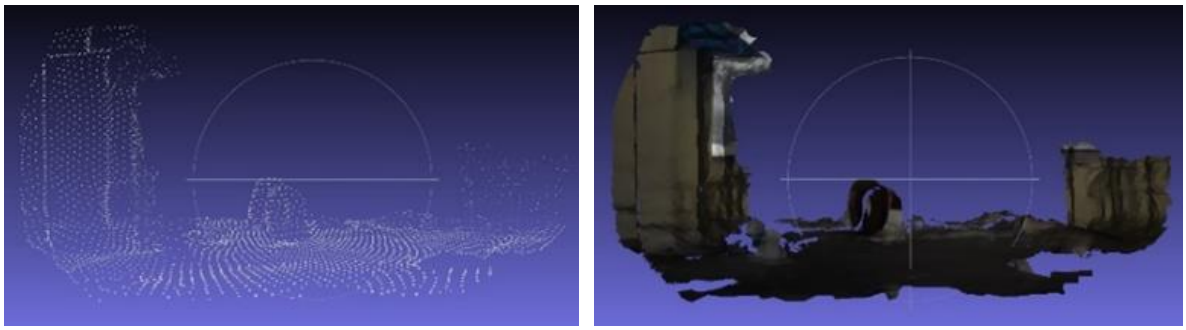
a) **b)**

Figura 179. Proceso de reconstrucción del mapa usando resolución HD720 con calidad de percepción de profundidad PERFORMANCE y un filtro de tipo alto. a) Nube de puntos b) Imagen texturizada con efectos de iluminación.



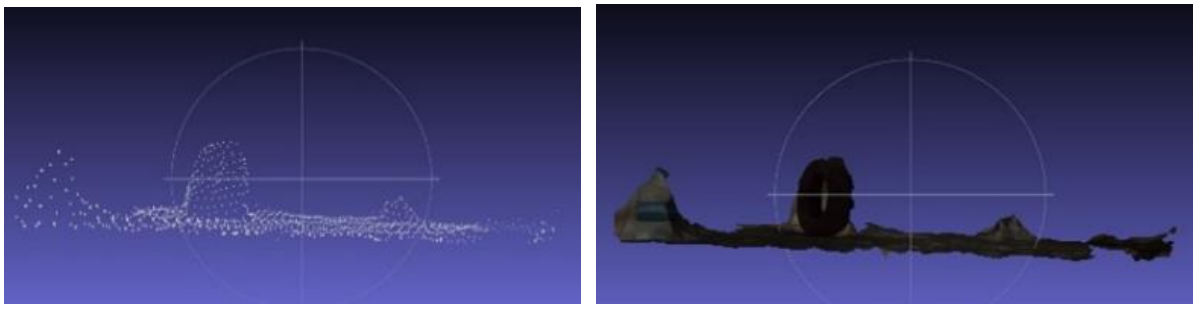
a) **b)**

Figura 180. Proceso de reconstrucción del mapa usando resolución HD720 con calidad de percepción de profundidad MEDIUM y un filtro de tipo alto. a) Nube de puntos b) Imagen texturizada con efectos de iluminación.



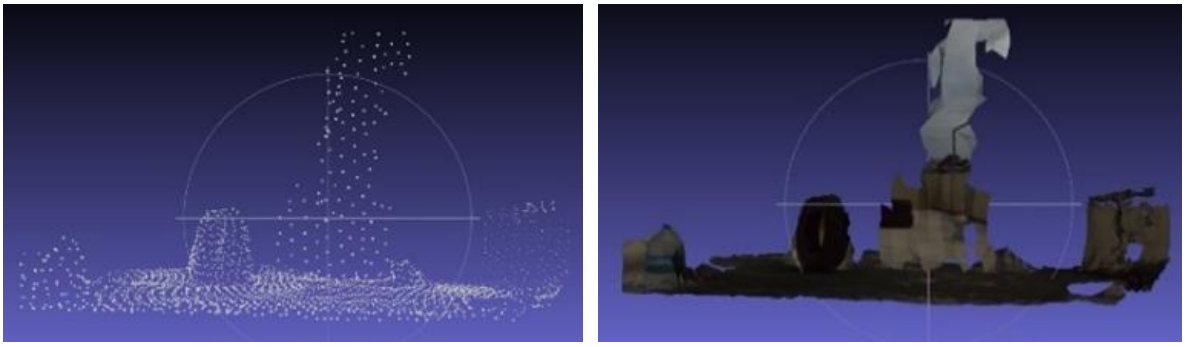
a) **b)**

Figura 181. Proceso de reconstrucción del mapa usando resolución HD720 con calidad de percepción de profundidad QUALITY y un filtro de tipo alto.
a) Nube de puntos b) Imagen texturizada con efectos de iluminación.



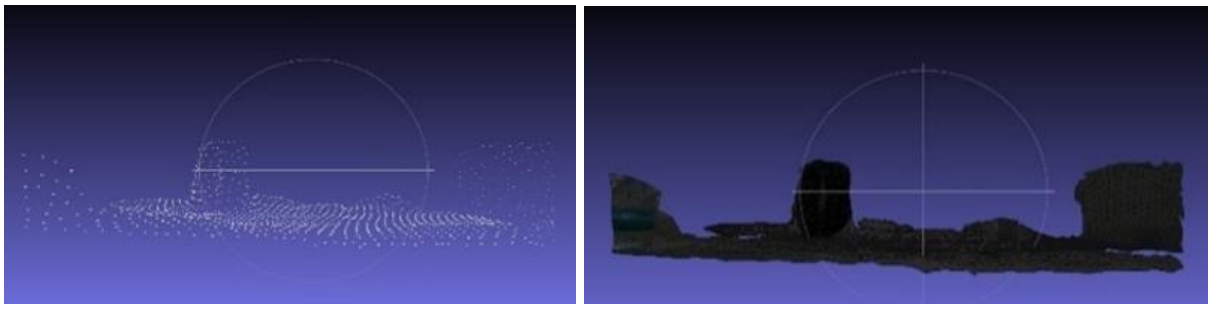
a) **b)**

Figura 182. Proceso de reconstrucción del mapa usando resolución HD1080 con calidad de percepción de profundidad PERFORMANCE y un filtro de tipo alto.
a) Nube de puntos b) Imagen texturizada con efectos de iluminación.



a) **b)**

Figura 183. Proceso de reconstrucción del mapa usando resolución HD1080 con calidad de percepción de profundidad MEDIUM y un filtro de tipo alto.
a) Nube de puntos b) Imagen texturizada con efectos de iluminación.



a) **b)**

Figura 184. Proceso de reconstrucción del mapa usando resolución HD1080 con calidad de percepción de profundidad QUALITY y un filtro de tipo alto. a) Nube de puntos b) Imagen texturizada con efectos de iluminación.

Los resultados de las pruebas para exterior se muestran desde la (Figura 176) hasta la (Figura 184) y se cambian los parámetros de igual manera que en las pruebas para interior. Se muestra que al usar mayores distancias la cámara no puede graficar objetos muy alejados, sin embargo, para nuestra aplicación su capacidad de percepción es aceptable, en este ejemplo logra captar en todos los casos la primera llanta. A diferencia del entorno interior no se logra diferenciar mucho entre las resoluciones y las calidades y eso es debido a los pocos vértices que se capturan. Si se aumenta la calidad de percepción de profundidad se captura parte del suelo.

Mientras más alta es la resolución más se tardan más en procesar, y aunque en estas pruebas se estableció un tiempo de mapeo igual para todos se nota que en ciertas ocasiones las calidades más altas no alcanzan a capturar ciertas imágenes e incluso fallan en ciertas texturas.

Debido a los resultados mostrados en este capítulo y a los resultados mostrados anteriormente incluyendo los del capítulo III, se decide usar una resolución VGA, con una calidad de percepción de profundidad PERFORMANCE, priorizando la capacidad de procesamiento y el tiempo de ejecución para la obtención de las imágenes.

5.3. Modos de trabajo

Se plantea usar dos modos de trabajo para la obtención del mapeado espacial estos tipos son:

- Modo tripulado
- Modo no tripulado

Estos modos difieren según a la necesidad, de operación si se quiere obtener un mapa luego de realizar algunas trayectorias o si se quiere percibir mediante se va navegando.

En modo tripulado no se usará los algoritmos para la navegación autónoma, sino que el vehículo será operado por un usuario. Aprovechando el ahorro computacional se procede a usar una resolución de captura más alta para tener un mejor detalle de visión del entorno. Se mostrará en pantalla en tiempo real las imágenes capturada por la cámara y se agrega la opción de realizar el mapeado 3D cuando el operador lo decida usando la barra espaciadora y se da por terminado el mapeado presionando nuevamente este botón. Además, se graficará la malla en tiempo real sobre las imágenes durante el mapeo y al finalizar el mapeado se guarda en un archivo OBJ para su futura visualización.

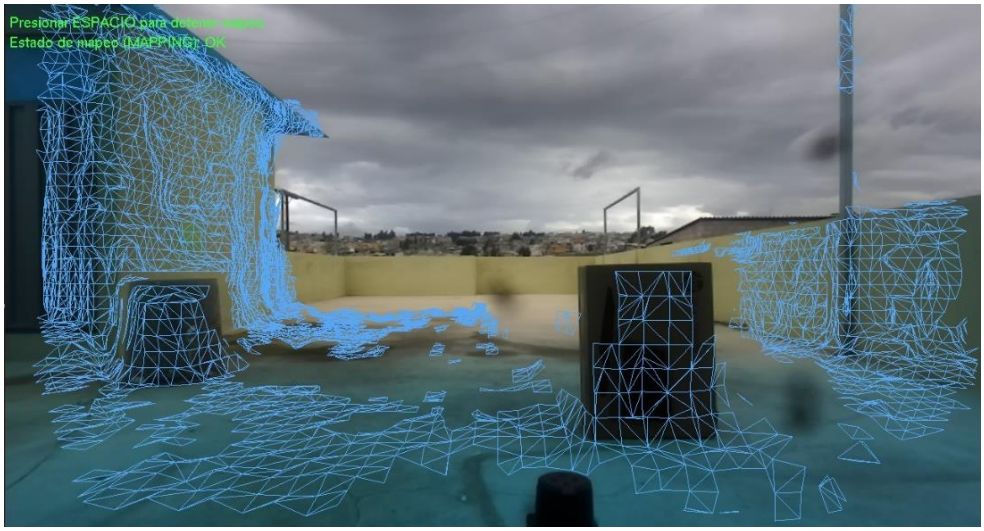


Figura 185. Pantalla de visualización de mapeo espacial en modo tripulado

En modo no tripulado no existe la necesidad de que un operador mire la malla en tiempo real por lo cual no se muestran las imágenes en pantalla, simplemente realiza esta acción de mapear

en paralelo al proceso de navegación autónoma y al terminar la navegación guarda automáticamente el mapa en un archivo OBJ para que pueda ser visualizada.

5.4. Filtros de optimización

Para mejorar el rendimiento, a menudo es conveniente reducir el número de polígonos por malla después de la captura. El filtro de malla permite optimizar modelos en 3D para reducir polígonos y preservar las funciones geométricas deseables. Hay tres pre-ajustes disponibles: alto, medio y bajo. El modo de filtrado bajo simplemente rellena los orificios y limpia los valores atípicos de la malla, mientras que los otros dos modos realizan la aniquilación de la malla (Stereolabs, 2018).

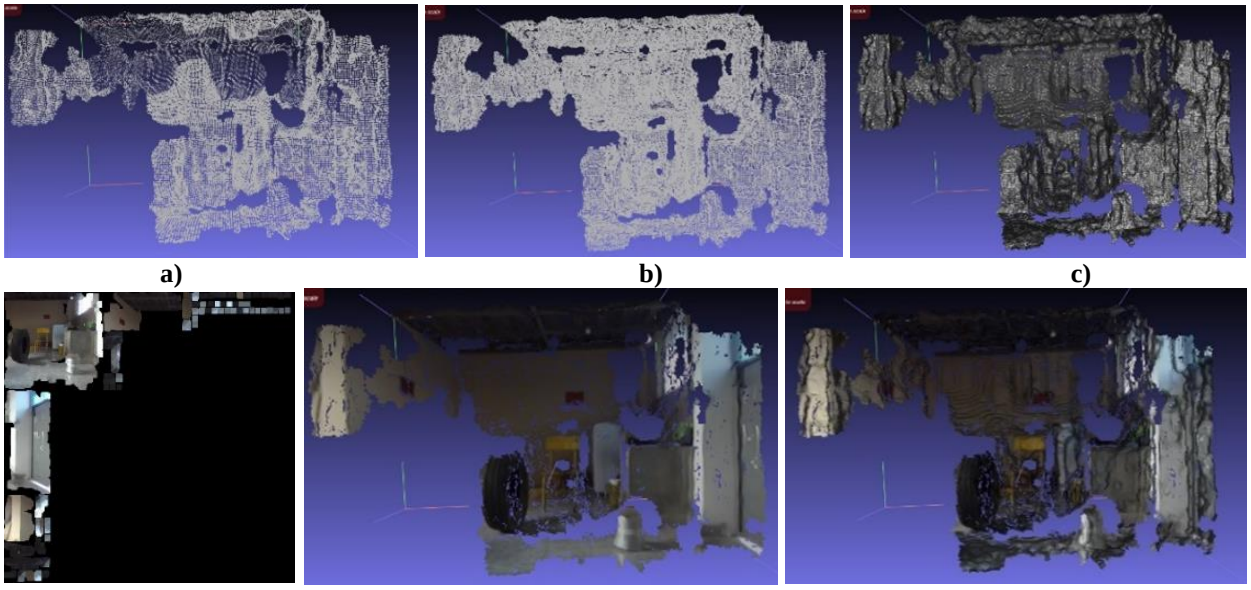
La malla se puede colorear proyectando las imágenes 2D capturadas durante el mapeo espacial a la superficie del modelo 3D. Este paso se llama texturizado de malla. Para crear la textura, se graba un subconjunto de las imágenes de la cámara izquierda durante el mapeo. Cada imagen se procesa y ensambla en un solo mapa de texturas. Este mapa de textura se proyecta en cada cara de la malla 3D usando coordenadas UV generadas automáticamente (Stereolabs, 2018).

Con el fin de caracterizar el mapeo y encontrar los ajustes adecuados para este trabajo se realiza pruebas en diferentes entornos y cambiando los principales parámetros, los entornos son los mismos que se utilizó en el Capítulo III, la cámara no realiza ningún movimiento para esta prueba, se queda estática y mapea desde una sola posición.

Este filtro aparentemente puede ser necesario tenerlo siempre y en su modo alto, sin embargo, esto no es del todo cierto ya que según las pruebas que se ha realizado en algunas veces falla rellenando los vértices y en otras elimina contenido importante, mostramos los resultados cambiando este parámetro para la caracterización del filtro.

5.4.1. Caracterización del filtro de optimización para modo no tripulado

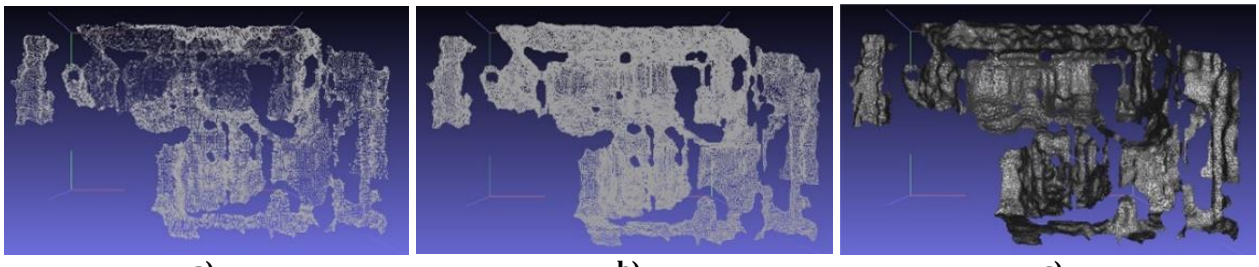
En el modo no tripulado se usa como parámetros la resolución de la cámara en modo VGA, con calidad de percepción de profundidad PERFORMANCE, por lo tanto con estas caracteristicas y tomando en cuenta que mientras se realiza el mapeado espacial se ejecuta en paralelo la planificación de trayectorias se procede a verificar el resultado ante los diferentes tipos de filtros en un entorno interior.



a) **b)** **c)**

d) **e)** **f)**

Figura 186. Proceso de reconstrucción del mapa usando un filtro BAJO.
a) Nube de puntos b) Creación de malla triangular c) Reconstrucción de objeto 3D
d) captura de imágenes e) aplicación de texturizado a partir de la imagen f) resultado final, mostrando efectos de iluminación.

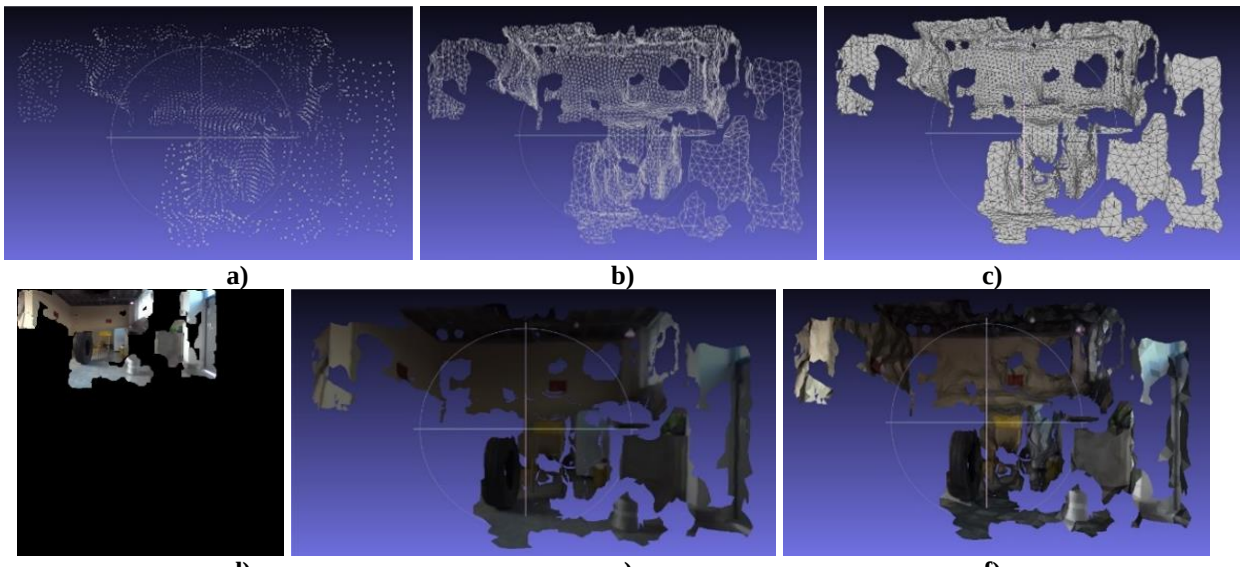


a) **b)** **c)**

CONTINÚA

d) **e)** **f)**

Figura 187. Proceso de reconstrucción del mapa usando un filtro MEDIO.
a) Nube de puntos b) Creación de malla triangular c) Reconstrucción de objeto 3D
d) captura de imágenes e) aplicación de texturizado a partir de la imagen f) resultado final, mostrando efectos de iluminación.



a) **b)** **c)**

d) **e)** **f)**

Figura 188. Proceso de reconstrucción del mapa usando un filtro ALTO.
a) Nube de puntos b) Creación de malla triangular c) Reconstrucción de objeto 3D
d) captura de imágenes e) aplicación de texturizado a partir de la imagen f) resultado final, mostrando efectos de iluminación.

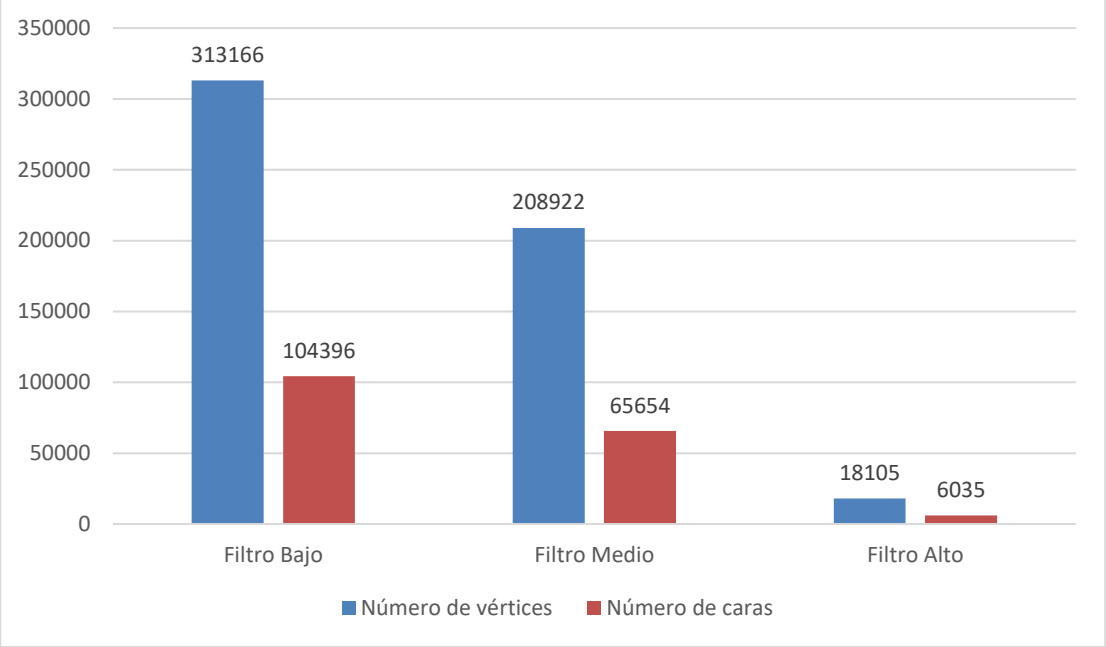


Figura 189. Número de vértices y caras producidas según el tipo de filtro para modo no tripulado

De los resultados mostrados desde la (Figura 186) hasta la (Figura 188). Se pudo notar una clara diferencia al usar los filtros, los filtros bajo y medio. Por defecto se toman una gran cantidad de puntos y crean muchos triángulos para la malla y como no se encuentran congruencias entre las triangulaciones dentro del plano no se unen y generan vacíos en la imagen 3D.

5.4.2. Caracterización del filtro de optimización para modo tripulado

En el modo tripulado se usa como parámetros la resolución de la cámara en modo HD720, con calidad de percepción de profundidad PERFORMANCE, por lo tanto con estas caracteristicas y tomando en cuenta que unicamente se realiza el mapeado y ningun otro proceso se procede a verificar el resultado ante los diferentes tipos de filtros en un entorno exterior.

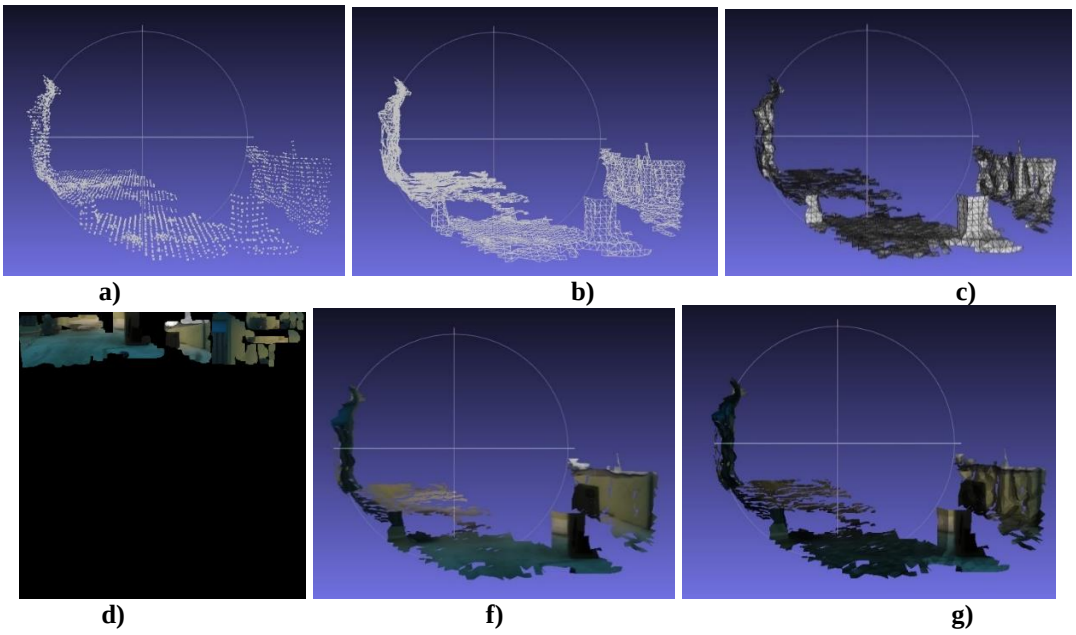


Figura 190. Proceso de reconstrucción del mapa usando un filtro BAJO.
a) Nube de puntos b) Creación de malla triangular c) Reconstrucción de objeto 3D
d) captura de imágenes e) aplicación de texturizado a partir de la imagen
f) resultado final, mostrando efectos de iluminación.

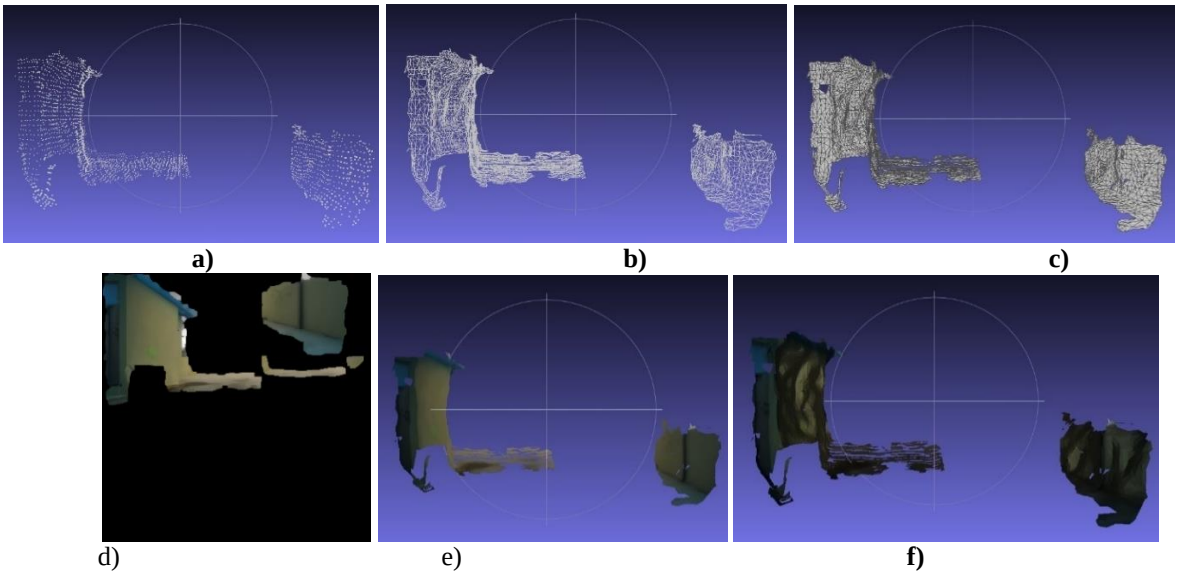
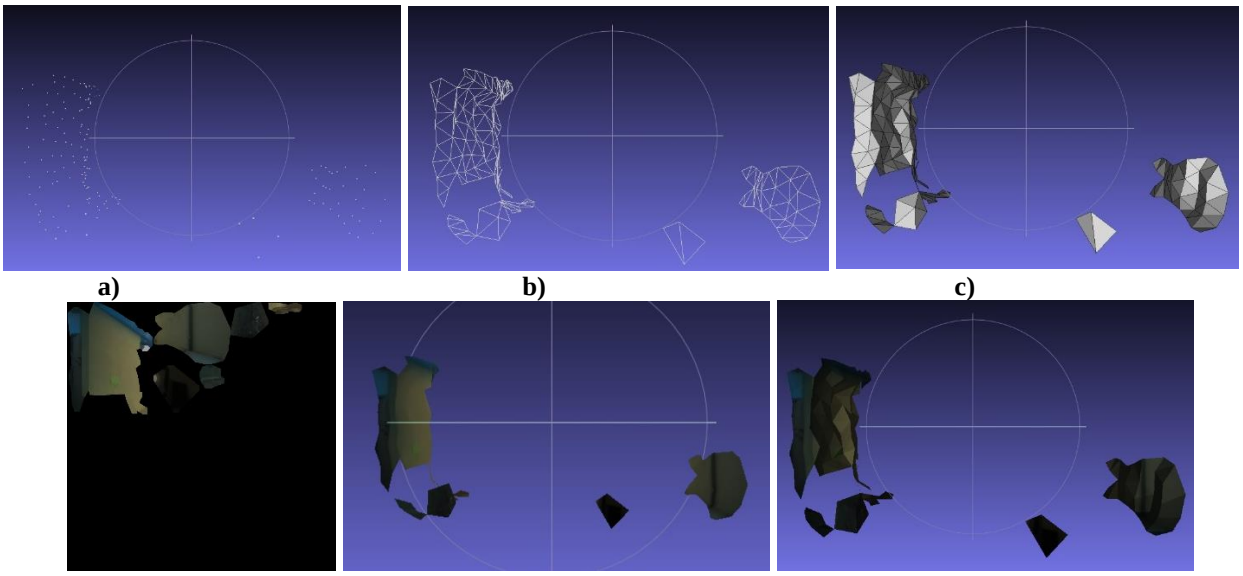


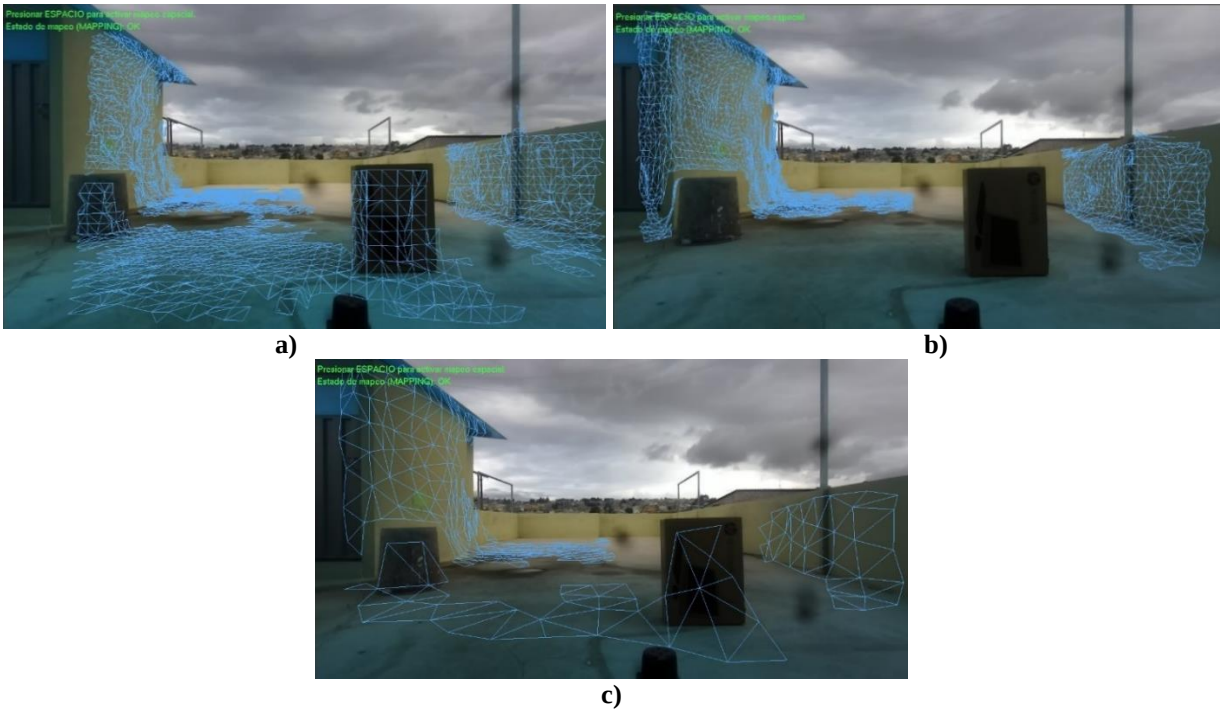
Figura 191. Proceso de reconstrucción del mapa usando un filtro MEDIO.
a) Nube de puntos b) Creación de malla triangular c) Reconstrucción de objeto 3D
d) captura de imágenes e) aplicación de texturizado a partir de la imagen
f) resultado final, mostrando efectos de iluminación.



a) **b)** **c)**

d) **e)** **f)**

Figura 192. Proceso de reconstrucción del mapa usando un filtro ALTO.
a) Nube de puntos b) Creación de malla triangular c) Reconstrucción de objeto 3D
d) captura de imágenes e) aplicación de texturizado a partir de la imagen
f) resultado final, mostrando efectos de iluminación.



a) **b)**

c)

Figura 193. Visualización en tiempo real de mapeado en modo tripulado.
a) Usando filtro BAJO b) Usando filtro MEDIO c) usando filtro ALTO

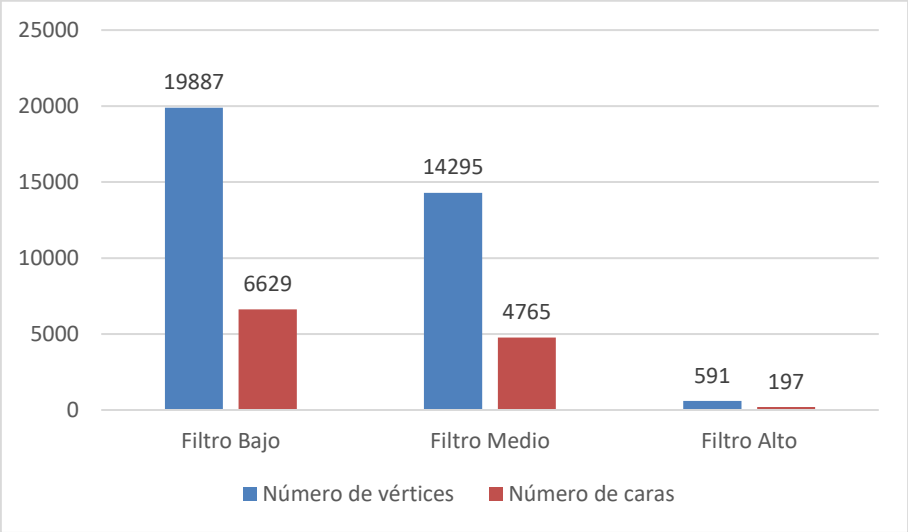


Figura 194. Número de vértices y caras producidas según el tipo de filtro para modo tripulado

Los resultados mostrados para el modo tripulado en un entorno exterior muestran una diferencia clara, entre los filtros. La información que se recibe usando el filtro alto es demasiada baja por lo cual se decide no usar la misma y como el fin de asegurar capturar la mayor información posible se decide utilizar el filtro bajo. El modo tripulado también cuenta con la visualización en tiempo real del mapeado, la información es guardada de la misma manera, pero la tripulante puede apreciar la cantidad de información que esta almacenando.

5.5. Tracking

Al tracking o también llamado seguimiento posicional se usa para conocer la posición del vehículo den seis grados de liberta, en el Capítulo IV, se desarrolló un algoritmo basado en conceptos cinemáticos y dinámicos para estimar los movimientos, pero en esta ocasión lo que se hace es obtener la posición en todo momento, pero usando sensores que nos permitan conocer la pose en cada instante de tiempo.

La cámara ZED tiene incorporada también con señores que permiten realizar el seguimiento de sus movimientos, y se tiene la disponibilidad de obtener estos datos por medio de la SDK.

ZED utiliza el seguimiento visual de su entorno para comprender el movimiento del usuario o sistema que lo sostiene. A medida que la cámara se mueve en el mundo real, informa su nueva posición y orientación. Esta información se llama la pose de la cámara con seis grados de libertad. La información de la posición se emite a la velocidad de cuadro de la cámara, hasta 100 veces por segundo en modo VGA La API ZED devuelve información de la postura para cada fotograma. La pose se da para el ojo izquierdo de la cámara. Es relativo a un marco de coordenadas de referencia, generalmente el Marco Mundial que se fija en el espacio (Stereolabs, 2018).

Para obtener la pose es necesario especificar previamente los parámetros de coordenadas y su sistema de medición los datos devueltos son los siguientes (Stereolabs, 2018):

- Posición: la ubicación de la cámara en el espacio está disponible como un vector $[X, Y, Z]$. Su norma representa la distancia total recorrida entre la posición actual de la cámara y el marco de coordenadas de referencia.
- Orientación: la orientación de la cámara en el espacio está disponible como un cuaternión $[X, Y, Z, W]$. Un cuaternión se puede manipular directamente para reflejar guiñada, cabeceo y balanceo en diferentes marcos de coordenadas.
- La clase POSE también contiene marca de tiempo, valor de confianza y una matriz de rotación que describe la rotación de la cámara con respecto al Marco Mundial.

La pose es devuelta al algoritmo de planificación de trayectorias, para que pueda comparar la POSE estimada usando el método matemático incluido en el algoritmo con la verdadera posición del vehículo obtenida del proceso del tracking de la cámara y de esta manera comprobar que se esté siguiendo la ruta correcta.

CAPITULO VI

6. ANÁLISIS Y DISCUSIÓN DE RESULTADOS

Este capítulo tiene la finalidad de evaluar el desempeño del sistema completo, considerando los escenarios en el cual puede estar el UGV. Una vez que los parámetros para el sistema fueron seleccionados y explicados en capítulos anteriores se plantean un conjunto de escenarios de navegación para diferentes entornos planteados a continuación:

- Entorno Cerrado Terreno regular plano, sin gradas o similares, con paredes separadas a una distancia mínima de 3 metros a lo ancho.
- Entorno Abierto: Terreno regular plano, sin gradas o similares.

A un entorno interior nos referimos a un entorno que cuenta con paredes cercanas entre si e iluminación artificial, por ejemplo, una casa, un salón de clases o una cochera, los exteriores vienen a ser espacios amplios donde el terreno es un poco más irregular y la luz es natural como por ejemplo la calle un patio o un parque.

Un factor determinante para el desempeño del vehículo es la variabilidad del entorno, ya que este varia tanto con el tipo de suelo, la dimensión, la iluminación y el clima. El tipo de suelo tiene que ser uniforme, las llantas del vehículo están diseñadas para transitar por lugares planos, pero con el rozamiento suficiente. El UGV no pode ir por pendientes inclinadas ya que no se considera la dinámica lo cual dificultaría mantener las posiciones, por lo tanto, se frena debido a la energía potencial gravitatoria. El vehículo puede navegar en espacios pequeños, siempre y cuando el espacio permita demostrar el funcionamiento, desplazamiento, giros y evasión de obstáculos. La iluminación es un factor determinante debido a que la estéreo cámara en la noche tiene un mal funcionamiento, por lo tanto, trabajar en exteriores a la noche no es posible. Por otro lado, en

interiores con iluminación constante se garantiza un correcto funcionamiento del sistema. Cuando llueve el vehículo no podrá navegar debido a que la computadora está colocada sobre el asiento del vehículo y está a la intemperie. Adicionalmente se debe considerar obstáculos que sean por lo menos de treinta centímetros de altura lo cual garantiza que el sensor laser los detecte. Tomando en cuenta las limitantes y los objetivos a conseguir se realizan los siguientes escenarios para los entornos antes mencionados:

Tabla 26
Escenarios definidos para pruebas del sistema

Escenario	Entorno	Código
Sin Obstáculos	Cerrado	CSO
Obstáculo Derecha		COD
Obstáculo Izquierda		COI
Obstáculo Central		COC
Doble Obstáculo		CDO
Obstáculo Móvil		COM
Sin Obstáculos	Abierto	ASO
Obstáculo Derecha		AOD
Obstáculo Izquierda		AOI
Obstáculo Central		AOC
Doble Obstáculo		ADO
Obstáculo Móvil		AOM

El entorno cerrado en el cual se efectuaron las pruebas es una cochera como se muestra en la (Figura 195), su dimensión es de 5.6 x 2.66 metros, lo cual lo caracteriza como un espacio corto y restrictivo para el vehículo. El mapa bidimensional se muestra en la (Figura 196), donde se aprecia lo percibido y el espacio real de trabajo.

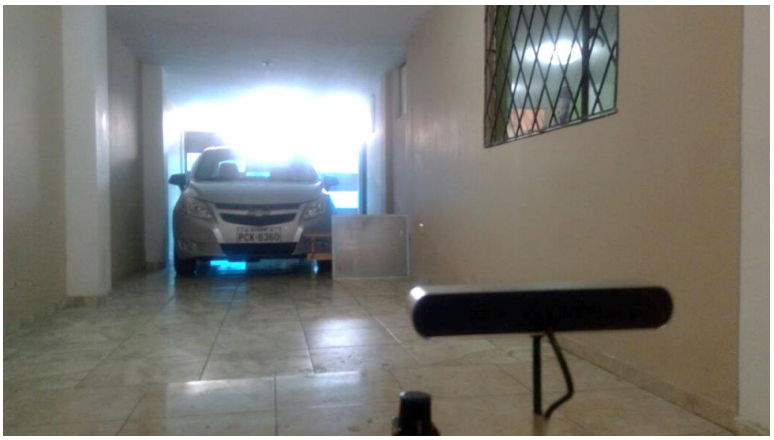


Figura 195. Escenario de prueba para entorno cerrado

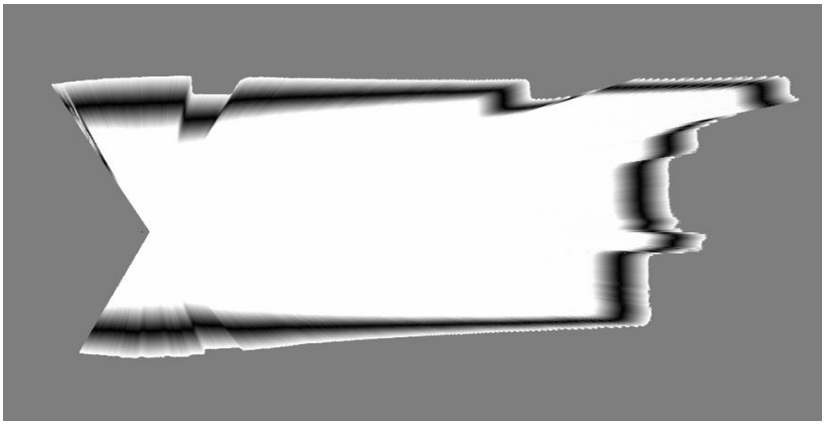
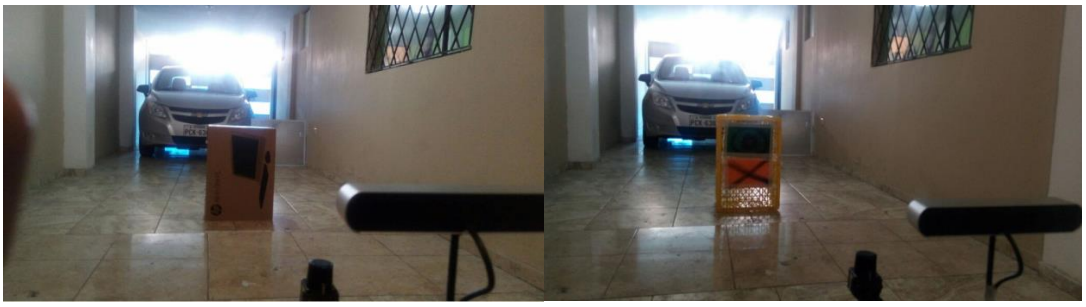


Figura 196. Mapa bidimensional para el entorno cerrado

Se utiliza dos obstáculos, un cartón de 57x42cm como se muestra en la (Figura 197 a)) y una canasta de 60x40cm como se muestra en la (Figura 197 b)).



a) b)

Figura 197. Obstáculos definidos para utilizar en las pruebas
a) Obstaculo de carton, b) Obstaculo canasta

Para evaluar y comparar los escenarios se establecen criterios de comparación los cuales permiten determinar los resultados del sistema y son:

- Convergencia de Trayectoria. - Si el UGV llega al destino o meta indicada.
- Seguimiento de trayectoria. - Determinada por la cámara estéreo.
- Colisión ante obstáculos.
- Colisión ante obstáculos dinámicos.
- Recuperación de trayectoria.
- Cantidad de segmentos locales.
- Cantidad de árboles realizados.
- Tiempo de ejecución.

Para los escenarios se reflejan ciertos resultados en función de la planificación de trayectorias.

6.1. Resultados de la planificación en los escenarios

6.1.1. Escenario CSO

La pose de inicio fue (400 400 1.5708), la pose final fue (700 400 0) y la pose final de la cámara fue (566 401 92.8441). La trayectoria seguida fue la siguiente:

0: 400 400 1.5708

1: 563 399 1.57082

2: 696 398 1.57085

Posterior la trayectoria realizada fue la mostrada y su resultado se muestra en la (Figura 198). Por otro lado, el seguimiento de trayectoria simulado se muestra en la (Figura 199)

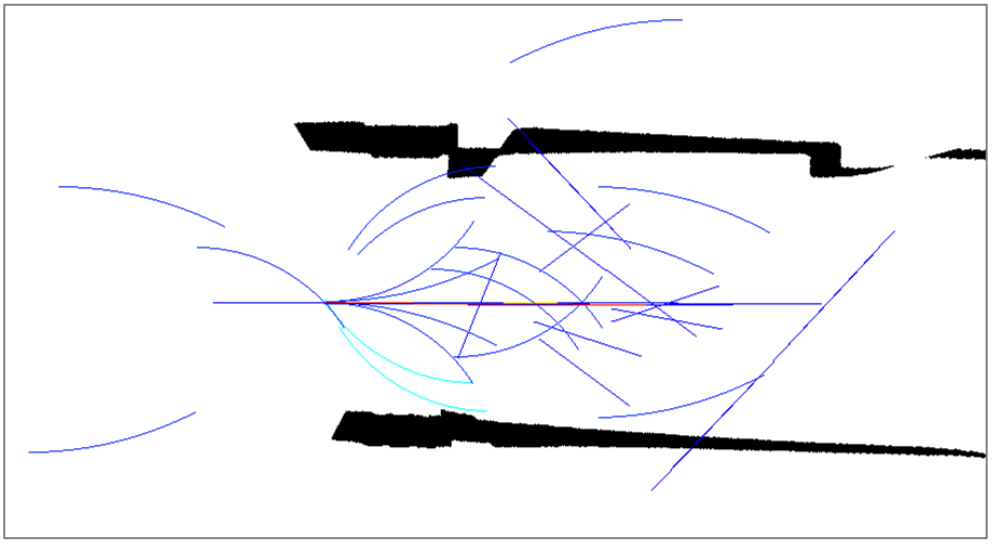


Figura 198. Trayectoria realizada para escenario sin obstáculo

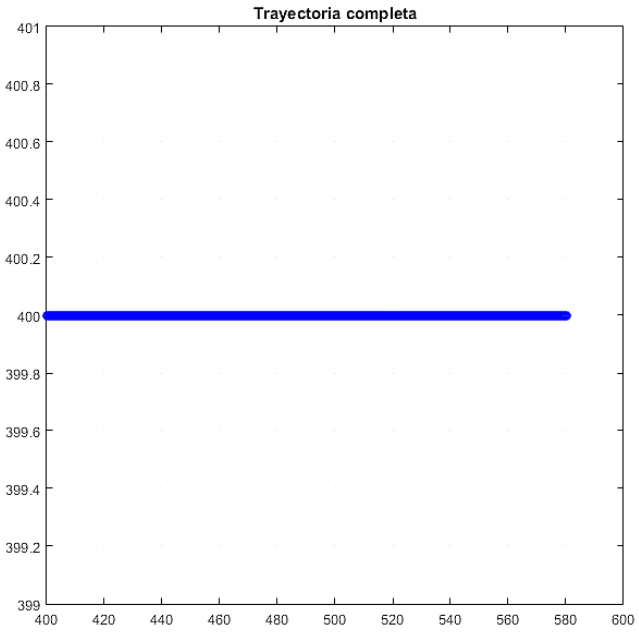


Figura 199. Seguimiento de trayectoria simulado escenario sin obstáculos

6.1.2. Escenario COD

La pose de inicio fue (400 400 1.5708), la pose final fue (800 400 0) y la pose final de la cámara fue (672 428 99.594). La trayectoria seguida fue la siguiente:

0: 400 400 1.5708

1: 493 370 2.17741

2: 547 345 1.81344

3: 487 348 1.44947

4: 546 355 1.44948

5: 620 383 0.964189

6: 709 427 1.25807

7: 761 455 0.894097

8: 669 419 1.50071

9: 726 433 1.13674

10: 774 467 0.772774

Posterior la trayectoria realizada fue la mostrada en la (Figura 200). El seguimiento de trayectoria simulado es el de la (Figura 201).

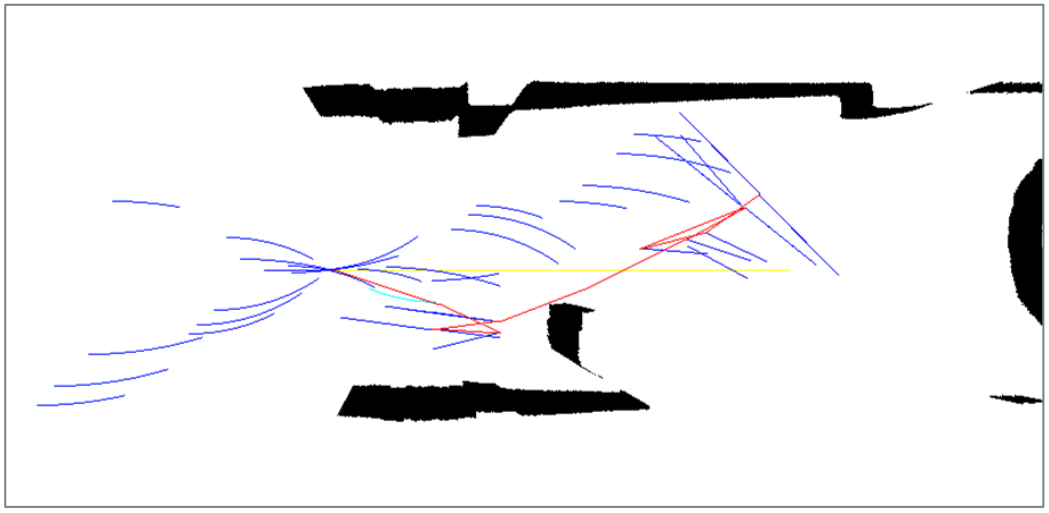


Figura 200. Trayectoria realizada para escenario obstáculo derecha

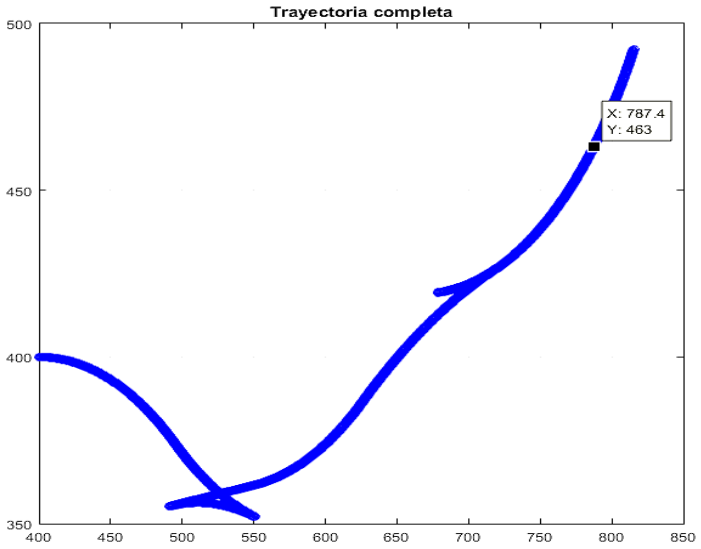


Figura 201. Seguimiento de trayectoria simulado escenario obstáculo derecha

6.1.3. Escenario COI

La pose de inicio fue (400 400 1.5708), la pose final fue (800 400 0) y la pose final de la cámara fue (711 387 85.8162). La trayectoria seguida fue la siguiente:

0: 400 400 1.5708

1: 493 370 2.17741

2: 586 340 1.5708

3: 644 350 1.20683

4: 697 376 1.0305

5: 792 399 1.63712

Posterior la trayectoria simulada fue la mostrada en la (Figura 202). El seguimiento de trayectoria fue el mostrado en la (Figura 203).

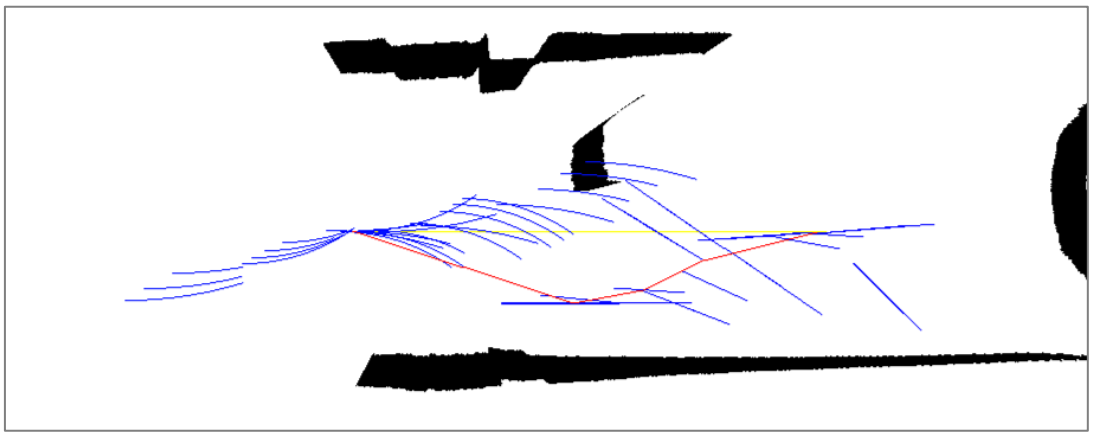


Figura 202. Trayectoria realizada para escenario obstáculo izquierda

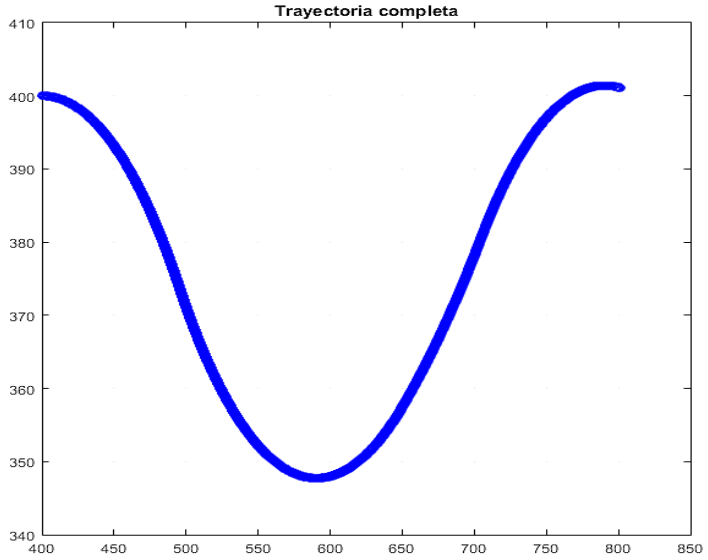


Figura 203. Seguimiento de trayectoria simulado escenario obstáculo izquierda

6.1.4. Escenario COC

La pose de inicio fue (400 400 1.5708), la pose final fue (800 400 0). La posición final no fue alcanzada debido al espació y tempo insuficiente.

6.1.5. Escenario CDO

La pose de inicio fue (400 400 1.5708), la pose final fue (900 400 0) y la pose final de la cámara fue (817 463 122.172). La trayectoria seguida fue la siguiente:

0: 400 400 90

1: 513 354 133.446

2: 626 308 90

3: 700 307 90.0007

4: 489 376 48.3431

5: 595 440 69.3906

6: 568 336 68.7453

7: 654 385 51.9073

8: 700 442 25.8399

9: 683 369 -0.227579

10: 665 441 -26.295

11: 647 334 116.542

12: 768 323 73.0961

13: 878 380 52.0486

14: 810 348 78.1161

15: 932 373 78.1173

16: 857 365 90.7458

17: 931 364 90.7465

Posterior la trayectoria simulada fue la mostrada en la (Figura 204). La trayectoria realizada se gráfica encima ya que realiza cuatro árboles. El seguimiento de trayectoria simulado es el de la (Figura 205).

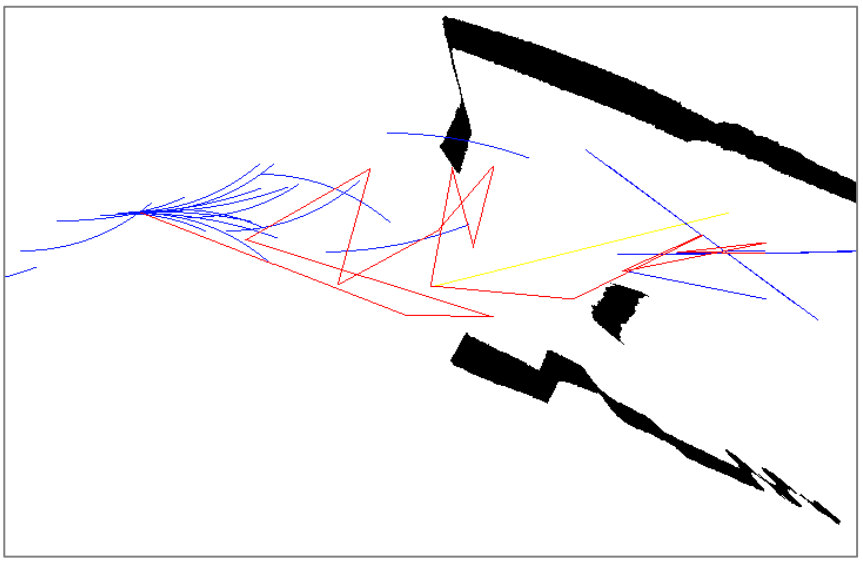


Figura 204. Trayectoria realizada para escenario doble obstáculo

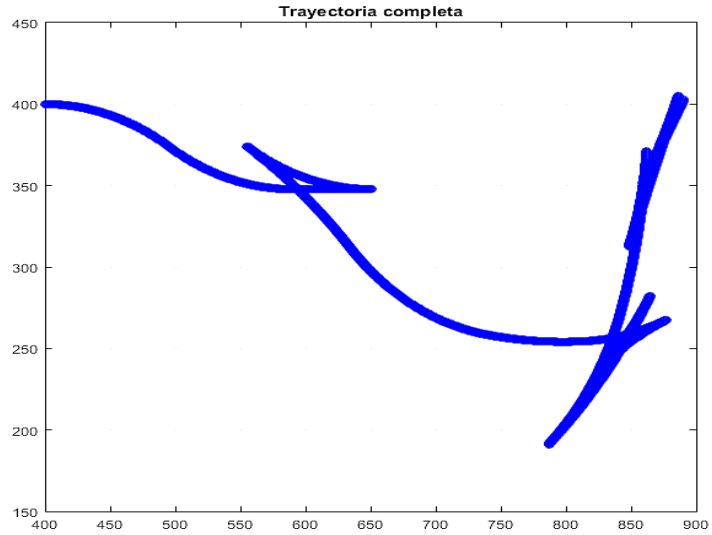


Figura 205. Seguimiento de trayectoria simulado escenario doble obstáculo

6.1.6. Escenario COM

La pose de inicio fue (400 400 1.5708), la pose final fue (800 400 0) y la pose final de la cámara fue (609 352 77.8701). La trayectoria seguida fue la siguiente:

0: 400 400 1.5708

1: 538 324 2.56565

2: 458 388 1.93477

3: 619 368 1.45281

4: 751 357 1.8466

5: 648 352 1.21572

6: 749 372 1.52135

7: 844 408 0.890473

8: 689 379 1.88532

9: 791 379 1.25444

Posterior la trayectoria simulada fue la mostrada en la (Figura 206). El seguimiento de trayectoria simulado es el de la (Figura 207).

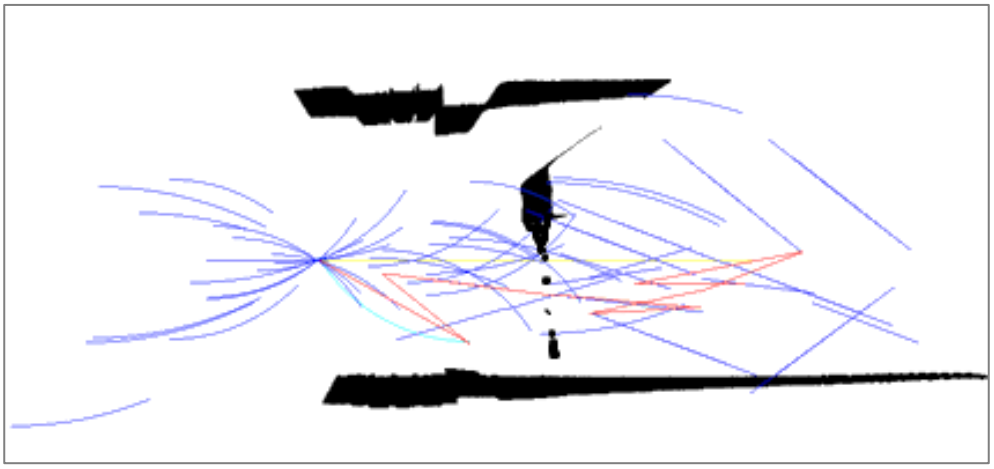


Figura 206. Trayectoria realizada para escenario onstáculo en movimiento

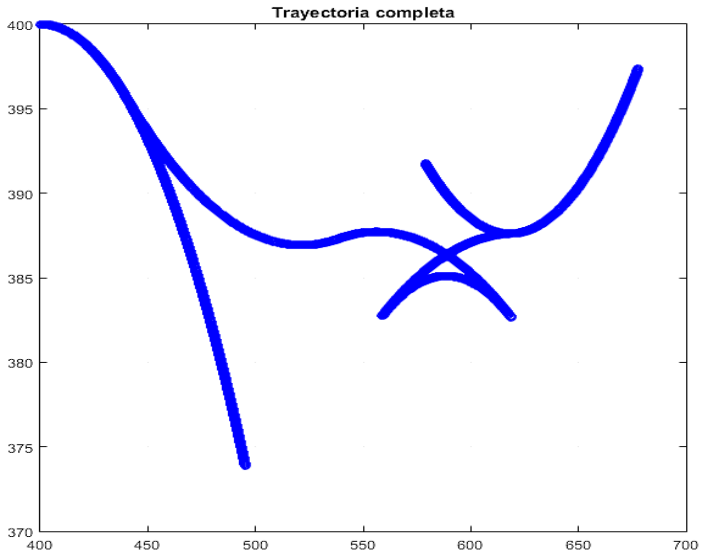


Figura 207. Seguimiento de trayectoria simulado escenario obstáculo en movimiento

En este caso el objeto fue una persona moviéndose de derecha a izquierda y deteniéndose en la posición izquierda.

6.1.7. Escenario ASO

La pose de inicio fue (400 400 1.5708), la pose final fue (800 400 0) y la pose final de la cámara fue (472 396 94.47). La trayectoria seguida fue la siguiente:

0: 400 400 90 0 0

1: 513 354 133.446 50 20

2: 626 308 90 50 -20

3: 698 324 63.9326 30 -20

4: 797 398 42.8851 50 -10

Posterior la trayectoria realizada fue la mostrada en la (Figura 208). Por otro lado, el seguimiento de trayectoria simulado se muestra en la (Figura 209):

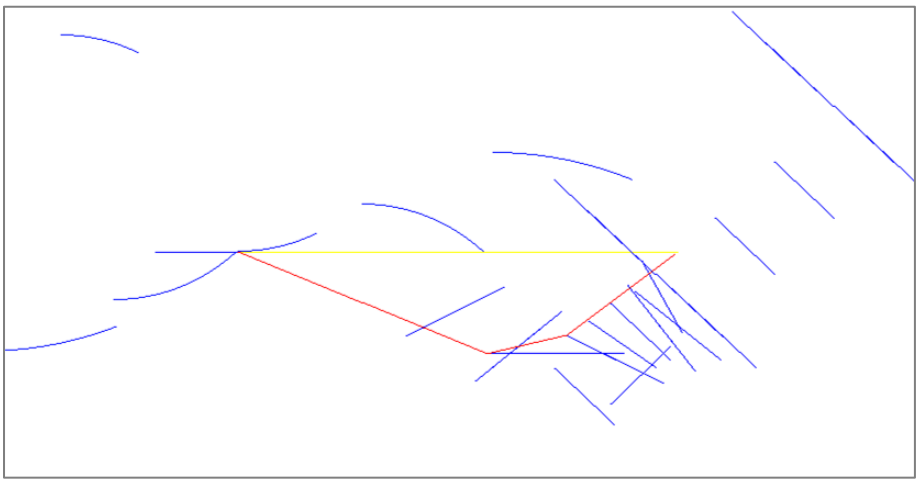


Figura 208. Trayectoria realizada para escenario sin obstáculo abierto

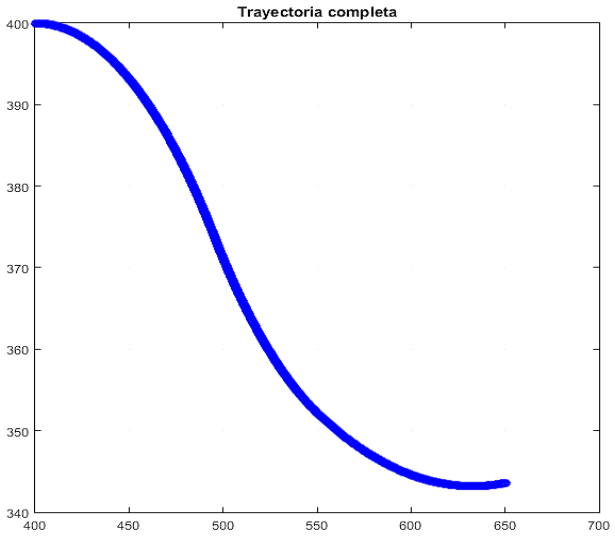


Figura 209. Seguimiento de trayectoria simulado escenario sin obstáculos abierto

6.1.8. Escenario AOD

La pose de inicio fue (400 400 1.5708), la pose final fue (800 400 0) y la pose final de la cámara fue (777 223 62.673). La trayectoria seguida fue la siguiente:

0: 400 400 90 0 0

1: 472 416 63.9326 30 -20

2: 566 446 80.7705 40 10

3: 634 474 54.7031 30 -20

4: 689 523 42.0746 30 -10

5: 739 578 42.0753 30 0

6: 665 394 81.1127 30 0

7: 783 366 124.558 50 20

Posterior la trayectoria realizada fue realizada en dos tramos como se muestra en (Figura 210, Figura 211). La trayectoria simulada se muestra en la (Figura 212).

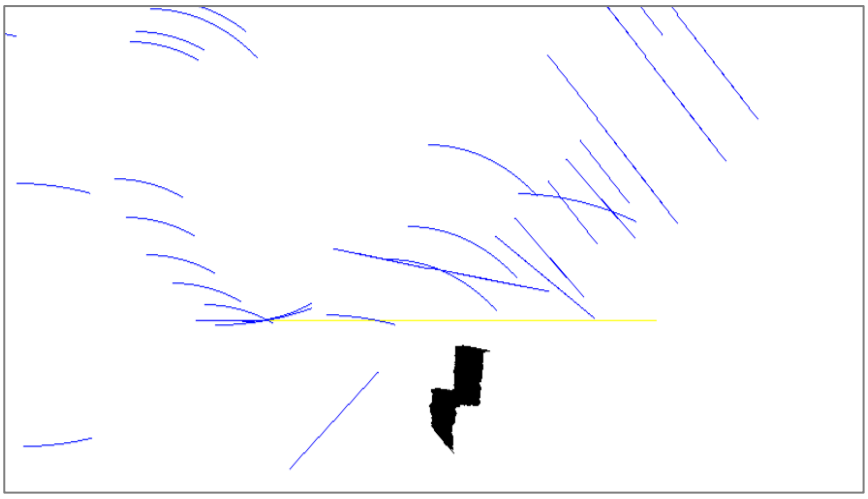


Figura 210. Trayectoria realizada para escenario obstáculo derecha abierto primera parte

Figura 211. Trayectoria realizada para escenario obstáculo derecha abierto segunda parte

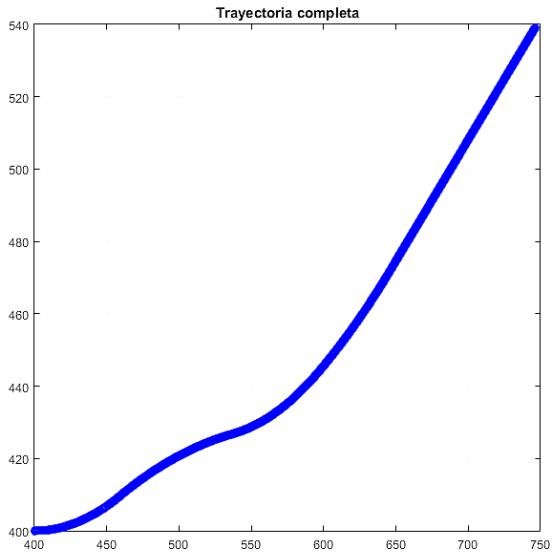


Figura 212. Seguimiento de trayectoria simulado escenario obstáculo derecha abierto

6.1.9. Escenario AOI

La pose de inicio fue (400 400 1.5708), la pose final fue (800 400 0) y la pose final de la cámara fue (647 332 62.3806). La trayectoria seguida fue la siguiente:

0: 400 400 90 0 0

1: 511 356 132.403 50 20

2: 571 320 107.809 29 -20

3: 640 313 83.2155 29 -20

4: 706 335 58.6217 29 -20

5: 797 358 92.1202 39.5 20

Posterior la trayectoria simulada fue la mostrada en la (Figura 213). El seguimiento de trayectoria simulado es el de la (Figura 214).

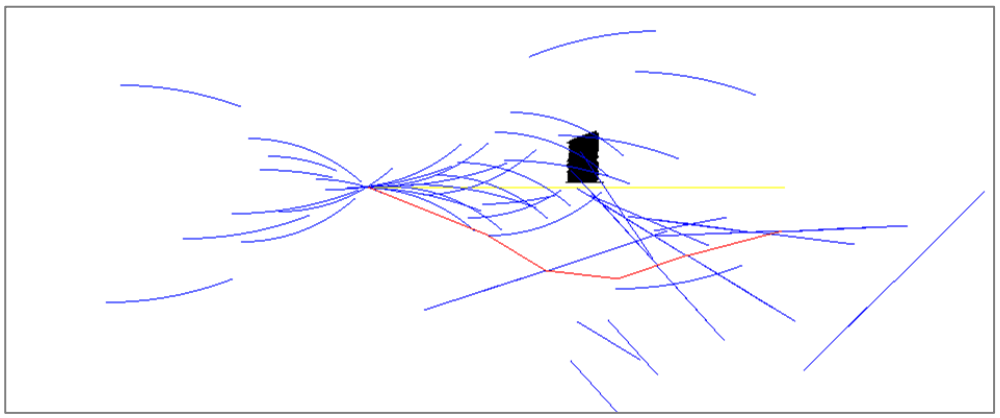


Figura 213. Trayectoria realizada para escenario obstáculo izquierda abierto

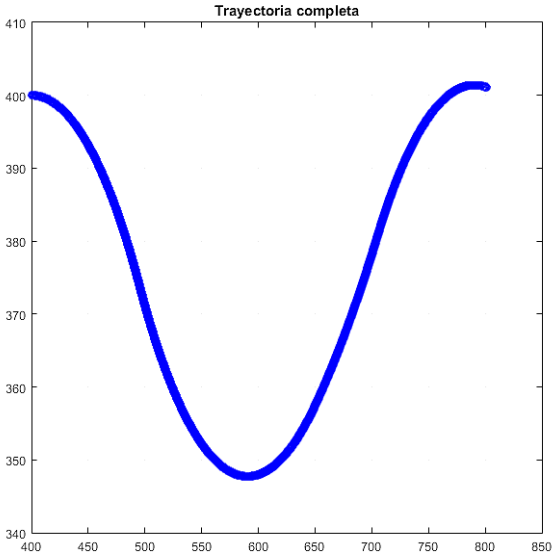


Figura 214. Seguimiento de trayectoria simulado escenario obstáculo izquierda abierto

6.1.10. Escenario AOC

La pose de inicio fue (400 400 1.5708), la pose final fue (800 400 0) y la pose final de la cámara fue (1160 643 137.278). La trayectoria seguida fue la siguiente:

0: 400 400 90 0 0

1: 513 354 133.446 50 20

2: 626 308 90 50 -20

3: 700 316 77.3715 30 -10

4: 599 247 33.9258 -50 20

5: 674 342 91.3545 30 10

6: 788 384 47.9088 50 -20

7: 830 445 21.8414 30 -20

8: 857 514 21.8421 30 0

9: 899 575 47.9095 30 20

10: 954 625 47.9102 30 0

11: 1009 675 47.911 30 0

12: 1083 742 47.9119 40 0

Posterior la trayectoria simulada fue la mostrada en (Figura 215, Figura 216) y corresponde a dos trayectorias. La trayectoria realizada se desarrolló en dos árboles. El seguimiento de trayectoria simulado es el de la (Figura 217).

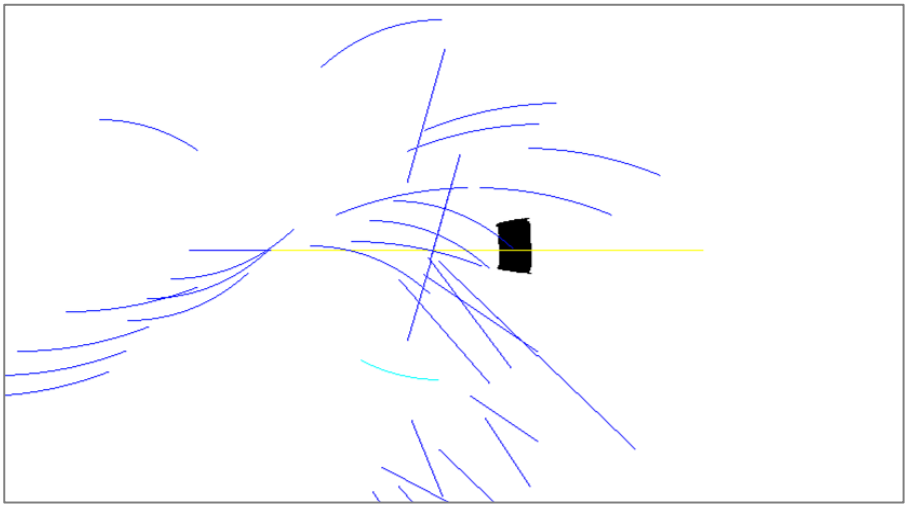


Figura 215. Trayectoria realizada para escenario obstaculo centro abierto parte 1

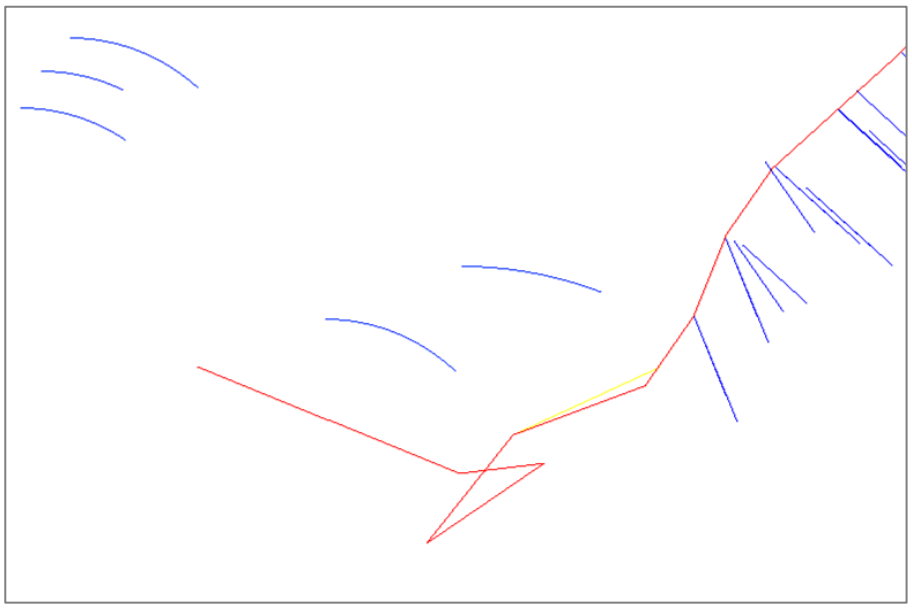


Figura 216. Trayectoria realizada para escenario obstáculo centro abierto parte 2

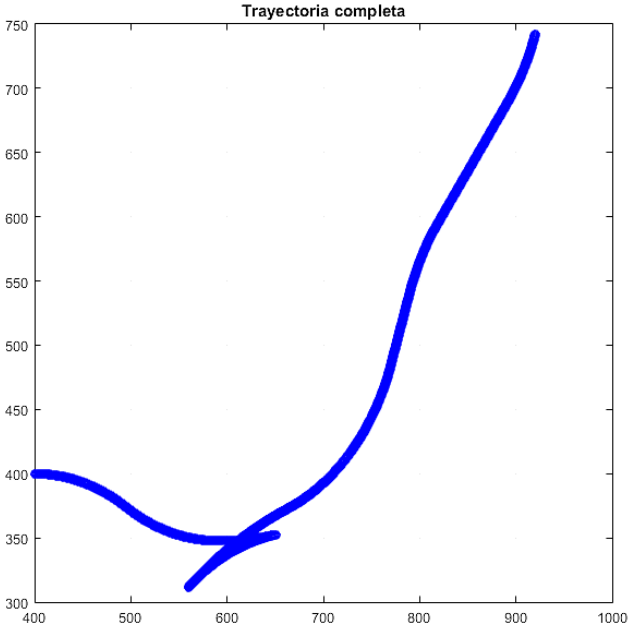


Figura 217. Seguimiento de trayectoria simulado escenario obstáculo centro abierto

6.1.11. Escenario ADO

La pose de inicio fue (400 400 1.5708), la pose final fue (800 400 0) y la pose final de la cámara fue (889 727 155.74). La trayectoria seguida fue la siguiente:

0: 400 400 90 0 0

1: 513 354 133.446 50 20

2: 626 308 90 50 -20

3: 532 278 55.2434 -40 20

4: 682 444 117.23 40 0

5: 776 411 100.392 40 -10

6: 850 414 74.3246 30 -20

7: 915 449 48.2572 30 -20

8: 970 498 48.2579 30 0

Posterior la trayectoria realizada fue la mostrada en (Figura 218, Figura 219). La trayectoria realizada se gráfica encima ya que realiza cuatro árboles. El seguimiento de trayectoria simulado es el de la (Figura 220).

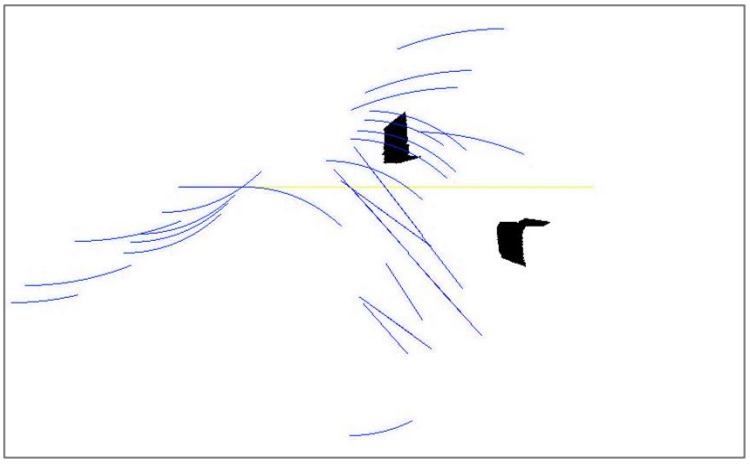


Figura 218. Trayectoria realizada para escenario doble obstáculo abierto parte 1

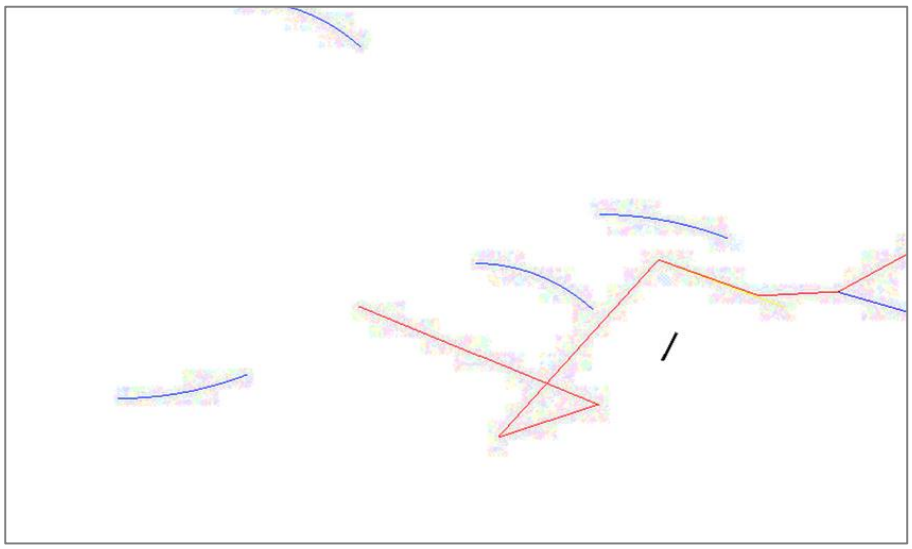


Figura 219. Trayectoria realizada para escenario doble obstáculo abierto parte 2

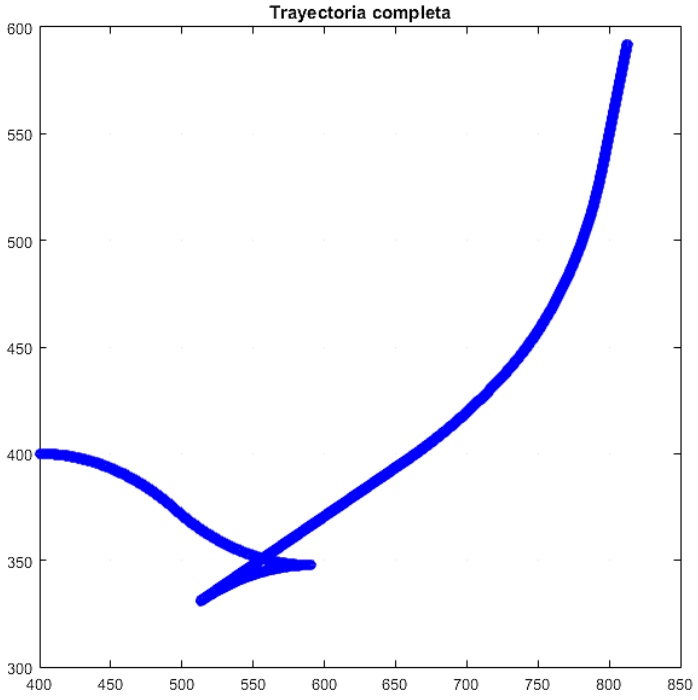


Figura 220. Seguimiento de trayectoria simulado escenario doble obstáculo abierto

6.1.12. Escenario AOM

La pose de inicio fue (400 400 1.5708), la pose final fue (800 400 0) y la pose final de la cámara fue (719 279 54.7925). La trayectoria seguida fue la siguiente:

0: 400 400 90 0 0

1: 513 354 133.446 50 20

2: 627 358 84.6459 30 0

3: 744 323 128.092 50 20

4: 805 217 171.537 50 20

5: 832 147 145.47 30 -20

6: 761 249 145.469 -50 0

Posterior la trayectoria simulada fue la mostrada en la (Figura 221). El seguimiento de trayectoria simulado es el de la (Figura 221).

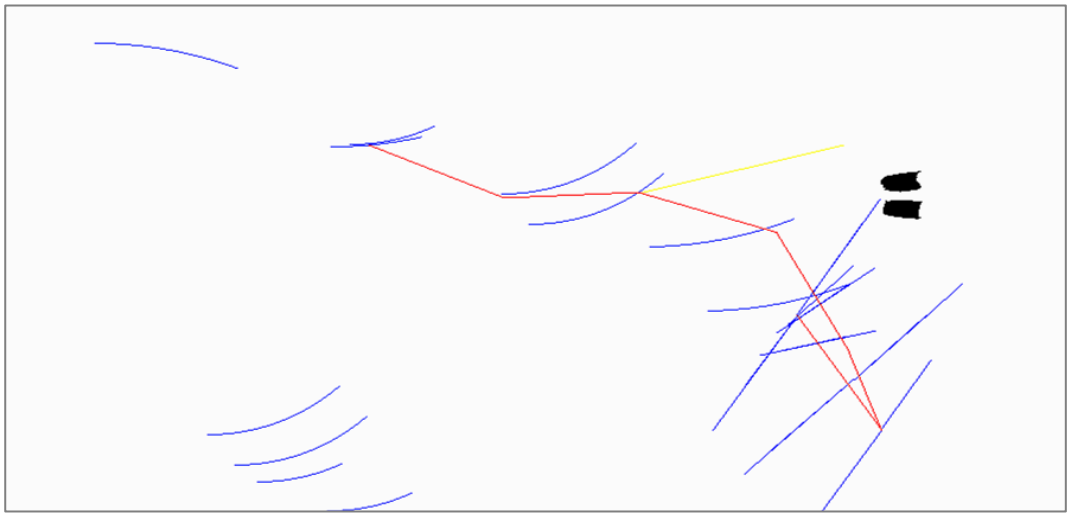


Figura 221. Trayectoria realizada para escenario onstáculo en movimiento abierto

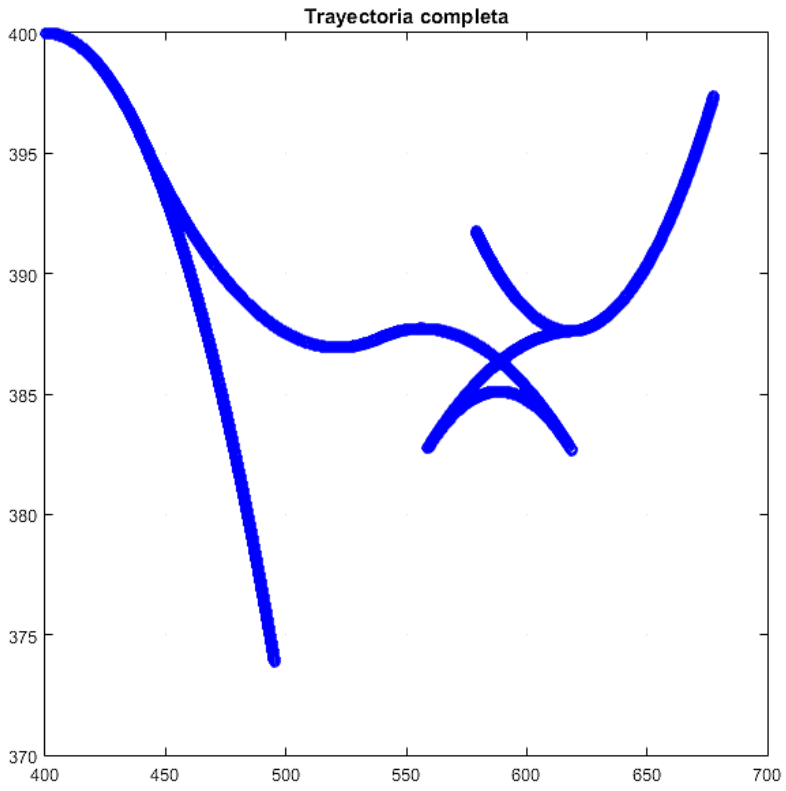


Figura 222. Seguimiento de trayectoria simulado escenario obstáculo en movimiento abierto

En este caso el objeto fue una persona moviéndose de derecha a izquierda y deteniéndose en la posición izquierda.

6.2. Análisis de resultados de la planificación en los escenarios

Como se señaló se ha tomado en cuenta varios criterios de comparación en cada escenario, por ello la (Tabla 27) presenta los resultados en los que se puede contrastar los escenarios. Y teniendo como principal observación que 11 de los 12 escenarios pudo alcanzar la meta, generando un porcentaje de asertividad del 91%.

Tabla 27
Criterios de comparación para cada escenario

Criterios	CSO	COD	COI	COC	CDO	COM	ASO	AOD	AOI	AOC	ADO	AOM
Convergencia de Trayectoria	SI	SI	SI	NO	SI	SI	SI	SI	SI	SI	SI	SI
Seguimiento de trayectoria	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI	NO	SI
Colisión ante obstáculos	NO	SI	NO	NO	NO	NO	NO	SI	NO	NO	SI	NO
Colisión ante obstáculos dinámicos	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
Recuperación de trayectoria	-	-	-	NO	-	SI	-	-	-	SI	-	SI
Cantidad de segmentos locales	2	10	5	-	17	9	4	7	5	12	8	6
Cantidad de árboles realizados.	1	1	1	-	4	2	1	2	1	2	3	2
Tiempo de ejecución	18	17	23	-	70	30	19	27	17	33	49	31

6.3. Profundidad medida por la cámara

Se evalúa las imágenes de profundidad mostradas por la cámara ante los parámetros preestablecidos en capítulos anteriores, así como también las mediciones y aproximaciones de distancia a objetos cercanos.

6.4. Mapeo 3D

En esta prueba se evalúa el desempeño de la cámara para obtener el mapa espacial en 3D, para los parámetros de obtención de datos preestablecidos en capítulos anteriores. Se evalúa al vehículo estático y luego de hacer una trayectoria en los diferentes modos de mapeo planificados para el vehículo.

Tabla 28

Parámetros definidos según el modo de funcionamiento

Parámetros	Modo NO tripulado	Modo tripulado
Resolución	1344x376(VGA)	2560x720 (HD720)
FPS	30	60
Calidad de percepción	PERFORMANCE	PERFORMANCE
Filtro de Malla	Bajo	Bajo
Visualización de Mapeado	NO	SI
Visualización de mapa de profundidad	SI	NO
Procesos en paralelo	SI	NO

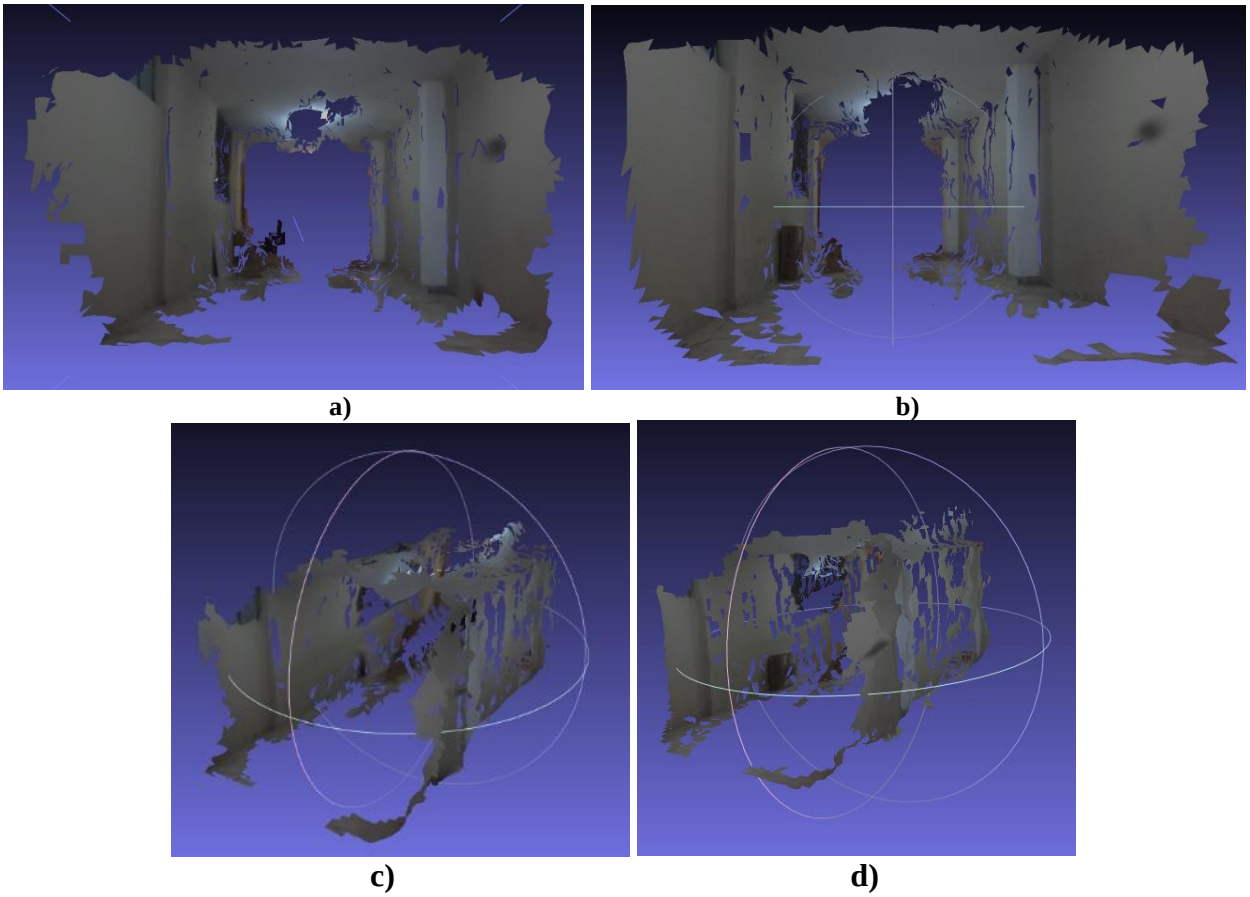


Figura 223. Resultado de mapeo en interior
a) y c) Modo no tripulado, b) y d) Modo tripulado.

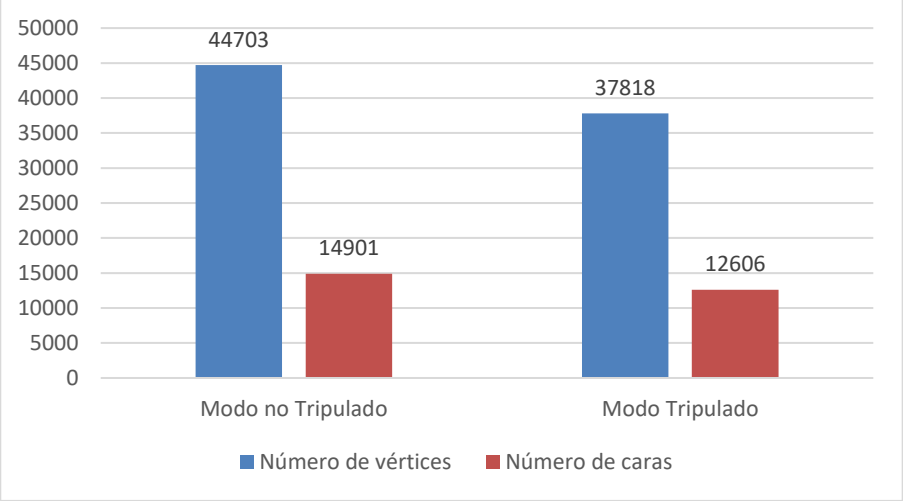


Figura 224. Numero de vértices creados por la malla de mapeo en un entorno interior sin movimiento del vehículo.

Considerando las condiciones en las que trabajan ambos modos, los resultados de los mapas no presentan mayores cambios y la percepción tridimensional del entorno es óptima. La creación de la malla presenta un numero de caras y vértices aproximados entre sí, pese a que en el modo no tripulado se tiene varios procesos en paralelo genera un buen mapa, y en el modo tripulado pese a contar con mayor definición de las texturas los vértices creados son menos que en el modo anterior pero esta diferencia no es significativa.

6.5. Pruebas en navegación

Con el fin de evaluar el sistema completo ante la evasión de obstáculos, se procede a realizar las respuestas de la percepción tridimensional ante un obstáculo luego de capturar todos los datos durante una navegación.

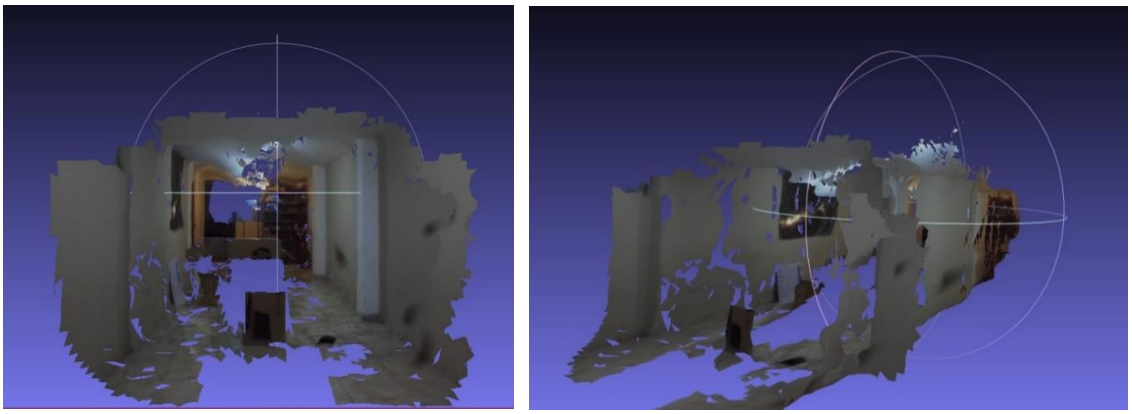


Figura 225. Resultado de mapeo en interior luego de la navegación.

6.6. Pruebas con relieves restrictivos

Esta prueba se lleva a cabo para conocer las limitaciones finales del proyecto conjunto, una vez que se ha unido todos los sensores y dispositivos que van montados en el mismo se procede a evaluar los casos en los cuales no podría funcionar, por ejemplo, obstáculos a una altura específica, o de formas especiales como una banca o de materiales difíciles de percibir como el vidrio.

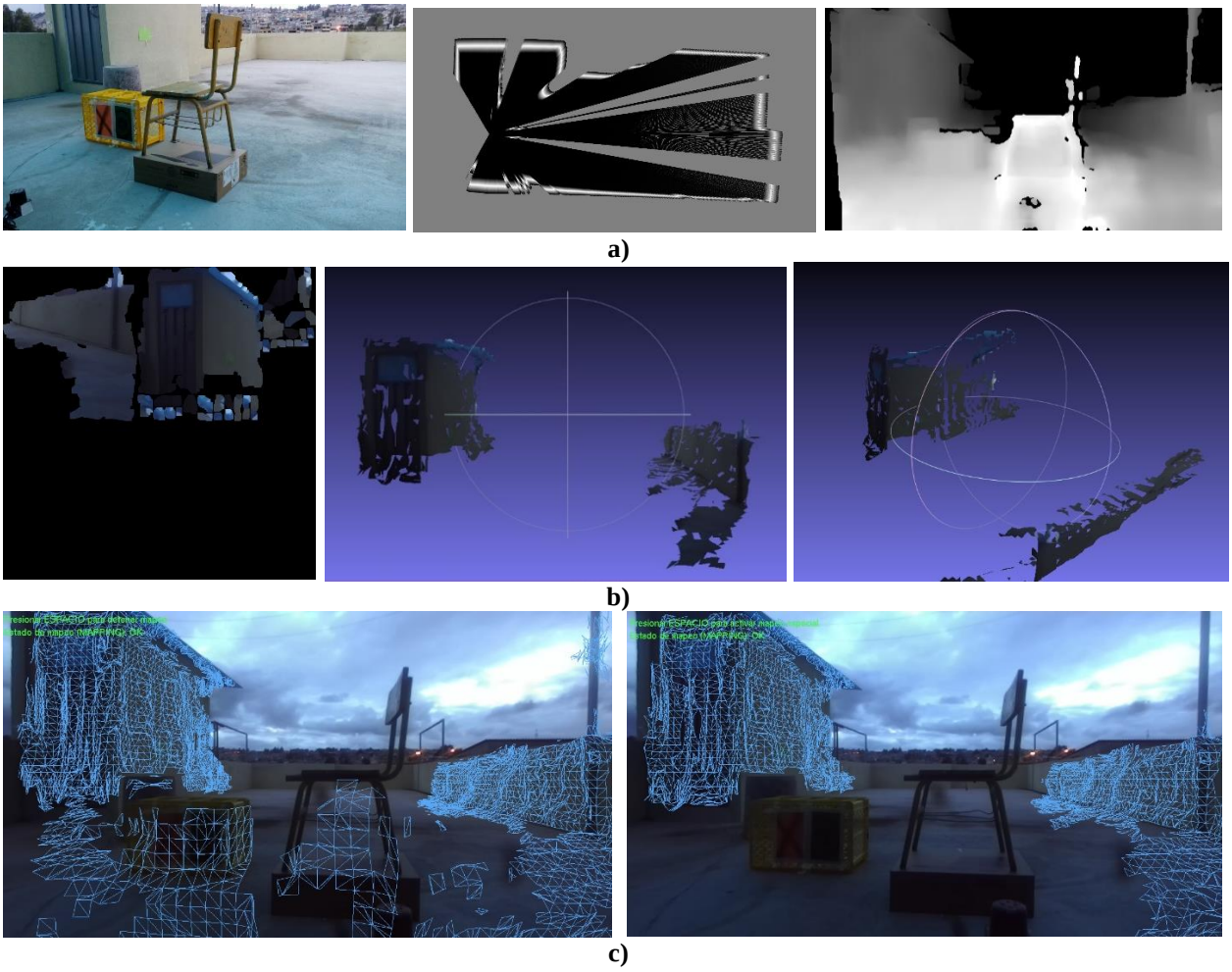
Las restricciones vienen determinadas por las capacidades de percepción de los sensores laser y la cámara estereoscópica, la cámara no puede percibir ciertos obstáculos como el vidrio, ya que es transparente y en la imagen no se notan sus puntos de interés, y en el caso del Lidiar, al ser un sensor en 2 dimensiones solo puede tomar un plano de referencia y los objetos que se encuentran por encima de este plano no serán detectados.

Los relieves restrictivos, pueden ser superados si se realiza un trabajo cooperativo entre dos sensores, para este trabajo se evalúan los casos en los que son restrictivos para cada sensor individualmente, para establecer los parámetros de funcionamiento, los obstáculos o relieves restrictivos que afectan directamente a este trabajo son los siguientes:

- Obstáculos superiores a la línea base de mapeo del sensor LIDAR

- Obstáculos con múltiples apoyos como sillas, mesas, cercas, etc., para el sensor LIDAR
- Obstáculos translucidos como vidrios, plásticos, etc., para la CÁMARA
- Obstáculos que se encuentren en lugares con poca luz, para la CÁMARA

Tomando en cuenta estos parámetros mencionados anteriormente se presentan las diferentes pruebas, primero se coloca un obstáculo con formas diferentes en el eje vertical, mostrados a continuación en la (Figura 226).



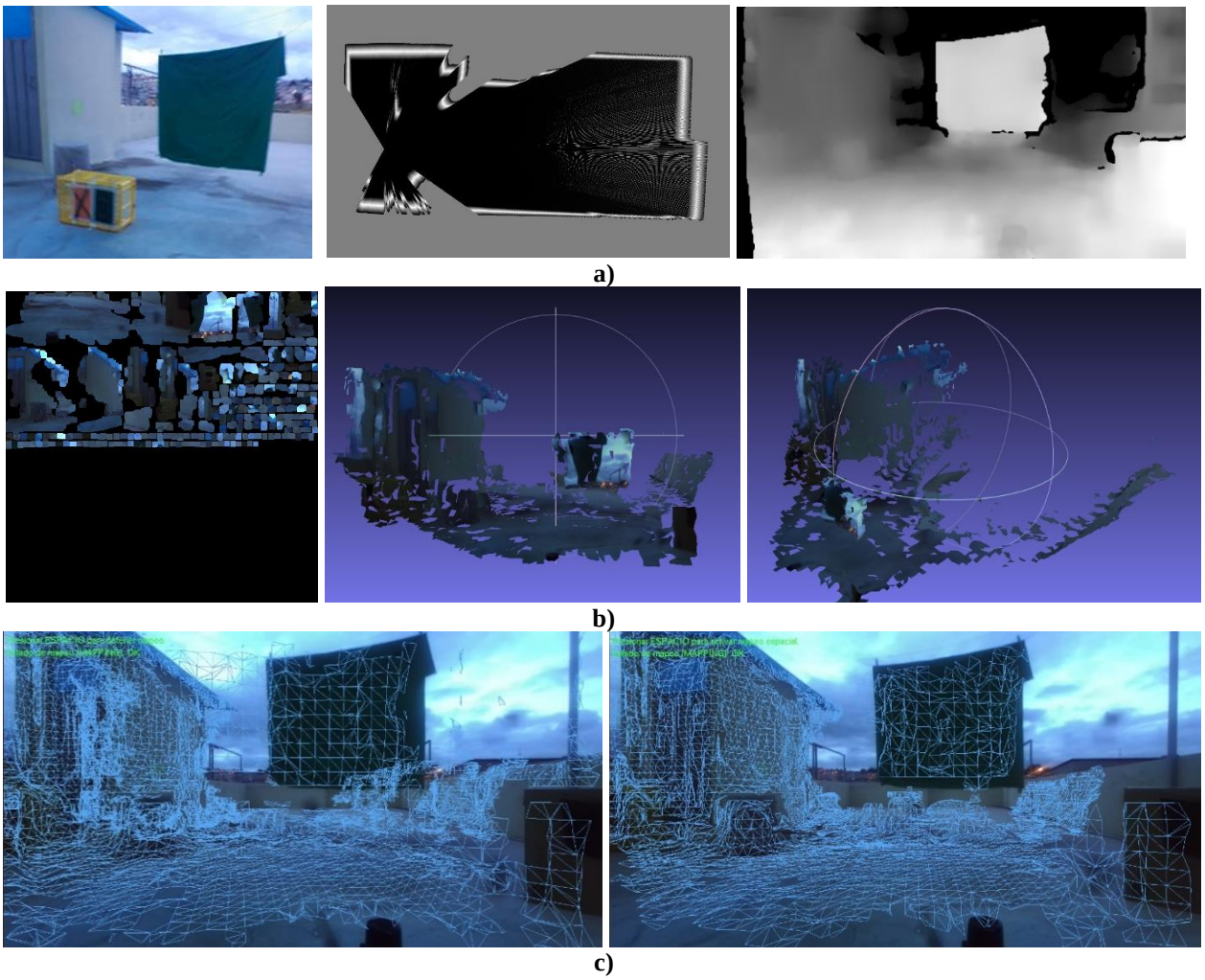
a)

b)

c)

Figura 226. Resultados para prueba en entorno restrictivo usando obstáculos con formas diferentes. a) Resultado de escaneo simple para navegación (SLAM) b) Resultado de mapeo espacial, c) Resultado de percepción 3D en modo tripulado

La siguiente prueba a realizarse, es un obstáculo que supere completamente la altura a la que trabaja el sensor lidar, los resultados de esta prueba se muestran en la (Figura 227).



a)

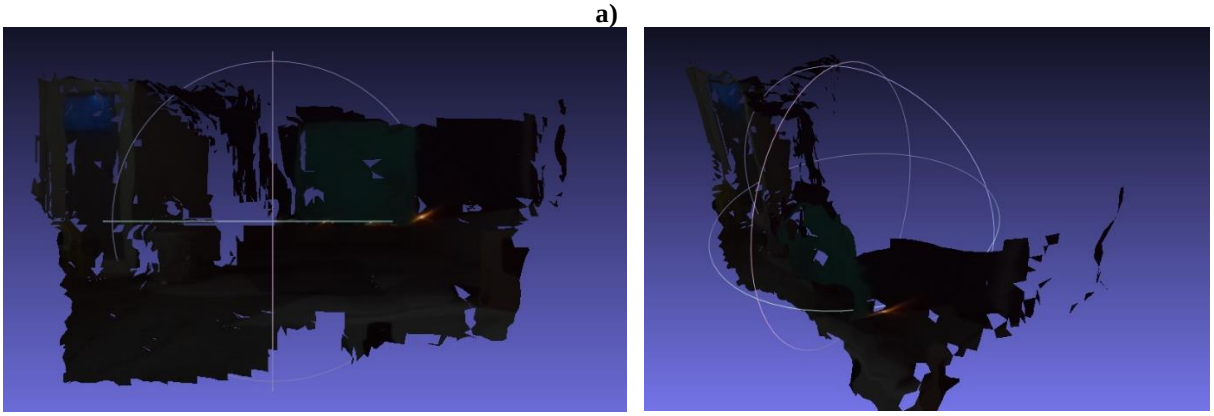
b)

c)

Figura 227. Resultados para prueba en entorno restrictivo usando obstáculos con altura superior a la del lidar. a) Resultado de escaneo simple para navegación (SLAM) b) Resultado de mapeo espacial, c) Resultado de percepción 3D en modo tripulado

La siguiente prueba que realizarse es similar a la anterior, pero ahora se evalúa los resultados en la noche, los resultados se muestran en la (Figura 228).

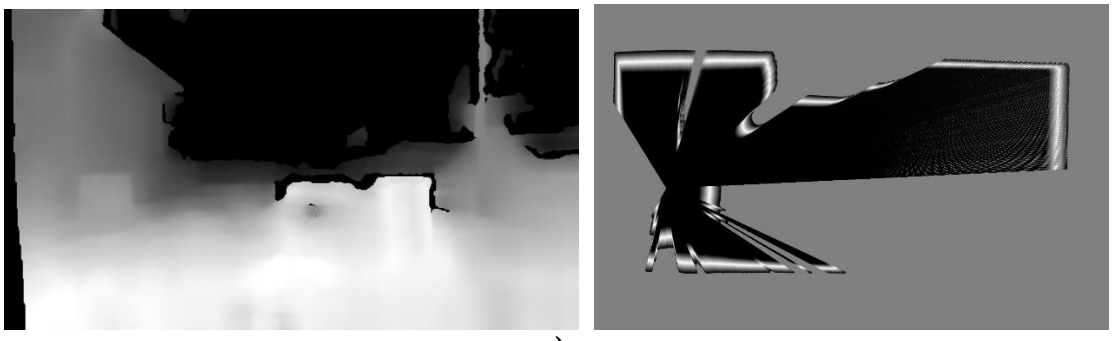
a)



b)

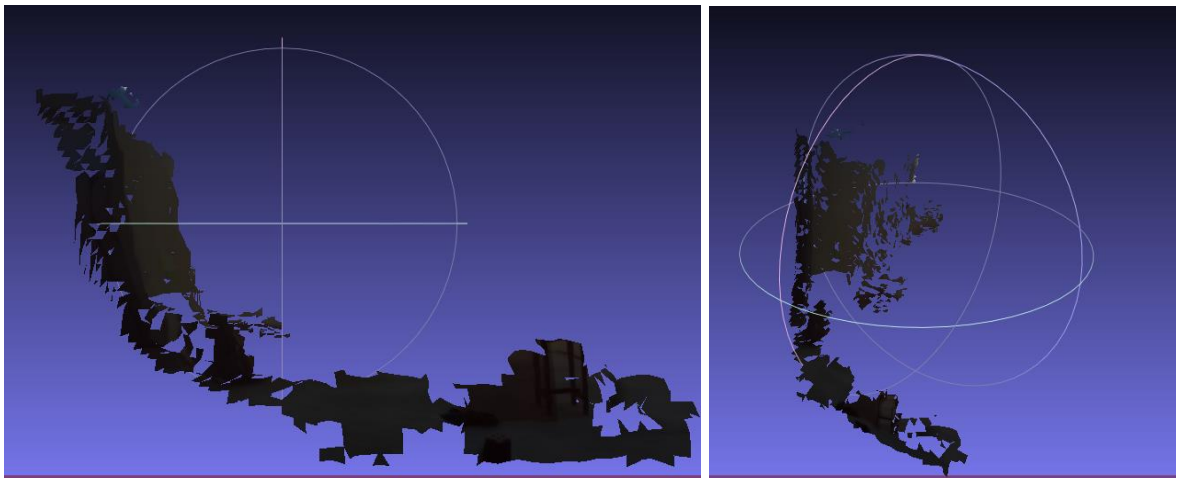
Figura 228. Resultados para prueba en entorno restrictivo usando obstáculos en la noche a) Percepción de cámara en la noche, b) Resultado de mapeo espacial en la noche.

Finalmente se realiza la prueba en la cual se coloca como obstáculo un vidrio transparente, los resultados de esta prueba se muestran a continuación en la (Figura 229).

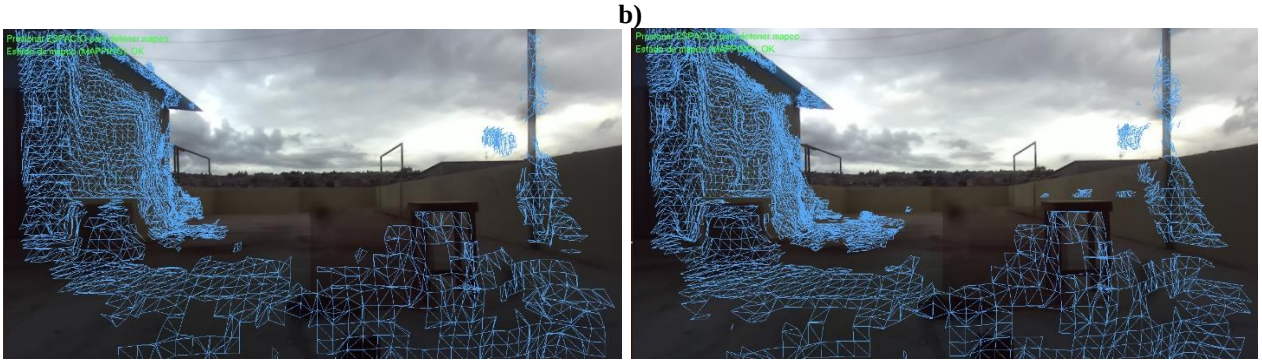


a)

CONTINÚA ⇨



b)



c)

Figura 229. Resultados para prueba en entorno restrictivo usando obstáculos transparentes
a) Resultado de escaneo simple para navegación (SLAM)
b) Resultado de mapeo espacial, c) Resultado de percepción 3D en modo tripulado

Como resultados, se tiene que: para la prueba de obstáculos con formas diferentes, en la línea de medición del LIDAR únicamente puede captar los apoyos de la silla, lo cual engaña al sensor y en caso de que la apertura de los apoyos sea suficiente para que el vehículo avance este lo hará, pero chocará debido a los obstáculos puestos debajo de la silla y con el espaldar y asiento de esta. Sin embargo, la cámara es capaz de detectar este obstáculo, tanto en la parte inferior como superior.

En el caso de los obstáculos con una altura superior al LIDAR, este logra engañar completamente al sensor, ya que este visualiza un campo abierto, sin embargo, la cámara logra captar muy bien este obstáculo.

En la prueba en la noche, el sensor LIDAR funciona perfectamente como siempre lo hace, por lo cual no tiene mucha relevancia la oscuridad para este sensor, sin embargo la capara si es engañada por el efecto de la oscuridad y la luz artificial, y se muestra todo como si fuese una pared o una barrera, por lo cual en la noche o en entornos oscuros para obstáculos que superen la altura del LIDAR, o que puedan no se podrá efectuar la navegación.

Finalmente, para obstáculos transparentes, como el vidrio utilizado, al sensor LIDAR no lo logra engañar y se puede evitar este obstáculo, sin embargo, a la cámara lo engaña por completo y no logra reconocer este obstáculo, incluso este obstáculo es difícil de reconocer para la visión humana. El vehículo será capaz de superar este obstáculo únicamente si este no supera la altura de medición del LIDAR.

CAPITULO VII

7. CONCLUSIONES Y RECOMENDACIONES

7.1. Conclusiones

El desarrollo de un sistema de navegación autónoma y percepción visual se realizó aplicando la metodología sugerida en el estudio del estado del arte. Implica definir y caracterizar cada parte del sistema. Se caracterizó el sensor LIDAR HOKUYO UST-20LX y la cámara de estéreo visión ZED, para lo cual se redefinieron varias características expresadas por sus fabricantes. En ciertas características se verificó y confirmó los valores preestablecidos, sin embargo, otros parámetros como rango de detección, precisión y exactitud son específicos para cada dispositivo y la forma en la que se los emplea. Pruebas como la respuesta al color y del ángulo de inclinación de las superficies, definen el entorno. Es así que se resuelve que en este sistema el sensor responde mejor ante superficies rectas y en tonos de color amarrillo.

El Sistema empieza a completarse cuando se desarrolla el algoritmo de planificación de trayectorias que junto con otros aspectos permitió la navegación del UGV en entornos desconocidos. Se determinó que un planificador no puede ser desarrollado sin antes definir el modelo cinemático del vehículo, considerando la geometría, el modo de locomoción, la respuesta que posee el motor y como se comporte el vehículo ante acciones de control eléctricas. Conjuntamente por medio del sensor laser se identificó el entorno, definiendo mapas iniciales y actualizándolos en el momento que se lo requiera. Los mapas de navegación en 2D se generaron con un algoritmo eficiente y que no generó mayor gasto computacional al momento de ejecutar directamente el planificador.

El algoritmo de planificación se enfocó en tres aspectos; entornos desconocidos, tiempo real y restricciones del vehículo. Fue necesario desarrollar el planificador por medio del algoritmo base RRT, dividiendo al planificador en dos; local y global. Para el algoritmo local se determinó que un vehículo Ackerman genera trayectorias circulares uniformes que dependen del tiempo, seguidamente fue necesario ajustar el modelo calculado a las condiciones reales generadas por restricciones dinámicas como rozamiento, energía suministrada por baterías y dimensiones del entorno. Las modificaciones realizadas en el algoritmo global RRT se enfocaron en modificar las restricciones ante las condiciones del UGV. Es así que, por medio de funciones como; función de riesgo, función de pose aleatoria real, mejor peso y mejor puntaje de control, se optimizo la trayectoria calculada para el vehículo y se desarrolló el algoritmo llamado Non-Holonómic Risk RRT.

La optimización de la trayectoria se la realizó por medio de calibración de parámetros influyentes como son; mapa, pose, profundidad del árbol, umbral de riesgo, probabilidad de meta real y paso a meta final. Una función de ponderación de pesos en las que variables como; tiempo de ejecución, errores de la pose, cantidad de segmentos locales e iteraciones que realizó el programa, tuvieron pesos diferentes permitiendo una navegación efectiva. La navegación se complementó con algoritmos como; control de giro, control de velocidad, percepción tridimensional, navegación segura y verificación de trayectoria. Es así que el sistema responde ante un entorno controlado en el cual si no existen obstáculos que pueda sortear no se mueve y en cada trayectoria realizada verifica su posición para estimar si se alcanzó o no la meta.

La obtención del mapeado espacial se llevó a cabo basándose en pruebas de caracterización para determinar los parámetros necesarios para satisfacer los objetivos planteados, entre los parámetros modificados se tiene: la calidad de percepción de la profundidad, la resolución de

imagen obtenida por la cámara, la resolución de malla para el mapeado y la distancia de percepción. Los parámetros finales se determinaron ponderando la calidad de los objetos graficados y el tiempo de respuesta, siendo este último un factor determinante para el trabajo. Debido a la baja capacidad de la tarjeta gráfica se limitó la capacidad de procesar imágenes y también la capacidad del computador afecto significativamente en los tiempos de cálculo para la triangulación y obtención de los datos espaciales, por lo cual finalmente se uso mallas gruesas con baja definición para mapear en tiempo real, especialmente en el modo no tripulado ya que se ejecuta el mapeado en paralelo con los algoritmos de planificación de trayectorias.

Los resultados de la percepción 3D difieren entre si debido a las características propias de cada parte del trabajo y a la variedad de aplicaciones que se otorga al sistema. Se percibe la profundidad en el plano de visión de la cámara, con este dato se obtiene la distancia promedio de un objeto hacia el centro de la cámara lo que permite evitar colisiones con objetos cercanos al vehículo. En la aplicación de mapeo espacial estos objetos pese a ser percibidos no son mapeados debido a que la malla en algunas ocasiones no logra completar triangulaciones con las otras partes del entorno y finalmente se eliminan. En entornos interiores el mapeo puede mostrar las paredes, techo y suelo del entorno dando como resultado un mapa muy detallado y completo. En entornos exteriores debido a la falta de paredes y techo mapea solo a los obstáculos predominantes en la imágenes capturadas y parte del suelo. En la noche la percepción de la profundidad falla, el resultado final tanto de la percepción de la profundidad como el mapa 3D y el rastreo posicional son erróneos, debido a que se toma al fondo capturado como una pared, por lo cual siempre presenta una profundidad y genera una pared inexistente.

Si bien el sistema posee una trayectoria optimizada y la percepción 3D es almacenada fue necesario identificar las limitaciones del sistema y respuesta ante eventos específicos. El uso de

obstáculos predefinido que sea apreciables para el sistema el sistema los evade sin dificultad alguna, el momento de que se colocan obstáculos delgados, transparentes o colgantes, dificultan el desempeño de la trayectoria. Al momento de limitar el espacio de trabajo se observó que el algoritmo puede fallar y generar colisiones, esto se debe a que el vehículo al tener dimensiones considerables es necesario acotar que el espacio debe ser suficiente para generar trayectorias integras. Así también, el entorno al ser exterior o interior diferencia su funcionamiento, esto se debe a que la cámara de visión depende más de la iluminación directa, una buena iluminación permite un buen desempeño sea un entorno cerrado o no.

7.2. Recomendaciones

Se recomienda utilizar sensores que simplifiquen el gasto computacional y los recursos necesarios para el funcionamiento. Si bien el sistema presentado cumple la expectativa, es necesario; utilizar un sensor laser económico, una cámara que no requiera de GPU y un vehículo con menos restricciones dinámicas. Un vehículo grande, pero con mayor ángulo de giro, tracción delantera, suspensión y ruedas de goma, permitiría que el sistema pueda desempeñarse en sistemas más restrictivos.

En el desarrollo del algoritmo una limitante fue que no se pudo realizar procesos en paralelo de forma completa, en la que se pudiese hacer uso de variables como la posición en cada instante de tiempo, para lo cual se podría utilizar lenguajes de programación que permiten dicho acometido como es con ROS. Adicionalmente se sugiere que se desarrolle una interfaz que permita mayor interacción con el usuario en el que se muestre posiblemente el movimiento dentro de la trayectoria en tiempo real, así verificar que el sistema responde ante entornos dinámicos.

Una aplicación que debe de ser desarrollada con esta base es la de self driving car, vehículos que puedan inclusive hacer estacionamiento autónomo, navegación en situaciones peligrosas con objetivos específicos y con el uso de herramientas a bordo como brazo o manipuladores robóticos.

Para el desarrollo de sistemas de percepción espacial basados en cámaras de visión estéreo se recomienda contar con más cámaras para tener más de un plano de visión o un sistema que genere un movimiento de rotación de la cámara para obtener un mapa completo del entorno. Adicional re sugiere contar con equipos de procesamiento capaces de soportar la obtención de imágenes en alta resolución y de esta manera obtener un mapa espacial de calidad y con alta resolución en las texturas.

Para trabajos futuros se recomienda combinar los resultados de los sensores utilizados para generar nuevas aplicaciones, por ejemplo, usar los datos del mapeo 3D sobre el plano generado por el sensor laser y obtener curvas de nivel simulando el funcionamiento de un sensor laser en tres dimensiones esta alternativa sería muy útil considerando el elevado costo de este tipo de sensores.

8. BIBLIOGRAFÍA

A. Geiger, P. L. (2012). *Are we ready for autonomous driving the KITTI vision benchmark suite.* IEEE Conf. on Computer Vision and Pattern Recognition (CVPR).

Acroname. (2014). *Scanning Laser Range Finder Smart-URG mini UST-20LX (UUST004) Specification.*

Aguilar, W. G., & Angulo, C. (2014). *Real-time video stabilization without phantom movements for micro aerial vehicles.* EURASIP Journal on Image and Video Processing, 1, 1-13.

Aguilar, W. G., & Angulo, C. (2014). *Robust video stabilization based on motion intention for low-cost micro aerial vehicles.* 11th International Multi-Conference on Systems, Signals & Devices (SSD). Barcelona, Spain.

Aguilar, W. G., & Angulo, C. (2016). *Real-Time Model-Based Video Stabilization for Microaerial Vehicles*. Neural Processing Letters, *43*(2), 459-477.

Aguilar, W. G., & Morales, S. (2016). *3D Environment Mapping Using the Kinect V2 and Path Planning Based on RRT Algorithms*. Electronics, *5*(4), 70.

Aguilar, W. G., Angulo, C., & Costa-Castello, R. (2017). *Autonomous Navigation Control for Quadrotors in Trajectories Tracking.* En Lecture Notes in Computer Science (págs. 287-297).

Aguilar, W. G., Casaliglla, V. P., Pólit, J. L., Abad, V., & Ruiz, H. (2017). *Obstacle Avoidance for Flight Safety on Unmanned Aerial Vehicles*. En Lecture Notes in Computer Science (págs. 575-584).

Aguilar, W. G., Cobeña, B., Rodriguez, G., Salcedo, V. S., & Collaguazo, B. (2018). *SVM and RGB-D Sensor Based Gesture Recognition for UAV Control.* International Conference on Augmented Reality, Virtual Reality and Computer Graphics (págs. 713-719). Springer.

Aguilar, W. G., Luna, M. A., Moya, J. F., Abad, V., Ruiz, H., Parra, H., & Angulo, C. (2017). *Pedestrian Detection for UAVs Using Cascade Classifiers and Saliency Maps.* En Lecture Notes in Computer Science (págs. 563-574).

Aguilar, W. G., Luna, M. A., Moya, J. F., Abad, V., Ruiz, H., Parra, H., & Lopez, W. (2017). *Cascade Classifiers and Saliency Maps Based People Detection.* En Lecture Notes in Computer Science (págs. 501-510).

Aguilar, W. G., Luna, M. A., Moya, J. F., Luna, M. P., Abad, V., Ruiz, H., & Parra, H. (2017). *Real-Time Detection and Simulation of Abnormal Crowd Behavior.* En Lecture Notes in Computer Science (págs. 420-428).

Aguilar, W. G., Manosalvas, J. F., Guillén, J. A., & Collaguazo, B. (2018). *Robust Motion Estimation Based on Multiple Monocular Camera for Indoor Autonomous Navigation of Micro Aerial Vehicle.* International Conference on Augmented Reality, Virtual Reality and Computer Graphics (págs. 547-561). Springer.

Aguilar, W. G., Morales, S., Ruiz, H., & Abad, V. (2017). *RRT* GL Based Optimal Path Planning for Real-Time Navigation of UAVs.* En Lecture Notes in Computer Science (págs. 585-595).

Aguilar, W. G., Morales, S., Ruiz, H., & Abad, V. (2017). *RRT* GL Based Path Planning for Virtual Aerial Navigation.* En Lecture Notes in Computer Science (págs. 176-184).

Aguilar, W. G., Rodríguez, G. A., Álvarez, L., Sandoval, D., Quisaguano, F., & Limaico, A. (2017). *Visual SLAM with a RGB-D Camera on a Quadrotor UAV Using on-Board Processing.* 596-606.

Aguilar, W. G., Rodríguez, G. A., Álvarez, L., Sandoval, S., Quisaguano, F., & Limaico, A. (2017). *On-Board Visual SLAM on a UGV Using a RGB-D Camera.* En Lecture Notes in Computer Science (págs. 298-308).

Aguilar, W. G., Rodríguez, G. A., Álvarez, L., Sandoval, S., Quisaguano, F., & Limaico, A. (2017). *Real-Time 3D Modeling with a RGB-D Camera and On-Board Processing.* En Lecture Notes in Computer Science (págs. 410-419).

Aguilar, W. G., Salcedo, V. S., Sandoval, D. S., & Cobeña, B. (2017). *Developing of a Video-Based Model for UAV Autonomous Navigation.* En Communications in Computer and Information Science (págs. 94-105).

Aguilar, W. G., Verónica, C., & José, P. (2017). *Obstacle Avoidance Based-Visual Navigation for Micro Aerial Vehicles*. Electronics, *6*(1), 10.

Aguilar, W. G., Verónica, C., & José, P. (2017). *Obstacle Avoidance for Low-Cost UAVs*. IEEE 11th International Conference on Semantic Computing (ICSC). San Diego.

Akhlaq, M. U., Izhar, U., & Shahbaz, U. (2014). *Depth estimation from a single camera image using power fit.* Robotics and Emerging Allied Technologies in Engineering (iCREATE), 2014 International Conference, 221-227.

AliExpress. (2018). *R550 10000 rpm DC 12 V 2a alta velocidad micro eléctrico Juguetes motor reversible ajustable la nave modelo Juguetes niños coche cortador de hierba*. Obtenido de https://es.aliexpress.com/store/product/10000rpm-DC-12V-15W-3N-cm-R550-2A-High-speed-Miniature-Electric-toys-motor-Long-output

Amaguaña, F., Collaguazo, B., Tituaña, J., & Aguilar, W. G. (2018). *Simulation System Based on Augmented Reality for Optimization of Training Tactics on Military Operations*. International Conference on Augmented Reality, Virtual Reality and Computer Graphics (págs. 394-403). Springer.

Amazon. (2018). *FY 6V 27mhz Control Box Receiver Match Remote Control, Transmitter Accessory Receive Controller Signal for Kids Powered Ride On Car Children's Ride on Toys Electric Replacement Parts*. Obtenido de https://www.amazon.com/Transmitter-Accessory-Controller-Childrens-Replacement/dp/B0746G51D9

Andrea, C. C., Byron, J. Q., Jorge, P. I., Inti, T. C., & Aguilar, W. G. (2018). *Geolocation and Counting of People with Aerial Thermal Imaging for Rescue Purposes*. International Conference on Augmented Reality, Virtual Reality and Computer Graphics (págs. 171-182). Springer.

Arduino. (2018). *Arduino MEGA 2560*. Obtenido de https://store.arduino.cc/usa/arduino-mega-2560-rev3

Arduino. (2018). *MPU-6050 Accelerometer + Gyro*. Obtenido de https://playground.arduino.cc/Main/MPU-6050

Aspire. (2018). *Especificaciones Tecnicas 4752 Series.*

Aurenhammer, F. (1991). *Voronoi diagrams—a survey of a fundamental geometric data structure.* ACM Computing Surveys (CSUR), 23(3), 345-405.

Baig, Q. &. (2011). *Fusion Between Laser and Stereo Vision Data For Moving Objects Tracking In Intersection Like Scenario.*

Baldwin, I. (2012). *Road vehicle localization with 2D push-broom LIDAR and 3D priors.* International Conference on Robotics and Automation.

Baltsavias, E. P. (1999). *Airborne laser scanning: basic relations and formulas.* ISPRS Journal of photogrammetry and remote sensing, 54(2-3), 199-214.

Baltzakis, H. A. (2003). *Fusion of laser and visual data for robot motion planning and collision avoidance.* Machine Vision and Applications.

Bambino, I. (2008). *Una introducción a los robots móviles.*

Bhat, M. S. (2015). *Embedded System based waiter and military robot path planning.* Control, Instrumentation, Communication and Computational Technologies.

Board, N. S., & Council, N. R. (2005). *Autonomous vehicles in support of naval operations.* National Academies Press.

Bonnick, A. (2008). *Automotive science and mathematics*. . Routledge.

Bouraine, A. S. (2016). *Real-time Safe Path Planning for Robot Navigation in Unknown Dynamic Environments*. Computing Systems and Applications.

Breiteneder, M. B. (2013). *Stereo Matching - State-of-the-Art and Research Challenges.* Advanced Topics in Computer Vision. Springer.

Campbell, J. R., Hlavka, D. L., Welton, E. J., Flynn, C. J., Turner, D. D., Spinhirne, J. D., & Hwang, I. H. (2002). *Full-time, eye-safe cloud and aerosol lidar observation at atmospheric radiation measurement program sites: Instruments and data processing.* Journal of Atmospheric and Oceanic Technology, 19(4), 431-442.

Cano Olivera, C. (2010). *Telémetro láser 3D con sensor Hokuyo UTM-30LX.* (Master's thesis).

Casanova, A. A. (s.f.). Pscicología de la percepción visual. Ph DrVision & Control of Action (VISCA) .

Chakraborty, S., & Basu, S. (2010). *Design of a smart Unmanned ground vehicle for Hazardous environments*. arXiv preprint arXiv: 1002.4180.

Chapman, S. (2014). *Máquinas Eléctricas.* Bogotá: McGrawHill.

Chen, Y., & Medioni, G. (1992). *Object modelling by registration of multiple range images.* Image and vision computing, 10(3), 145-155.

Choi, M. W., Park, J. S., Lee, B. S., & Lee, M. H. (2008). *The performance of independent wheels steering vehicle (4WS) applied Ackerman geometry*. Control, Automation and Systems, 2008. ICCAS 2008. International Conference, 197-202.

Chong, Z. J. (2013). *Synthetic 2D LIDAR for Precise Vehicle Localization in 3D Urban.* International Conference on Robotics and Automation.

Collis, R. T., & Ligda, M. G. (1964). *Laser radar echoes from the clear atmosphere*. Nature, 203(4944), 508.

Correa, A. C. (2005). *Sistemas robóticos teleoperados.* . Ciencia e Ingeniería Neogranadina, (15), 62-69.

Correa, J. (2010). *Modelado y smulación dinamica de vehículos de competición.*

Costella, V., & Rodríguez, H. (2017). *Caracterización de un sensor de rango láser de bajo costo previo a una implementación de Slam.* Prisma Tecnológico, 7(1), 30-34.

Cousins, R. a. (2011). *3D is here: Point Cloud Library (PCL)*. Robotics and Automation (ICRA).

Crossquads. (2018). *Kinder Elektro MRT Concept*. Obtenido de http://www.cross-quad.de/de/kinder-elektro-auto-mrt-concept-2x30w-12v-rc-mp3.html

Currie, D., Dell'Agnello, S., & Monache, G. D. (2011). *A Lunar Laser Ranging Retroreflector Array for the 21st Century*. Acta Astronautica, 68(7-8), 667–680.

De Mata, J. A. (2016). *Calibración geométrica de cámaras no métricas. Estudio de metodologías y modelos matemáticos de distorsión.* (Doctoral dissertation, Universidad Politécnica de Madrid).

Deakin, R. E. (2016). EDM: *Notes on Electronic Distance Measurement.*

Dietmayer, S. W. (2008). *3D vehicle detection using a laser 3D vehicle detection using a laser*. IET Intelligent Transport Systems.

Dubins, L. E. (1957). *On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents.* American Journal of mathematics, 79(3), 497-516.

Duchoň, F. F. (2014). *Path Planning with Modified a Star Algorithm for a Mobile Robot.* Procedia Engineering, 59-69.

Engelhard, N. &. (2011). *Real-time 3D visual SLAM with a hand-held RGB-D camera*. Proc. of the RGB-D Workshop on 3D Perception in Robotics.

FirstPower. (2018). *Standard Battery Series (FP, LFP CFP TYPES)*. Obtenido de http://www.efirstpower.com/Products_Standard_Battery.html

Fong, E. H., Adams, W., Crabbe, F. L., & Schultz, A. C. (2003). *Representing a 3-d environment with a 2 1/2-d map structure*. Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference , Vol. 3, pp. 2986-2991.

Frintrop, S. (2015). *Bio-inspired Vision Systems.*

Fulgenzi, C., Spalanzani, A., Laugier, C., & Tay, C. (2010). *Risk based motion planning and navigation in uncertain dynamic environment*. HAL , 14.

Gage, D. W. (1995). UGV history 101: *A brief history of Unmanned Ground Vehicle (UGV) development efforts.* Aval Comand Control and ocean survilance Center San Diego.

GeekFactory. (2018). *Monster Motor Shield*. Obtenido de https://www.geekfactory.mx/tienda/motores-y-controladores/monster-motor-shield/

Gehrig, S. K., & Stein, F. J. (2007). *Collision avoidance for vehicle-following systems.* IEEE transactions on intelligent transportation systems, 8(2), 233-244.

Gim, S., Adouane, L., Lee, S., & Dérutin, J. P. (2017). *Clothoids composition method for smooth path generation of car-like vehicle navigation.* Journal of Intelligent & Robotic System, 88(1), 129-146.

Github. (2018). *Stereolabs ZED - Python Integration (beta).* Obtenido de https://github.com/stereolabs/zed-python

Gómez Cardenas, J. J. (2007). *Cinemática relativista.*

Gonzalez, C. I., Melin, P., Castro, J. R., & Castillo, O. (2017). Edge *Detection Methods and Filters Used on Digital Image Processing.* Edge Detection Methods Based on Generalized Type-2 Fuzzy Logic, 11-16.

Gonzalez, R. C., & Woods, R. E. (2002). *Digital image processing*.

Green, C. G., Massatt, P. D., & Rhodus, N. W. (1989). *The GPS 21 primary satellite constellation*. Navigation, 36(1), 9-24.

Gupta, A., Divekar, R., & Agrawal, M. (2010). *Autonomous parallel parking system for Ackerman steering four wheelers*. Computational Intelligence and Computing Research (ICCIC), 2010 IEEE International Conference, 1-6.

Gutmann, J.-S. F. (2012). *3D Perception and Environment Map Generation for Humanoid Robot Navigation*. Intelligent Systems Research Laboratory.

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

Hernandez Beleño, R. D., Bernardes Vítor, G., Vaqueiro Ferreira, J., & Siqueira Meirelles, P. (2014). *Planeación y Seguimiento de Trayectorias de un Vehículo Terrestre con Base en el Control de Dirección en un Ambiente Real*. Scientia et technica, 19(4).

Hernández, J. M., Sanz, G. P., & Guijarro, M. (2011). *Técnicas de procesamiento de imágenes estereoscópicas*. Revista del CES Felipe II.

Ho, Y.-K. J.-S. (1999). *Traffic Parameter Extraction Using Video-based Vehicle Tracking.* International Conference on Intelligent Transportation Systems, 764-769.

Hong, L., & Chen, G. (2004). *Segment-based stereo matching using graph cuts*. *Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Proceedings of the 2004 IEEE Computer Society Conference, Vol. 1, pp. I-I.

Huang, L. (2009). *A Novel Multi-Planar LIDAR and Computer Vision Calibration Procedure Using 2D Patterns for Automated Navigation.* IEEE Intelligent Vehicles Symposium.

Hwang, J. P. (2007). *Multi-Classifier Based LIDAR and Camera Fusion*. Intelligent Transportation Systems Conference.

Instructables. (2018). *Tutorial for Motor Shield VNH2SP30*. Obtenido de http://www.instructables.com/id/Monster-Motor-Shield-VNH2SP30/

Islam, F. (2012). *RRT*-Smart: Rapid convergence implementation of RRT* towards optimal solution.* Proceedings of IEEE International Conference on Mechatronics and Automation.

Jaklin, G. N. (2013). *Real-time path planning in heterogeneous environments.* Computer Animation & virtual worlds.

James, L. &. (2000). RRT-Connect : *An Efficient Approach to Single-Query Path Planning.* International Conference on Robotics & Automation.

Janes, A., Sillitti, A., & Succi, G. (2012). *Using the Eclipse C/C++ Development Tooling as a Robust, Fully Functional, Actively Maintained, Open Source C++ Parser.* Forking the Commons: Developmental Tensions and Evolutionary Patterns in Open Source Software.

Jara-Olmedo, A., Medina-Pazmiño, W., Mesías, R., Araujo-Villaroel, B., Aguilar, W. G., & Pardo, J. A. (2018). *Interface of Optimal Electro-Optical/Infrared for Unmanned Aerial Vehicles.* En Smart Innovation, Systems and Technologies (págs. 372-380).

Jeffrey, S., PhD, K., ABPP, FACRM, John DeLuca PhD, A., & Bruce Caplan PhD, A. (. (2011). *Encyclopedia of Clinical Neuropsychology*. Springer-Verlag New York.

Jiménez, J. G., & Baturone, A. O. (1996). *Estimación de la posición de un Robot Móvil.* Automática, (29), 3-18.

Joglekar, A., Joshi, D., Khemani, R., Nair, S., & Sahare, S. (2011). *Depth estimation using monocular camera.* International journal of computer science and information technologies, 2(4), 1758-1763.

Junxiang Ge, F. S. (2016). *RRT-GD: An Efficient Rapidly-exploring Random Tree Approach with Goal Directionality for Redundant Manipulator Path Planning.* International Conference on Robotics and Biomimetics.

Kanade, T., & Okutomi, M. (1994). *A stereo matching algorithm with an adaptive window: Theory and experiment.* . IEEE transactions on pattern analysis and machine intelligence, 16(9), 920-932.

Kanika, G., & P, D. (2016). *Unmanned Ground Vehicle Navigation Using Distance Measurement.* International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 5, Issue 4.

Kavraki, L. E., Kolountzakis, M. N., & Latombe, J. C. (1998). A*nalysis of probabilistic roadmaps for path planning.* IEEE Transactions on Robotics and Automation, 14(1), 166-171.

Kheng, L. W. (2012). *Camera Models and Imaging.* Class lecture, CS4243, National University of Singapore.

Kim, H., Kim, D., Shin, J.-U., Kim, H., & Myung, H. (2014). *Angular rate-constrained path planning algorithm for unmanned surface vehicles.* Ocean Engineering, 84, 37-44.

Kim, J. (2003). *Motion Planning of Aerial Robot using Rapidly-exploring Random Trees with Dynamic Constraints.* International Conference on Robotics &Automation .

Kim, M. H. (2014). *Angular rate-constrained path planning algorithm for unmanned surface vehicles.* Ocean Engineering, 37-44.

Klimentjew, D. H. (2010). *Multi Sensor Fusion of Camera and 3D Laser Range Finder for Object Recognition*. Integration for Intelligent Systems.

Kohlbrecher, S. &. (2011). *A Flexible and Scalable SLAM System with Full 3D Motion Estimation.* International Symposium on Safety,Security and Rescue Robotics.

Kuen-Han, L. C.-H. (2012). *Mapping and Localization in 3D Environments Using a 2D Laser Scanner and a Stereo Camera.* Journal of information sience and engineering.

Kumar, S. (2010). *Sensor Fusion of Laser & Stereo Vision Camera for Depth Estimation and Obstacle Avoidance.* International Journal of Computer Applications.

Kümmerle, B. R. (2014). *Autonomous Robot Navigation in Highly Populated Pedestrian Zones.* Field Robotics.

Kurt, T. E. (2015). *GitHub*. Obtenido de arduino-serial: https://github.com/todbot/arduino-serial

Lakhwani, A. M., Shah, K. H., Vaghela, A. S., Panchal, D. S., & Rathod, S. R. (2018). *Review on Basics of Computer Vision and Its Applications*. Research & Reviews: Journal of Computational Biology, 6(2), 33-40.

Langley, C. (2005). *Soldiers in the Laboratory.* Scientits for Global Responsability.

Laser Safety Facts. (2018). *About laser classes*. Obtenido de http://www.lasersafetyfacts.com/laserclasses.html

Latombe, J. C. (1990). *Robot Motion Planning* (the Kluwer international series in engineering and computer science).

LaValle, S. M. (1998). *Rapidly-exploring random trees: A new tool for path planning.*

Lecumberry, F. (2005). *Cálculo de disparidad en imágenes estéreo, una comparación.* XI Congreso Argentino de Ciencias de la Computación.

Lee, J. H. (2016). *Learning High-Dimensional Mixture Models for Fast Collision Detection in Rapidly-Exploring Random Trees.* IEEE International Conference on Robotics and Automation (ICRA).

Lens, S. (2015). *Efficient and Precise Trajectory Planning for Nonholonomic Mobile Robots.* (Doctoral dissertation, Université de Liège, Liège, Belgique).

Levoy, G. T. (1994). *Zippered polygon meshes from range images.* SIGGRAPH'94.

Lite, D. (2016). *Learning High-Dimensional Mixture Models for Collision Deteccion*. International Conference on Robotics and Biometrics.

Liu, W. (2015). *Autonomous vehicle planning system design under perception limitation in pedestrian environment.*

López García, D. A. (2011). *Nuevas aportaciones en algoritmos de planificación para la ejecución de maniobras en robots autónomos no holónomos.*

López Valverde, A. (2009). *Sistema locomotor y de localización de un microrobot (EUROBOT 2009)* . (Bachelor's thesis).

Luettel, T. H., & Wuensche, H. J. (2012). *Autonomous ground vehicles—Concepts and a path to the future.* Proceedings of the IEEE, 100(Special Centennial Issue), 1831-1839.

M. Y. Kim, H. C. (2004). *An Active Trinocular Vision System for Sensing Mobile Robot Navigation Environments.* IEEE Intelligent Robots and Systems Proceedings, 1698-1703.

Martínez, M. M. (2010). *Técnicas de visión estereoscópica para determinar la estructura tridimensional de la escena.*

Matematikcentrum Lundus. (2015). *The Pinhole Camera Model.* Computer Vision Lecture 1.

Matheis, M. (2016). *Evaluation of depth-camera-systems for usage in semi-controlled assembly environments.*

May, D. H. (2009). *Robust 3D-Mapping with Time-of-Flight Cameras.* Intelligent Robots and Systems.

Mazola Colllazo, N. (1991). *Manual del Sistema Internacional de Unidades*. Editorial Pueblo y Educación.

Meers, S. &. (2004). *A Vision System for Providing 3D Perception of the Environment via Transcutaneous Electro-Neural Stimulation.* Proceedings of the Eighth International Conference on Information Visualisation.

Merrick, K. (2017). *Value systems for developmental cognitive robotics: A survey.*

Meza, F., & Ramos, P. (2015). *Modelado Matemático Motor DC Conexión Independiente* . Guayaquil.

Montúfar, P., Salazar, H., Aguilar, W. G., Segura, L., & Loza, D. (2017). *Kalman filter implementation in a working cell to classify parts that are in motion.* Chilean Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON). Pucón, Chile.

Moon, S. (2013). *Spline-Based RRT Path Planner for Non-Holonomic.* Springer Science.

Moravec, H., & Elfes, A. (1985). *High resolution maps from wide angle sonar.* Robotics and Automation. Proceedings. 1985 IEEE International Conference, 116-121.

Muñoz, V. (1995). *Planificacion de trayectorias para robots móviles. (*Doctoral dissertation, Tesis Doctoral. Universidad de Málaga).

Murray, R. M., Li, Z., Sastry, S. S., & Sastry, S. S. (1994). A mathematical introduction to robotic manipulation. CRC press.

Naderi, K., Rajamaki, J., & Hamalainen, P. (2015). *RT-RRT*: A Real-Time Path Planning Algorithm Based On RRT*.* MIG .

Navas Flores, D. F., Suasnavas, V., & Ramiro, J. (2018). *Diseño e implementación de un prototipo de detección y localización de obstáculos a través de reconstrucción tridimensional mediante el uso de una cámara estereoscópica.* (Bachelor's thesis, Quito, 2018.).

Nie, C., Corcho, X. P., & Spenko, M. (2013). *Robots on the move: Versatility and complexity in mobile robot locomotion.* IEEE Robotics & Automation Magazine, 20(4), 72-82.

Normey-Rico, J. E., Alcalá, I., Gómez-Ortega, J., & Camacho, E. F. (2001). *Mobile robot path tracking using a robust PID controller.* Control Engineering Practice, 9(11), 1209-1214.

Nuchter, S. H. (2003). *Automatic Model Refinement for 3D Reconstruction with Mobile Robots.* 3-D Digital Imaging and Modeling.

NVIDIA. (2018). *CUDA Toolkit 9.1*. Obtenido de https://developer.nvidia.com/cuda-downloads

Orbea, D., Moposita, J., Aguilar, W. G., Paredes, M., Reyes, R. P., & Montoya, L. (2017). *Vertical take off and landing with fixed rotor*. Chilean Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON). Pucón, Chile.

Pajares, G., & Cruz, J. M. (2007). *Visión por Computador: Imágenes digitales y aplicaciones*. RA-MA, 978-8.

Panich, S. (2010). *Comparison of Distance Measurement Between*. Journal of Computer Science 6 , 1079-1081.

Pardo, J. A., Aguilar, W. G., & Toulkeridis, T. (2017). *Wireless communication system for the transmission of thermal images from a UAV.* Chilean Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON). Pucón, Chile.

Pérez Porto, J., & Merino, M. (2018). *Definición de intensidad*. Obtenido de https://definicion.de/intensidad/

Peter Kaldén, E. S. (2015). *Development of a low-cost laser range-finder (LIDAR).* Master's Thesis in Systems, Control and Mechatronics.

Piazzi, A., Bianco, C. L., Bertozzi, M., Fascioli, A., & Broggi, A. (2002). *Quintic G/sup 2/-splines for the iterative steering of vision-based autonomous vehicles*. IEEE Transactions on Intelligent Transportation Systems, 3(1), 27-36.

Pierrottet, D., Amzajerdian, F., Petway, L., Barnes, B., Lockard, G., & Rubio, M. (2008). *Linear FMCW laser radar for precision range and vector velocity measurements.* MRS Online Proceedings Library Archive, 1076.

Pillai, R. L. (2016). *High-Performance and Tunable Stereo Reconstruction.* IEEE International Conference on Robotics and Automation (ICRA).

Pino, A. M. (s.f.). *Entorno virtual de visualización 3D de la vía óptica y sistema oculomotor, a partir de secciones seriadas de resonancia magnética.* Universidad de Salamanca.

Portillo, M. T., & Plata, J. A. (2008). P. CH. *Mahalanobis y las aplicaciones de su distancia estadística*. CULCyT: Cultura Científica y Tecnológica, (27), 13-20.

Purcaru, I. C. (2013). *Hybrid PSO-GSA robot path planning algorithm in static environments with danger zones.* System Theory, Control and Computing.

R, O., Zhou, C., Adams, J., & Bodenheimer, B. (2003). *Interface Evaluation for Mobile Robot Teleoperation.* Proceedings of the ACM Southeast Conference (ACMSE03), 112-118.

Rastelli, J. P., Lattarulo, R., & Nashashibi, F. (2014). *Dynamic trajectory generation using continuous-curvature algorithms for door to door assistance vehicles.* Intelligent Vehicles Symposium Proceedings, 510-515.

Reeds, J., & Shepp, L. (1990). *Optimal paths for a car that goes both forwards and backwards.* Pacific journal of mathematics, 145(2), 367-393.

Reeds, J., & Shepp, L. (1990). *Optimal paths for a car that goes both forwards and backwards.* Pacific journal of mathematic, 367-393.

Risheng Kang, H. L. (2016). F*ast Convergence RRT for Asymptotically-optimal Motion Planning.* International Conference on Robotics and Biomimetics.

Roberge, V. (2012). *Comparison of Parallel Genetic Algorithm and Paticle swarm.*

ROS.org. (2018). *hector_mapping* . Obtenido de http://wiki.ros.org/hector_mapping

Ruiz, A. (2015). *Sistemas de Percepción y Visión por Computador.*

S. Guttman, L. G., & R, B. (2007). *Estimación de Profundidad.* Una Introducción. Vision Research, 47 219230.

S. K. Gehrig, F. E. (2009). *A real-time low-power stereo vision engine using semi-global matching.* Computer Vision Systems. Springer.

Salgado, M. F., Tierra, A., & Aguilar, W. G. (2017). T*ravel Planning in Public Transport Networks Applying the Algorithm A* for Metropolitan District of Quito.* Analysis, 4, 11.

Salgado, M. F., Tierra, A., Sandoval, D. S., & Aguilar, W. G. (2017). T*ravel Time Estimation of Public Transport Networks Based on Commercial Incidence Areas in Quito Historic Center.* Analysis, *294*, 78448.

Sánchez, E. V., Oliva Meyer, M. Á., & Sánchez Lozano, M. (2013). *El sistema de Dirección.*

Santos, J. M., Portugal, D., & Rocha, R. P. (2013). *An evaluation of 2D SLAM techniques available in robot operating system.* Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium, 1-6.

Sanz, P. R., Mezcua, B. R., & Pena, J. M. (2012). *Depth estimation-an introduction.* Current Advancements in Stereo Vision, InTech.

Sara Bouraine, T. F. (2016). *Real-time Safe Path Planning for Robot Navigation in Unknown Dynamic Environments.* Computing Systems and Applications.

Sathiyanarayanan, M. (2011). *Unmanned Ground Vehicle.* Undergraduate Thesis, Rajiv Gandhi Instititute of Technology.

Saurav, K. (2010). *Sensor Fusion of Laser & Stereo Vision Camera for Depth Estimation and Obstacle Avoidance.* International Journal of Computer Applications.

Saxena, A., Schulte, J., & Ng, A. Y. (2007). *Depth Estimation Using Monocular and Stereo Cues*. In IJCAI, Vol. 7.

Se, J. (2008). *Stereo-Vision Based 3D Modeling and Localization for Unmanned Vehicles.* Internatonal Journal of Inteligent Control and Systems.

Sleeswyk, A. W. (1981). *Vitruvius' odometer.* Scientific American, 245(4), 188-201.

Soucy, M., Godin, G., & Rioux, M. (1996). *A texture-mapping approach for the compression of colored 3D triangulations.* The Visual Computer, 12(10), 503-514.

Souza, A. A., Maia, R., & Gonçalves, L. M. (2012). *Probabilistic occupancy grid to robotic mapping with stereo vision.* Current Advancements in Stereo Vision.

Stavrakis, E. (s.f.). *Stereoscopic Non-Photorealistic Rendering.*

Stereolabs. (2018). *Camera Class Reference*. Obtenido de https://www.stereolabs.com/developers/documentation/API/v2.2.0/

Stereolabs. (2018). *Positional Tracking*. Obtenido de https://docs.stereolabs.com/overview/positional-tracking/introduction/

Stereolabs. (2018). *SDK Downloads*. Obtenido de https://www.stereolabs.com/developers/release/2.3

Stereolabs. (2018). *Spatial Mapping*. Obtenido de https://docs.stereolabs.com/overview/spatial-mapping/introduction/

Steux, B., & El Hamzaoui, O. (2010). *tinySLAM: A SLAM algorithm in less than 200 lines C-language program.* Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference , 1975-1979.

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

Sun, J., Li, Y., Kang, S. B., & Shum, H. Y. (2005). *Symmetric stereo matching for occlusion handling.* Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference, Vol. 2, pp. 399-406.

Sun, S.-L. &.-L. (2004). *Multi-sensor optimal information fusion Kalman filter.* ELSIVER.

Szeliski, D. S. (2002). *A taxonomy and evaluation of dense two-frame stereo correspondence algorithms.* Int'l J. of Computer Vision, vol. 47, no. 1-3.

Tdk IvenSense. (2018). *MPU 6050*. Obtenido de https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/

Team-Paragon. (2013). *Motor Comparison Chart for FIRST Robotics*. Obtenido de http://team-paragon.org

Tedrake, J. B. (2015). *Pushbroom stereo for high-speed navigation in cluttered environments*. IEEE Int'l Conf. on Robotics and Automation (ICRA).

Tezanos, R. C. (2015). *Percepción Basada en Visión estereoscópica, planificación de trayectorias y strategias de navegación para exploración robótica autónoma*.

Torres Moreno, J. L. (2014). *Análisis multidominio de vehículos eléctricos.* Universidad de Almeria, Tesis octoral.

Turk, G., & Levoy, M. (1994). *Zippered polygon meshes from range images.* Proceedings of the 21st annual conference on Computer graphics and interactive techniques, 311-318.

Universidad Carlos III de Madrid. (2016). *Autonomous Unmanned Ground Vehicle Systems.* Intelligent Systems Laboratory.

URG Library document. (2018). *Building the library and sample programs*. Obtenido de http://urgnetwork.sourceforge.net/html/

Valle, L. (1998). *Rapidly Exploring Random Trees: A new tool for path planning.* Iowa State University.

Veitch-Michaelis, J. L. (2017). Fusion of LIDAR with stereo camera data-an assessment . *London: (Doctoral dissertation, UCL (University College London)).*

Verma, N. (2014). *Path planning for unmanned aerial vehicle based on genetic algorithm & artificial neural network in 3D.* Data Mining and Intelligent Computing.

Visual Computing Lab . (2018). *MeshLab*. Obtenido de http://www.meshlab.net/#description

Wang, C. M. (1988). *Location estimation and uncertainty analysis for mobile robots.* Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference , 1231-1235.

Wang, X. (2013). *2D Mapping Solutionsfor Low Cost Mobile Robot.*

Watson, G. R., & DeBardeleben, N. A. (2006). *Developing Scientific Applications Using Eclipse.* Computing in Science & Engineering 8, 50.

Weingarten, G. S. (2003). *A Fast and Robust 3D Feature Extraction Algorithm for Structured Environment Reconstruction.* Obtenido de https://infoscience.epfl.ch/record/97503/files/P250.pdf

Wu, Q., Hartley, J. K., & Al-Dabass. (2005). *Time-dependent stochastic shortest path (s) algorithms for a scheduled transportation network.* International Journal of Simulation Systems, Science & Technology, 6(78), 53-60.

Wulf, O. A. (2004). *2D mapping of cluttered indoor environments by means of 3D perception.* Robotics and Automation.

Yan-Jiang Zhao, B. K. (2015). *3D Motion Planning for Robot-Assisted Active Flexible Needle Based on Rapidly-Exploring Random Trees* . Journal of Automation and Control Engineering , 3(5).

Yoshida, K. a. (1996). *A stereo machine for video-rate dense depth mapping and its new applications.* Computer Vision and Pattern Recognition.

Zbontar, L. (2015). *Computing the stereo matching cost with ˇ a convolutional neural network.* IEEE Conf. on Computer Vision and Pattern Recognition (CVPR).

Zill, D. (2012). *Ecuaciones Diferenciales.* Mexico: Thomson.

Zong, L., Gong, X., & Guo, C. (2012). *Vehicle System Dynamics: International Journal of Vehicle Mechanics and Inverse neuro-fuzzy MR damper model and its application in vibration control of vehicle suspension system.*