DESARROLLO DE UN SISTEMA DE PERCEPCIÓN ROBÓTICA PARA LA LOCALIZACIÓN MÉTRICA Y TOPOLÓGICA EN AMBIENTES SEMI-ESTRUCTURADOS

JUAN DAVID RICO ROMERO
2180674
DIEGO ALEJANDRO ESCOBAR MONTOYA
2160157

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE AUTOMÁTICA Y ELECTRÓNICA
PROGRAMA INGENIERÍA MECATRÓNICA
SANTIAGO DE CALI
2021

DESARROLLO DE UN SISTEMA DE PERCEPCIÓN ROBÓTICA PARA LA LOCALIZACIÓN MÉTRICA Y TOPOLÓGICA EN AMBIENTES SEMI-ESTRUCTURADOS

JUAN DAVID RICO ROMERO
DIEGO ALEJANDRO ESCOBAR MONTOYA

Proyecto de grado para optar al título de
Ingeniero Mecatrónico

Director
VICTOR ADOLFO ROMERO CANO
MAGISTER EN INGENIERÍA ELÉCTRICA
DOCTOR EN ROBÓTICA DE CAMPO

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE AUTOMÁTICA Y ELECTRÓNICA
PROGRAMA INGENIERÍA MECATRÓNICA
SANTIAGO DE CALI
2021

Nota de aceptación:

Aprobado por el Comité de Grado en cumplimiento de los requisitos exigidos por la Universidad Autónoma de Occidente para optar al título de Ingeniero Mecatrónico

JOSE LUIS PANIAGUA JARAMILLO
Jurado

JUAN CARLOS PERAFAN
Jurado

Santiago de Cali, 28 de mayo de 2021

AGRADECIMIENTOS

Primero que todo, doy gracias a Dios por la oportunidad de finalizar mi carrera universitaria, a mi madre Luz Stella Escobar, mi madrina Blanca Adíela Pérez, mi prima Beatriz Elena García y demás familiares que me brindaron su incondicional apoyo y compañía en todo momento; doy gracias a mis amigos que hicieron parte de esta larga y compleja travesía, a los buenos momentos compartidos, alegrías y tristezas ya que todo esto me hizo crecer como persona y como profesional.

Doy gracias a todos mis profesores, especialmente a Cesar Augusto Romero, Jimmy Tombé, Juan Manuel Nuñes, Diego Fernando Almario y Cesar Marino Rojas, quienes siempre me brindaron lo mejor de sí mismos en cada etapa de mi carrera y con sus consejos, historias, anécdotas y vasta experiencia, hicieron de mi un profesional apasionado por lo que hago. A el semillero de robótica y sistemas autónomos (RAS) , el cual fue parte fundamental e indispensable tanto en la construcción del nuestro proyecto como en mi formación profesional. Finalmente, a nuestro director Víctor romero cano, quien nos apoyó incondicionalmente, nos motivó a crecer, a dar lo mejor de nosotros y asumir retos.

De nuevo, gracias a todos.

“La sabiduría no es producto de la educación, sino del intento de toda la vida para adquirirla” Albert Einstein

DIEGO ALEJANDRO ESCOBAR MONTOYA

Agradezco a mi padre Fredy Alexander Rico Castaño, mi madre Nancy Magnolia

Romero Molina y mi abuela Argenis Castaño Lozano, quienes me apoyaron en todo momento de mi carrera profesional. A ellos les dedico este trabajo ya que resume gran parte de mis estudios durante estos años. Así mismo agradezco a mi director de tesis Víctor Adolfo Romero Cano, quien dirigió este trabajo con mucho profesionalismo y me guio con su extenso conocimiento en la robótica móvil, de igual forma hago un extenso reconocimiento al semillero de robotica o RAS(Robotic Autonomous Systems) quien nos suministró todos los complementos necesarios para desarrollar este proyecto.

JUAN DAVID RICO ROMERO

CONTENIDO

LISTA DE FIGURAS

pág.

LISTA DE TABLAS

pág.

LISTA DE ANEXOS

pág.

RESUMEN

El presente proyecto consistió en el desarrollo de un sistema de percepción capaz de localizar tanto métrica como topológicamente un robot móvil tipo Ackermann en ambientes semiestructurados. Este sistema de percepción se dividió en dos partes, la primera en la ubicación del robot métricamente por coordenadas cartesianas y la y la segunda en la identificación de lugares del entorno de trabajo por visión computacional (redes neuronales convolucionales).

El primer sistema utilizó sensores propioceptivos como lo son el IMU y los encoders de los motores, con el fin de generar la localización métrica del modelo físico en un plano cartesiano. Asimismo, este sistema empleó la ubicación virtual del robot contenido en el simulador Gazebo, con el objetivo de estimar una "super" odometría con error pequeño respecto a la medida real del entorno; resaltando que la odometría final calculada por el sistema, utilizó un filtro de Kalman el cual generaba la estimación en base al conjunto de información de la entradas (localización real y virtual).

Por otro lado, el segundo sistema empleó un tipo de red neuronal artificial capaz de identificar objetos en imágenes digitales. Esta red fue la CNN o red neuronal convolucional, la cual permitió la localización de objetos o lugares contenidos en una imagen digital. Adicionalmente, para la adquisición de datos de dicha red, se empleó una cámara monocular con transmisión de datos por puerto USB, de igual manera fue utilizada para obtener imágenes de validación para el sistema topológico, cada uno de estos sistemas estuvo compuesto por una serie de programas y nodos de ROS, que al integrarse y complementarse con un entorno simulado (Gazebo) permitieron la unificación de sistemas (métrico y topológico).

Finalmente el objetivo de este proyecto, es servir como base para un robot guía de personas al interior del campus de la Universidad Autónoma de Occidente específicamente en el sótano dos, basados en una plataforma robótica tipo Ackermann ya existente en la universidad.

Palabras clave:

Ackermann, sensor, semiestructurados, propioceptivo.

INTRODUCCIÓN

Por siglos el ser humano se ha dedicado a construir maquinas que le permitan imitar las partes y funciones tanto de animales como del cuerpo humano. Sin embargo, estos desarrollos no han sido mas que objetos de admiración sin una aplicación real. El primer acercamiento real hacia una verdadera automatización, ocurrió en el siglo XVII por el matemático Blas Pascal con su máquina calculadora. Luego encontramos que en la revolución industrial ocurrió un punto de inflexión, en el cual, se impulsó el desarrollo de maquinas que sirvieran de apoyo en trabajos pesados o en aplicaciones que permitieran mejorar la eficiencia en la producción. Además, es allí donde en su continua evolución, surge la percepción robótica y todo lo que esto implicaba en términos de desarrollo, ya que no era suficiente solo realizar tareas preestablecidas, sino que dependiendo de las condiciones del entorno, se generara una u otra respuesta.

Una de las varias ramas de la robótica en la que se ha trabajado durante mucho tiempo, ha sido en los sistemas de percepción de ambientes, dado que las aplicaciones en las que se ha implementado, son un paso para un futuro mejor. un ejemplo de esto, son los sistemas de navegación autónoma de tesla, el cual implementa un sistema avanzado que requiere de una caracterización del entorno con el fin de conducir adecuadamente el vehículo de manera autónoma. El desarrollo de este tipo de sistemas para aplicaciones de menor complejidad, tendría un impacto positivo en diferentes ámbitos, lo que se traduce en un aumento de la productividad en ciertas áreas, desarrollo de hardware que aproveche esta tecnología y un incremento en la inversión con el fin de seguir mejorándola.

Por otro lado, en cuanto a la investigación se ha evidenciado que en el área de la robótica y más específicamente en sistemas de percepción de ambientes, se utilizan los vehículos no tripulados tanto aéreos como terrestres. Sin embargo, según la aplicación, se analiza el entorno de interés donde se planea implementar el proyecto y se seleccionan los sensores. En ambientes semiestructurados con topología plana los sensores más utilizados son los propioceptivos tipo IMU y enconder, ya sea por su confiabilidad como por su bajo error (según el método de implementación).

Una de las aplicaciones mas utilizadas en los sistemas de percepción de ambientes semiestructurados con topología plana, son la localización métrica y topológica, la cual, permite conocer mediante líneas por donde ha transitado el robot según un sistema de coordenadas, el cual cuenta con su respectivo punto de referencia (sistema métrico). Finalmente realiza el análisis topológico, en el cual determina mediante características del entorno su ubicación.

La integración de ambos sistemas permite conocer la ubicación en el mapa 3D, las líneas por onde ha transitado y la respectiva predicción de ubicación según los puntos de interés.

1. PLANTEAMIENTO DEL PROBLEMA

En un mundo cada día más tecnológico, la calidad de vida tiene una estrecha relación con la creación de experiencias únicas en tareas cotidianas, como la automatización del transporte de elementos de un punto a otro o la limpieza de áreas por navegación autónoma. Estas experiencias permiten disminuir el tiempo en la ejecución de tareas y reducción de riesgos. La robótica ha abordado estas problemáticas desde muchas líneas, especialmente en la robótica móvil donde la auto localización es un factor importante para los robots móviles [1], ya que una de sus principales características es tener la capacidad de navegar en múltiples entornos. Debido a esto, alrededor del mundo se han desarrollado alternativas que cumplan con este fin.

Por otra parte, la percepción espacial es una habilidad fundamental necesaria para que los robots móviles autónomos se muevan de manera robusta y segura en el mundo real [2] del mismo modo, el semillero de robótica y sistemas autónomos de la Universidad Autónoma de Occidente no es ajeno a esta tendencia, por lo cual, viene desarrollando este tipo de tecnologías, entre ellas se encuentra la localización métrica y topológica de robots móviles, las cuales permiten su desplazamiento, localización y verificación de su objetivo. Sin embargo, estos sistemas aun no son aplicados a un problema en concreto como lo es guiar personas dentro del sotado dos de la Universidad Autónoma de Occidente. Estos individuos por lo general se localizan y orientan por indicaciones de terceros, como lo son guardas, profesores, personal administrativo o simplemente compañeros de estudio. No obstante, la gran mayoría de personas nunca llegan a su lugar de destino con esta información, aumentando la desorientación del individuo. Asimismo, este problema crece en gran medida cuando se debe orientar a una persona con movilidad reducida o algún tipo de discapacidad. Por lo cual, se plantea la creación de un sistema de percepción espacial, capaz de localizar métrica y topológicamente un robot móvil para la orientación de personas dentro de un espacio de superficie plana, como es el caso del sótano dos de la Universidad Autónoma de Occidente. Por esta razón se desea abordar esta problemática y generar una solución al siguiente interrogante.

¿Cómo dar la capacidad a un robot móvil de conocer su entorno y localizarse en el mismo?

2. JUSTIFICACIÓN

A lo largo de la vida humana, se encuentra la necesidad de conocer la ubicación y el entorno en el cual se deambula, con el fin de poder realizar desplazamientos hasta un punto determinado. Al arribar a un entorno desconocido es importante contar con el apoyo o guía de personal que tenga previos conocimientos de las ubicaciones en su entorno, ya que los desplazamientos pueden dificultarse para personas que no cuenten con las mismas capacidades motoras o sensoriales porque algunas de estas personas suelen tener mayores problemas para captar toda la información necesaria al ubicarse.

La presente investigación, tiene como objeto crear un sistema robótico que permita suplir la necesidad de guiar a personas de la comunidad educativa en general, ya que personas del común que buscan una orientación dentro del sótano dos presentan dificultades de localización. Además, estas mismas limitaciones se incrementan cuando se trata de personas con movilidad reducida o discapacidad sensorial que se movilizan en la superficie plana del sótano dos de la Universidad Autónoma de Occidente. Por lo cual, se evidencia que este sector de la población encuentra barreras limitantes que impiden el desplazamiento adecuado a lo largo del sótano dos, como la falta de personas guías o soluciones tecnológicas para facilitar la ubicación de las personas desorientadas.

De acuerdo a lo anterior, se requiere un sistema que logre ampliar la percepción espacial y mejore la orientación-guía de las personas. Por lo cual, un sistema con la capacidad de satisfacer esta necesidad es una plataforma móvil con percepción espacial robótica, la cual puede apoyar la ubicación de las personas por medio de procedimientos de percepción visual y sensorial que están constituidos por habilidades espaciales precisas, robustas y rápidas [3], tales como odometría visual, localización, mapeo simultáneos (SLAM), visión artificial [4][5][6], planeación y generación de trayectorias. Así mismo, estos sistemas pueden ser implementados en robots móviles, los cuales tienen la capacidad de poder desplazarse en diferentes entornos, ajo este contexto contribuimos a la expansión de la tecnología al servicio social de la población.

3. OBJETIVOS

3.1 OBJETIVO GENERAL

Desarrollar un sistema de localización métrica y topológica para un robot móvil que permita el desplazamiento, localización y verificación de su objetivo planeado.

3.2 OBJETIVOS ESPECIFICOS

- Hacer revisión del estado del arte sobre los diferentes sistemas robóticos de localización 2D en ambientes semiestructurados como universidades o centros de investigación.

- Implementar una técnica de localización para robots móviles en ambientes semiestructurados.

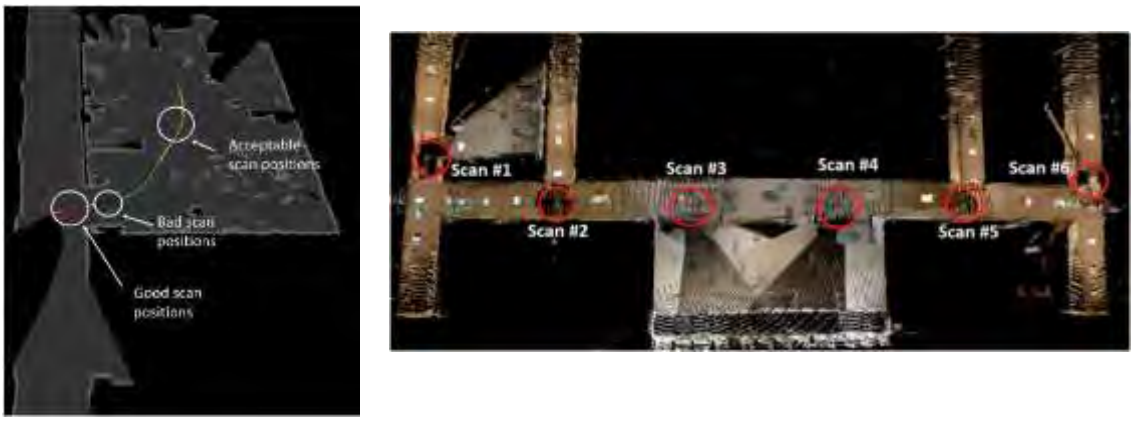
- Desarrollar un sistema de reconocimiento de lugares basado en apariencia para la localización topológica del robot móvil en la universidad.

- Evaluar el desempeño de la implementación.

4. ANTECEDENTES

Durante el desarrollo de esta sección se da a conocer proyectos que pretenden resolver diferentes situaciones de percepción espaciales, los cuales guardan un grado de similitud muy alto con la aplicación del proyecto que se planteó en este documento y sirven como base de partida para su desarrollo.

El sistema desarrollado por Pileun Kim, Jingdao Chen, Yong K. Cho en el área de automatización en la construcción, donde se plantea un robot móvil autónomo que navega y escanea continuamente el sitio, el cual se basa en un mapa de nube de puntos. El sistema del robot móvil utiliza la técnica de localización y mapeo simultáneo 2D (SLAM) para estimar las posiciones y orientaciones en tiempo real del robot en el plano x y. Finalmente, la información de localización 2D es usada para crear nubes de puntos 3D de entornos desconocidos en tiempo real para determinar sus rutas de navegación como un proceso de exploración previa. Este sistema tiene la capacidad de capturar datos de nubes de puntos mapeados en RGB con calidad de levantamiento y registrar automáticamente los escaneos para la reconstrucción geométrica del sitio.



A) B)

Fig. 1. Evaluación de trayectoria. [7].

Sistema de localización autónoma para robots móviles basado en fusión de sensores propioceptivos.

Este proyecto desarrolló un sistema de localización autónomo capaz de suministrar una mejor estimación de posición en comparación a muchos sistemas como lo son

aquellos basados en odometría pura. En este caso, se empleó un robot para la ejecución de una trayectoria programada, con el fin de adquirir datos para sensores internos como acelerómetros, giroscopios y odómetros con la finalidad de fusionarlos y obtener un modelo de medición. (Ver Fig. 2)

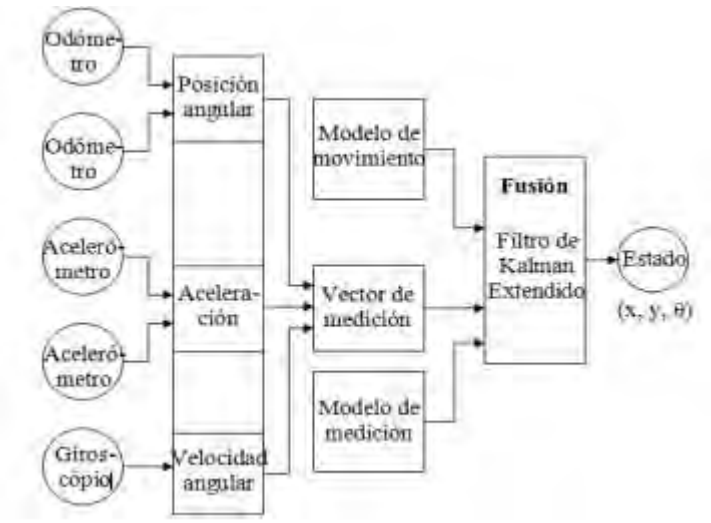


Fig. 2. Estructura de fusión de sensores [8].

Diseño de un sistema de navegación autónomo para robots móviles usando fusión de sensores y controladores neuro difusos.

El sistema de navegación fue diseñado en dos etapas. En la primera se desarrolló e implementó un controlador que permitió al robot planear mediante el uso de controladores neuro-difusos y el uso de cuadrículas de certeza. La segunda etapa del proyecto tomó el problema existente de la localización en un entorno desconocido, en el cual se usó la fusión de sensores odométricos, inerciales y redes de posicionamiento global.

Adicionalmente, mediante el uso de un robot Pioneer P3-AT, se implementó un híbrido entre los métodos de navegación por fusión de conductas y los métodos basados en campo de fuerzas, lo cual hizo posible aprovechar las ventajas de ambos métodos. (Ver Fig. 3)

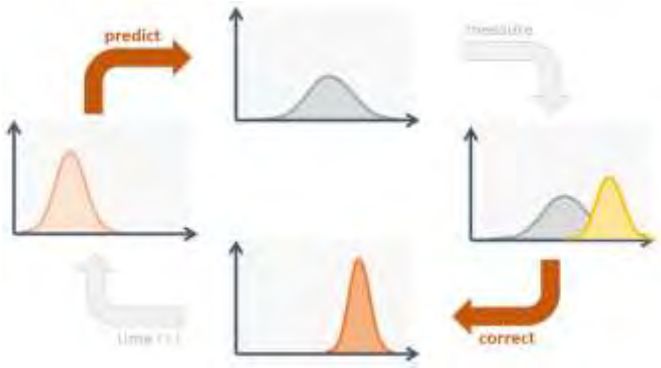


Fig. 3. Controlador Fuzzy [9].

5. MARCO CONCEPTUAL

5.1 ANATOMÍA DE UN ROBOT MÓVIL

El dispositivo mecatrónico por excelencia es el robot móvil, ya que al contar con la capacidad de percibir, entender y desplazarse en un entorno espacial, este puede generar su propio desplazamiento por la acción de actuadores rotacionales o traslacionales. Además, gran parte de los dispositivos robóticos se pueden adaptar a entornos variados como el campus de la Universidad Autónoma de Occidente, donde este combina un ambiente de terrenos planos (oficinas, cafeterías, pasillos) e inclinados (rampas y terrenos con geografía variada).

5.1.1 Sistemas de locomoción

- Ruedas/cintas de deslizamiento
- Diferencial
- Síncrona
- Triciclo
- Ackerman
- Omnidireccionales
- Patas

Para iniciar la introducción en los sistemas de locomoción de robots móviles, es importante tener en cuenta conceptos de diseño en la estructura física del robot, relacionado con el chasis, carcasa y cálculos de esfuerzo entre otros. Así como también, es relevante tener en cuenta la solución de problemas al momento de realizar las pruebas de movimiento del mismo.

5.1.1.1 Maniobrabilidad

También llamada manejabilidad es una característica de comportamiento en un vehículo, la cual representa la sensibilidad de los mandos del coche con respecto a los comandos ingresados por el piloto de dicho automotor, sin importar las condiciones del entorno.

5.1.1.2 Controlabilidad

Es la capacidad de cambiar el estado actual o inicial a cualquier otro valor o estado conocido dentro de un intervalo de tiempo admisible, sin embargo, esto no garantiza que este permanezca allí sin cambiar.

5.1.1.3 Tracción

Es el mecanismo que se encarga de transmitir la potencia del motor hacia las ruedas ya sea a los neumáticos frontales, traseros o a ambos, esto se llama tracción trasera, delantera o integral.

5.1.1.4 Ruedas. (Ver imagen 17)

- Motriz: Es la rueda que proporciona fuerza de tracción al robot.

- Rueda directriz: Es la rueda que se encarga de dar el direccionamiento con el fin de llegar a una orientación deseada mediante un método controlable.

- Rueda fija: Son las ruedas que sólo giran en torno a su eje sin tracción motriz.

- Rueda de castor: Ruedas orientables no controladas.

5.1.2 Sistemas de coordenadas

El concepto de sistema de coordenadas en el campo de la robótica móvil tiene gran importancia, ya que permite determinar o estimar la posición de uno o varios objetos dentro de un marco de referencia. Adicionalmente, existen diferentes tipos de

sistemas de coordenadas como lo son las cartesianas, esféricas, cilíndricas y polares.

5.1.2.1 Coordenadas cartesianas

Este tipo de coordenadas están definidas principalmente por dos tipos de sistemas. El primero bidimensional o 2D (ver Fig. 4 izquierda) compuesto por dos ejes ortogonales, el segundo siendo tridimensional o 3D compuesto por tres ejes ortogonales.(ver Fig. 4 derecha)

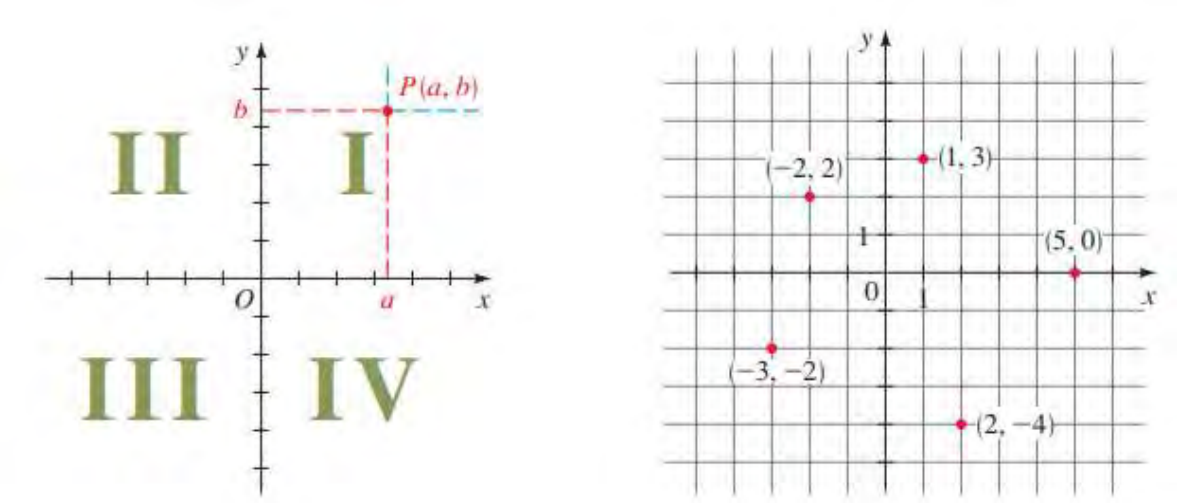


Fig. 4. Coordenadas cartesianas [10]

5.1.2.2 Coordenadas polares

Las coordenadas polares se caracterizan por utilizar dos variables, un ángulo y una distancia denominada radio como se puede observar en la Fig. 5. Estas variables permiten ubicar o representar cualquier objeto en un plano 2D. (ver Fig. 5)

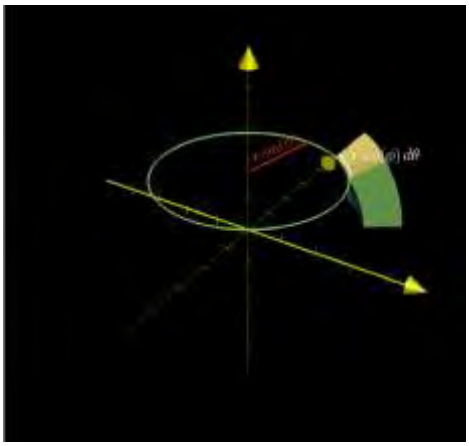


Fig. 5. Coordenadas polares [11]

5.1.2.3 Coordenadas cilíndricas

Las coordenadas cilíndricas utilizan tres variables como lo es el ángulo, distancia de radio y un intervalo llamado altura. Este sistema es similar a la previamente presentada coordenada polar, con la diferencia visible en su variable de altura, ya que agrega una tercera dimensión. (ver Fig. 6)

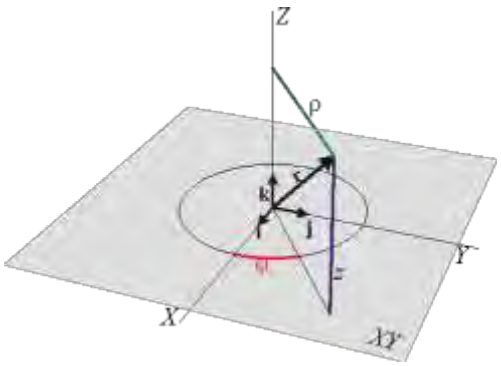


Fig. 6. Coordenadas cilíndricas [12]

5.1.3 Cinemática de un robot móvil

La cinemática en los robots móviles tienen gran relevancia al momento de comprender el comportamiento físico, debido a que representa la base para crear los sistemas de control necesarios para su correcto funcionamiento. En el caso de un robot móvil, las ruedas y su tipo son las encargadas de permitir su movimiento y plantear algunas restricciones de rodamiento.

Adicionalmente, las restricciones de movimiento de las ruedas y su conFig.ción están definidas por expresiones matemáticas que describen la circulación del robot en su área de trabajo.

5.1.3.1 Centro instantáneo de rotación

También llamado ICR por sus siglas en inglés (Instantaneous center of rotation), representa el punto en el cual se cruzan los ejes de las ruedas. Este elemento permite la capacidad de realizar giros controlados en diferentes direcciones. (ver Fig. 7 algunas conFig.ciones de ICR)

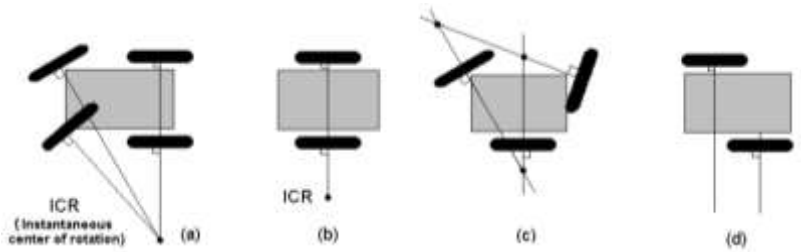


Fig. 7. Centros de rotación instantáneos en diferentes conFig.ciones de robot. [13]

5.1.3.2 Restricciones holonomicas y no holonomicas

Un robot móvil holonómico es una plataforma robótica que posee la capacidad de modificar su dirección de forma instantánea sin necesidad de cambiar su conFig.ción. Por lo tanto, un robot no holonómico debe adaptar su postura o conFig.ción para desplazarse a algunos lugares. Un claro ejemplo es una conFig.ción de robot tipo Ackermann, ya que es necesario fijar una posición angular en las ruedas de dirección, con el fin de lograr un movimiento hacia algún lugar deseado.

5.1.3.3 Orientación de un robot móvil en ángulos de Euler y cuaterniones unitarios

La orientación de un robot móvil basado en ángulos de Euler, se compone por tres coordenadas angulares, las cuales permiten conocer la orientación de un sistema de referencia respecto a otro que generalmente es fijo. Además, los cuaterniones

unitarios tienen un grado inferior de complejidad al momento de componerse, además su uso es mayor en este tipo de aplicaciones robóticas. También, como se puede observar en la Fig. 8 (corresponde a la ecuación 1), los ángulos de Euler utilizan las variables de rotación denominadas “Roll,Pitch y Yaw”, las cuales corresponden a la orientación de cada eje “X,Y,Z”.

$$Angulos\ de\ Euler = \begin{bmatrix} \Phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan\left(\frac{2*(w*x+y*z)}{1-2*(x^2+y^2)}\right) \\ \arcsin(2*(w*y-z*x)) \\ \arctan\left(\frac{2*(w*z+x*y)}{1-2*(y^2+z^2)}\right) \end{bmatrix} \quad \textbf{(1)}$$

Fig. 8. Ángulos de Euler.

Al referirse a los cuaterniones, se debe comprender que se habla de un vector compuesto por 4 variables Q = [Q0 Q1 Q2 Q3]T o Q = [w x y z]T. (ver Fig. 9 que corresponde a la ecuación (2)).

$$Cuaterniones = \begin{bmatrix} \cos\left(\frac{\Phi}{2}\right)*\cos\left(\frac{\theta}{2}\right)*\cos\left(\frac{\psi}{2}\right)+\cos\left(\frac{\Phi}{2}\right)*\cos\left(\frac{\theta}{2}\right)*\cos\left(\frac{\psi}{2}\right) \\ \sin\left(\frac{\Phi}{2}\right)*\cos\left(\frac{\theta}{2}\right)*\cos\left(\frac{\psi}{2}\right)-\cos\left(\frac{\Phi}{2}\right)*\sin\left(\frac{\theta}{2}\right)*\sin\left(\frac{\psi}{2}\right) \\ \cos\left(\frac{\Phi}{2}\right)*\sin\left(\frac{\theta}{2}\right)*\cos\left(\frac{\psi}{2}\right)+\sin\left(\frac{\Phi}{2}\right)*\cos\left(\frac{\theta}{2}\right)*\sin\left(\frac{\psi}{2}\right) \\ \cos\left(\frac{\Phi}{2}\right)*\cos\left(\frac{\theta}{2}\right)*\sin\left(\frac{\psi}{2}\right)-\sin\left(\frac{\Phi}{2}\right)*\sin\left(\frac{\theta}{2}\right)*\cos\left(\frac{\psi}{2}\right) \end{bmatrix} \quad \textbf{(2)}$$

Fig. 9. Cuaterniones.

5.1.3.4 Cinemática directa

Es el análisis matemático que se lleva a cabo con el fin de obtener las velocidades del robot respecto a un marco de referencia global. Los datos necesarios para hacer este análisis, son proporcionados por sensores que miden la velocidad de cada rueda y su dirección. (ver sistema de coordenadas global y local en robot móvil Fig. 10).

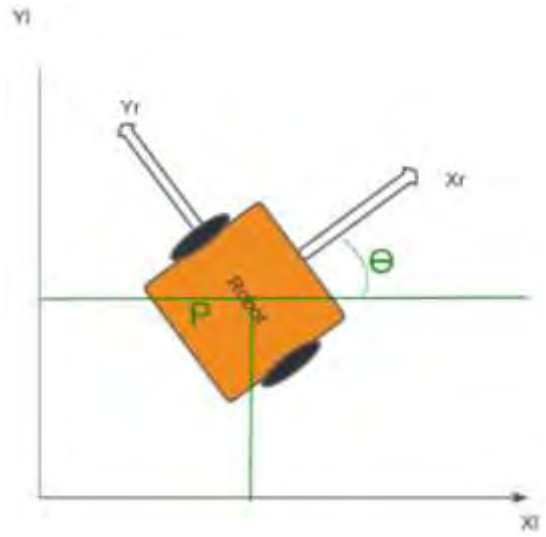


Fig. 10. Sistema de coordenadas global y local del robot [14]

En todas las operaciones se va a considerar que en el robot se encuentra un marco de referencia P, el cual es asignado de manera arbitraria en el chasis. Este punto funcionará como punto de referencia y origen del marco local. A continuación se presenta las ecuaciones correspondientes a la posición y velocidad global;" Xl, Yl", las cuales corresponden a la posición en el eje "X" y "Y" además del "Θ" que es el ángulo entre el eje "Xr" del robot y el eje global "Xl", ver Fig.s 11 y 12.

$$\xi = [Xl,\ Yl,\ \theta l] \tag{3}$$

Fig. 11. Posición global.

$$\dot{\xi} = [\dot{X}l,\ \dot{Y}l,\ \dot{\theta}l] \tag{4}$$

Fig. 12. Velocidad global.

De igual manera se cuenta con las expresiones matemáticas que permiten representar la posición y velocidad según el sistema de coordenadas local, ver Fig. 13 y 14.

$$\xi r = [Xr,\ Yr,\ \theta r] \tag{5}$$

Fig. 13. Posición según sistema coordenado local.

$$\dot{\xi}r = [\dot{X}r,\ \dot{Y}r,\ \dot{\theta}r] \tag{6}$$

Fig. 14. Velocidad según sistema coordenado local.

Conocidas las ecuaciones anteriores, se describe el movimiento del robot en términos de sus componentes usando una matriz de rotación (ver Fig. 15) , la cual es necesaria para mapear el movimiento a lo largo de los ejes de referencia. Esta matriz por lo general se utiliza para rotar ejes de un ángulo a otro. De igual manera, se emplea junto con otras expresiones matemáticas (matrices adicionales) para determinar la cinemática inversa (hallar ángulos de articulaciones).

$$R(\theta) = \begin{bmatrix} \cos\theta & sen\theta & 0 \\ -sen\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

Fig. 15. Matriz de rotación.

La relación directamente proporcional entre el marco local y global, se puede expresar mediante la siguiente ecuación (8) (ver Fig. 16)

$$\dot{\xi}r = \mathbf{R}(\theta)^* \dot{\xi} \tag{8}$$

Fig. 16. Ecuación de relación entre marcos de referencia.

Al momento de diseñar robots se analiza el tipo de aplicación en la cual va a operar el dispositivo mecatrónico, con el fin de seleccionar adecuadamente el tipo de conFig.ción y de llantas que se requieren para su análisis. De acuerdo a esto, el comportamiento cinemático anterior (cinemática directa) es general para todo tipo de robot diferencial (dos ruedas fijas) y no se consideran otros factores como lo son las características de las ruedas y la conFig.ción de estas.

Por otro lado, en el modelado de la contribución de cada rueda se consideran dos restricciones, la restricción de rodamiento y la restricción de deslizamiento.

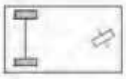
No. de Ruedas	Arreglos	Descripción	Ejemplos típicos
2		Una rueda directriz en la parte delantera, una rueda de tracción en la parte trasera	Bicicleta, motocicleta
		Accionamiento diferencial de dos ruedas con el centro de masa por debajo de los ejes	Robot personal Cye
3		De dos ruedas de accionamiento diferencial centrado con un tercer punto de contacto	Nomad Scout
		Dos ruedas accionadas de forma independiente en la parte frontal/trasera, una rueda omnidireccional sin motor en la parte trasera/frontal	Muchos robots de interiores, incluyendo los robots de EPFL, Pygmalion y Alice
		Dos ruedas de tracción conectadas en la parte trasera, una rueda directriz libre en la parte delantera	Mini truck de Piaggio
		Dos ruedas libres en la parte trasera, una rueda directriz de tracción en la parte de adelante	Neptune, de la universidad Carnegie Mallon
		Tres ruedas motorizadas, suecas o esféricas, arregladas en forma de triángulo; es posible un movimiento omnidireccional	El Tribolo diseñado en EPFL, Palm Pilot Robot Kit

Fig. 17. Tipos de ruedas [15]

En la Fig. 17 se puede observar los tipos de ruedas que puede poseer un robot móvil. Para el caso de la plataforma robótica (chimuelo) empleada en esta investigación, se compone de ruedas tipo estándar. Este robot cuenta con dos tipos de llantas (véase Fig. 18), fijas (Fig. izquierda) y dirigidas (Fig. derecha). A continuación se muestran las variables utilizadas para describir ambas ruedas. Además, una diferencia que resalta entre los dos tipos, es el eje de rotación de la rueda dirigida y la variable que la representa (ángulo B), el cual es constante en la rueda fija y variable en la rueda dirigida.

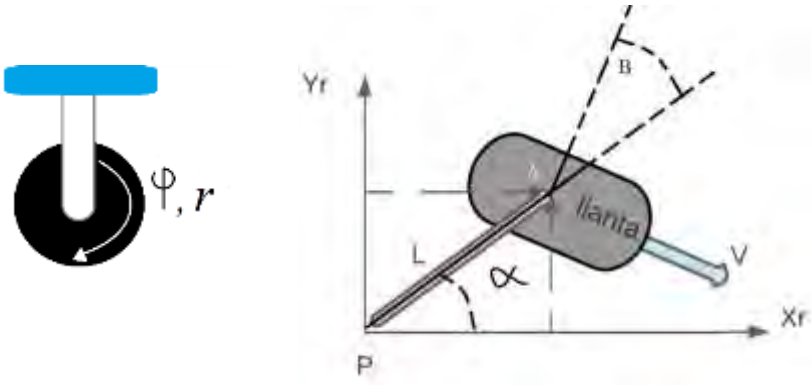


Fig. 18. Rueda fija estándar dirigida [16].

En la imagen 18 se puede observar la variable P, la cual corresponde al punto de referencia local en el robot, el punto A el cual determina el centro de la rueda, la magnitud L que corresponde a la distancia entre el punto P y el punto A, la cual tiene un ángulo α respecto al eje X local y finalmente se encuentra la variable R, la cual representa al radio de la rueda y posición rotacional. A continuación se puede observar su representación matricial compuesta por “s” para ruedas dirigidas y “f” para fijas, ver Fig. 19.

$$\varphi(t) = \begin{bmatrix} \varphi s(t) \\ \varphi f(t) \end{bmatrix} \tag{9}$$

Fig. 19. Representación matricial para ruedas.

Luego de conocer todas las variables involucradas, es necesario comprender las restricciones de las ruedas. Como ya se había comentado anteriormente estas restricciones se dividen en dos tipos, rodamiento y deslizamiento. La primera se puede observar en la Fig. 20, la cual obliga a que la cantidad de movimiento a lo largo de la dirección del plano de la rueda izquierda o derecha, sea igual al giro de la rueda sobre su eje horizontal, esto con el fin de obtener un puro rodamiento en el punto del contacto sin deslizamiento lateral, ver Fig. 20.

$$[sin(\alpha+\beta),\ -cos(\alpha+\beta),\ (-L)*cos(\beta)]*R(\Theta)\dot{\xi} - R\dot{\varphi} = 0 \tag{10}$$

Fig. 20. Restricción de rodamiento

La forma simplificada de manera matricial es la siguiente, en la cual (J1(s) es la matriz de restricciones, (R(Θ) matriz de rotación, (ξ punto) matriz de velocidades del marco de referencia global, (J2) matriz de radios la cual contiene los valores de cada radio de las llantas y tiene como dimensión el número de ruedas (NR) x número de ruedas (NR) y por último las velocidades (φ punto) del robot, ver ecuación (11) (Fig. 21).

$$J1(\beta_s)R(\theta)\dot{\xi}_I - J2\dot{\varphi}(t) = 0$$

(11)

Fig. 21.Simplificación de ecuación 10

La segunda restricción (deslizamiento) plantea que no haya ningún deslizamiento lateral, lo cual representa que la rueda no debe resbalar ortogonalmente al plano de la llanta, forzando a que el componente del movimiento ortogonal de la rueda sea igual a cero, ver Fig. 22.

$$[cos(\alpha+\beta),\ \ sin(\alpha+\beta),\ Lsen(\beta)] * R(\Theta)\dot{\xi} = 0$$

(12)

Fig. 22. Restricción de rodadura.

5.1.3.5 Cinemática inversa

Es la metodología matemática para obtener el valor de las velocidades y ángulos de dirección desea dos en las ruedas, con el fin de lograr una velocidad del origen local en el global. De acuerdo a la ecuación (13)(Fig. 23) se puede despejar la variable $\dot{\varphi}$

$$\dot{\varphi} = J_1(\beta_S)\, R(\theta)\, \dot{\xi}_I\, J_2^{-1} \qquad \textbf{(13)}$$

Fig. 23. Velocidad de las ruedas .

5.1.4 Percepción robótica.

La percepción robótica es la encargada de capturar, procesar y generar una orden de respuesta a otro subsistema en base a información. Esta información es captada por los sensores que son fundamentales, ya que tienen la capacidad de aportar información tanto del entorno de trabajo como del estado interno del robot, con el objetivo de conocer cómo se encuentra y donde se encuentra. Por ejemplo, los parámetros internos tales como posición, velocidad o ángulo de giro de las ruedas o las articulaciones, son muy necesarios en la mayoría de las aplicaciones.

Por otro lado, la información que es captada por sensores en la estructura del robot, se denominan sensores internos (propioceptivos). Por el contrario, los sensores externos (exteroceptivos) están fuera de la estructura del robot y permiten obtener información del entorno.

Adicionalmente, el sistema sensorial en los seres humanos, animales y plantas está supremamente desarrollado, generando la información necesaria que le permite a otro sistema llevar a cabo muchas acciones, como por ejemplo ingerir un alimento o desplazarse.

Es por esa razón, que en los sistemas robóticos la correcta selección de los sensores, permitirá desarrollar tareas concretas y según los criterios del diseñador como fortalecer ciertas características que sean deseadas con el menor costo tanto económico como computacional.

5.1.4.1 Sensores

El sensor es uno de los componentes fundamentales de todo robot, máquina o cualquier dispositivo moderno que pueda realizar una realimentación de un estado. Un tipo de sensor muy característico en la ejecución de tareas que dependan del entorno es el transductor, ya que detecta una magnitud física y es capaz de convertirla en una señal eléctrica, la cual puede ser interpretada, procesada, almacenada y posteriormente de ser necesario, tomar una decisión basada en su información.

Características de los sensores.

- **Fiabilidad**: esta característica es directamente relacionada con el ambiente en el cual se desplaza la plataforma robótica. Para un ambiente estático se espera que no existan variaciones en el sensor y por consecuencia se pueda confiar en ellos. Pero como es habitual, existen cambios que deben ser compensados, ya sea por circuitos o métodos estadísticos que arrojen el mínimo margen de error, lo cual incrementa la carga computacional.

- **Rango de valores:** son los valores mínimos y máximos que arroja el sensor

- **Precisión:** es la cantidad de valores que puede entregar el sensor en un rango de valores.

- **Velocidad de muestreo:** es la cantidad de datos que puede obtener un sistema en un determinado tiempo (frecuencia de toma de datos). Con una velocidad de muestreo mayor se puede conocer con mejor exactitud el estado actual del robot. Además, es recomendable que los sensores empleados tenga una mayor frecuencia de muestreo que la capacidad de procesamiento. Por otra parte, la velocidad de muestreo(VM) en múltiples sistemas puede ser alterada por medio de código fuente, un claro ejemplo es el sistema operativo robótico (ROS), el cual permite modificar la velocidad de muestreo de la emisión y recepción de datos contenidos en los nodos (scripts de código) del mismo. Cabe resaltar que la VM debe ser igual en los nodos que ejecutan una comunicación entre ellos, esto permite que haya una sinergia perfecta entre sensores y actuadores. No obstante, si la VM es diferente podría afectar el correcto funcionamiento del sistema.

Clasificación de sensores.

- **Sensores exteroceptivos:** son aquellos que miden el estado del entorno e interacciones con el robot. Un ejemplo es la nube de puntos generada por un sensor Lidar. (ver Fig. 24)

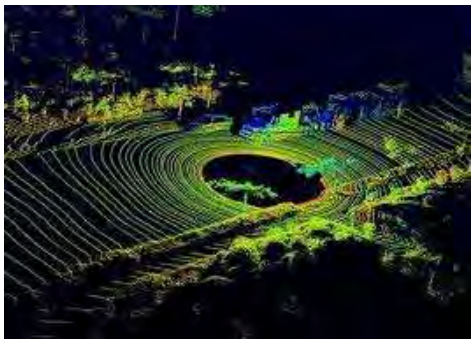


Fig. 24. Nube de puntos. [17]

- **Sensores propioceptivos:** son aquellos que miden el estado interno del propio robot. Por ejemplo, los encoders que se encuentran en un brazo robótico. (ver Fig. 25)

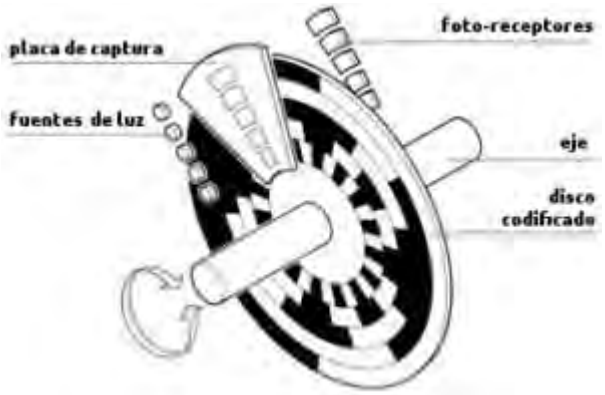


Fig. 25. Enconder. [18]

- **Sensores pasivos:** este tipo de sensores generan señales por medio de una fuente auxiliar, ya sea resistivo, capacitivo o inductivo.

- **Sensores activos:** los sensores activos generan señales de forma autónoma sin requerir una fuente auxiliar.

5.1.4.2 Señales

- **Señal binaria:** corresponde a una señal que solo contiene dos estados (ver Fig. 26), los cuales normalmente se referencian como alto o bajo y que corresponden a el valor máximo de la fuente o el valor máximo admisible por el procesador y un valor mínimo que normalmente corresponde a el valor de GND o tierra del circuito.

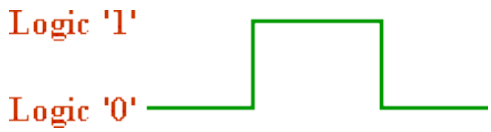


Fig. 26. Señal binaria. [19]

- **Señal digital:** este tipo de señal es muy usada en sensores que requieren suministrar considerables cantidades de información , la cual viene codificada en pulsos o sistemas BCD, binario, etc. (ver Fig. 27)

Fig. 27. Señal digital.[20]

- Señal analógica. la señal analógica viene representada ya sea en rangos de corriente o voltaje y es habitualmente usada en aplicaciones industriales. (ver Fig. 28)

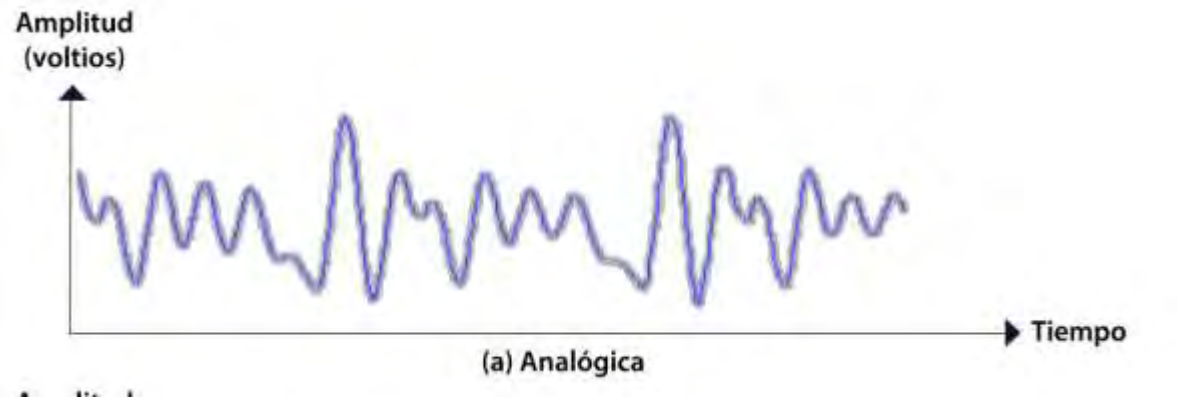


Fig. 28. Señal analógica. [21]

5.1.5 Actuadores

Un actuador es un dispositivo mecánico cuya función es proporcionar fuerza para desplazar a otro dispositivo mecánico. La fuerza que provoca el actuador proviene de tres fuentes posibles: presión neumática, presión hidráulica, y fuerza motriz eléctrica (motor eléctrico o solenoide). Para el caso de robótica, el actuador más común es el motor eléctrico, que cuenta con diferentes clases en las cuales cada una tiene sus respectivas ventajas y desventajas.

5.1.5.1 Servomotor

El servomotor es un dispositivo que permite controlar con precisión la velocidad, el par motor y la posición, ya que cuenta en su interior con un encoder decodificador de giros mecánicos en pulsos eléctricos. Además de esto, contiene un circuito de control que constantemente censa su posición angular o velocidad actual y genera esfuerzos de control para llevar el actuador a un punto de equilibrio (error = 0). (ver Fig. 29 correspondiente al diagrama de bloque de un servomotor)

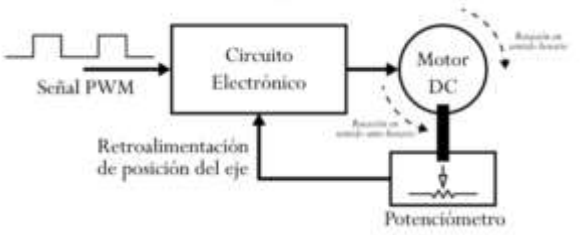


Fig. 29. Diagrama de bloque del servomotor.[22]

5.1.5.2 Motor paso a paso

El motor de pasos es un dispositivo que convierte las señales provenientes de un controlador, el cual conmuta el paso de corriente entre las bobinas que envuelven el motor en su interior. Este motor presenta las ventajas de tener precisión y repetitividad en cuanto al posicionamiento. (ver Fig. 30 correspondiente a conFig.ción interna de motor paso a paso)

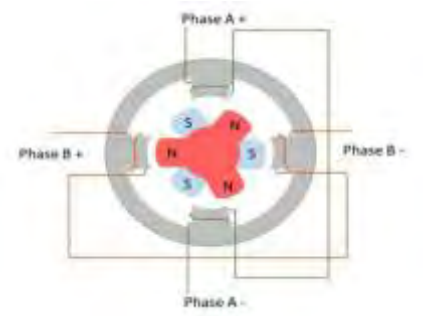


Fig. 30. Motor paso a paso. [23]

5.2 LENGUAJES DE PROGRAMACIÓN EN ROBOTICA

5.2.1 Python

Python es uno de los lenguajes de programación de alto nivel más usado en la actualidad, el cual al ser interpretado tiene como filosofía la legibilidad del código. Además, es desarrollado como un proyecto de código abierto y libre, administrado por la empresa Python software foundation. También, este lenguaje ofrece muchas ventajas, una de las más características es el desarrollo de aplicaciones en un tiempo relativamente corto.

5.2.2 C++

C + + es un lenguaje que lleva muchos años en el mundo y tuvo como intención inicial extender el uso del lenguaje base C. Este lenguaje permite la programación orientada a objetos de forma fácil y práctica, por lo cual es muy cómodo crear comunicaciones entre clases que necesiten de atributos o valores. Además, este otorga como característica principal ser un lenguaje híbrido. Asimismo C++ es flexible y estructurado debido a que este presenta las ventajas que poseen los lenguajes de alto y bajo nivel.

5.3 ENTORNOS DE DESARROLLO Y PROGRAMACIÓN

Para este proyecto el entorno de desarrollo a emplear, es el sistema operativo robotico ROS (ver Fig. 31 el logo), el cual es un framework para el desarrollo de software en robots. Durante varios años ha progresado y madurado como una herramienta gracias a su “opensource”, lo que ha permitido a su gran comunidad el desarrollo continuo de nuevas aplicaciones, potenciando sus capacidades

tecnológicas, mejorando su software. Además, permitiendo un mejor enfoque en la robótica móvil y finalmente, brindando una sólida base de partida para muchas personas que se dedican tanto a la industria como a la investigación y desarrollo.

Por otro lado, gracias a la forma como trabaja este sistema operativo, permite una modularidad en el desarrollo y una fácil integración de sus componentes, lo que conlleva a una considerable reducción en los tiempos de desarrollo, corrección de errores, tiempos y facilidad de simulación al no contar con el hardware definitivo entre algunas de sus ventajas.

Fig. 31. Ros.[24]

5.3.1 Conceptos básicos

5.3.1.1 Nodo

Un nodo es un archivo de programación ejecutable en sistema operativo robótico (ROS), usualmente utiliza lenguajes de programación conocidos mundialmente como Python, C++ o simplemente basados en C. Estos nodos permiten la fácil y cómoda emisión y recepción de datos de un nodo a otro, siempre que estos trabajen en ROS.

5.3.1.2 Mensaje

Los mensajes son tipos de datos que se utilizan durante una suscripción o publicación en un tópico especifico contenidos en nodos.

5.3.1.3 Tópico

El “topic” o tópico es el medio por el cual un nodo publica o recibe un mensaje.

5.3.1.4 Master

El master es el encargado de asignar la nomenclatura y registro a los nodos que se están ejecutando en ese momento. Su funcionamiento es fundamental y obligatorio ya que permite que los nodos (individuales) se ubiquen y comuniquen entre sí.

5.4 RED NEURONAL CONVOLUCIONAL

Una red neuronal convolucional o CNN, por sus siglas en inglés, es una red neuronal artificial, la cual tiene la facultad de identificar múltiples características en imágenes y clasificarlas. Cuenta con múltiples neuronas conectadas entre sí cuyas conexiones poseen diferentes pesos y bias. Esta red está compuesta por dos capas, las cuales determinan las características de cada imagen. La primera consiste en un extractor de características el cual se compone por dos capas, convolucional y submuestreo, mientras que la segunda está compuesta por un mapa de características, el cual contiene todas las propiedades numéricas de la imagen tratada, este mapa se determina con el producto matricial entre la imagen y un filtro “kernel”.

6. METODOLOGIA

Para plantear el desarrollo del sistema de localización, el primer factor a tener en cuenta es contar con una base científica sólida, en la cual se pueda basar la propuesta teniendo en cuenta los objetivos a los cuales está dirigido, debido a que pretende ser la base de futuros desarrollos. Por lo tanto, se deben considerar las restricciones y requerimientos de la plataforma en que se implementa.

La técnica en la que se basa la investigación, se orienta a la exhaustiva recolección de sistemas existentes, su correspondiente caracterización, evaluación teórica de ventajas, desventajas, restricciones y requerimientos que tienen. De igual manera, una identificación del hardware que requerirá para su futura implementación y finalmente se procederá con el desarrollo en un simulador.

6.1 ETAPA 1: REVISION DEL ESTADO DEL ARTE

6.1.1 Actividad 1: sistemas de localización más comunes

Los sistemas de localización más comunes en la Robótica dependen del entorno de trabajo. Durante la investigación, se identificaron como los más comunes, los sistemas de localización en exteriores, los cuales permiten un rango de error mayor. En consecuencia, son utilizados en ambientes sin un tamaño establecido, por lo tanto obtienen la característica de cambiar de ambiente con menor dificultad. En este caso encontramos el sistema basado en el sensor GPS.

6.1.1.1 Gps

El sistema de posicionamiento global (GPS), es una tecnología que utiliza satélites en órbita (ver Fig. 32) para enviar información sobre el posicionamiento de un objetivo en la superficie terrestre. El GPS recibe información por medio de un receptor, el cual determina el tiempo que toma a la señal en realizar el viaje entre el satélite y el GPS, con esto el dispositivo es capaz de calcular la distancia que hay entre el GPS y el satélite, además el receptor necesita la conexión de tres o más satélites para generar una posición exacta del objeto mediante la triangulación de diferentes señales.

Fig. 32. Red de satélites [25]

La tecnología del GPS es usada para muchas actividades, por ejemplo la localización de sistemas móviles (carros, motos, robots móviles), personas (exploradores), seguimiento de objetivos, turismo, etc. Estas actividades requieren un GPS debido a que recorren grandes distancias y su posición y localización son esenciales para ser monitoreadas. Sin embargo, el GPS es una solución inviable para el proyecto actual debido a que el sistema robótico móvil no va a recorrer grandes distancias y además se requiere una alta precisión en entornos cerrados. Por lo tanto, la efectividad del GPS es muy baja y además, el sistema robótico va a operar en entornos subterráneos donde la señal del GPS se ve seriamente comprometida.

Luego se encuentran los sistemas de localización que se pueden utilizar tanto en exteriores como en interiores, como lo son el LIDAR, la cámara estéreo y el imu.

6.1.1.2 IMU

El imu (unidad de medición inercial) es un dispositivo electrónico que puede obtener mediciones de aceleración, orientación y fuerzas gravitacionales.

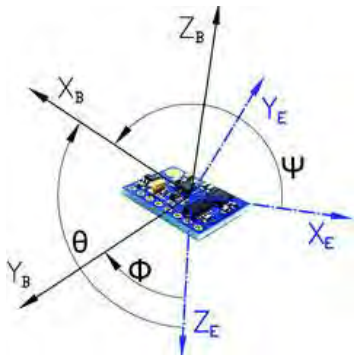


Fig. 33. Unidad de medición inercial [26]

El nombre por el que se conoce este tipo de sensor es IMU y por sí solo, es la base de un sistema de localización muy utilizado. Además, su funcionamiento se basa en la detección de cambios en el cabeceo (pich), alabeo (roll) y giñada(yaw) (ver Fig. 33 y 34 los ejes de un IMU), el cual contiene tres acelerómetros y tres giroscopios ubicados de manera ortogonal en cada eje.

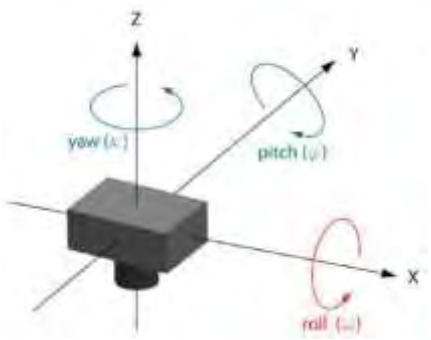


Fig. 34. Ejes [27]

Algunos IMU's además poseen un magnetómetro, el cual se encarga de medir el campo magnético de la tierra y así determinar el norte, sur, este y oeste.

El objetivo de usar acelerómetros es medir la fuerza G o también conocida como aceleración inercial y el giroscopio permite medir la posición rotacional respecto a un eje.

6.1.1.3 Lidar

Es un sensor muy utilizado en la robótica para localizar en un mapa o para hacer SLAM, el cual consiste en un haz de luz que es emitido de manera ininterrumpida en su entorno que al rebotar en una superficie retorna al sensor, el cual mide el tiempo que dura en hacer esta acción, lo que permite aproximar la distancia. A continuación se puede observar un ejemplo de cómo se visualiza en un entorno virtual los valores que arroja el sensor. (ver Fig. 24 y 35 distintas nubes de puntos en entornos externos)

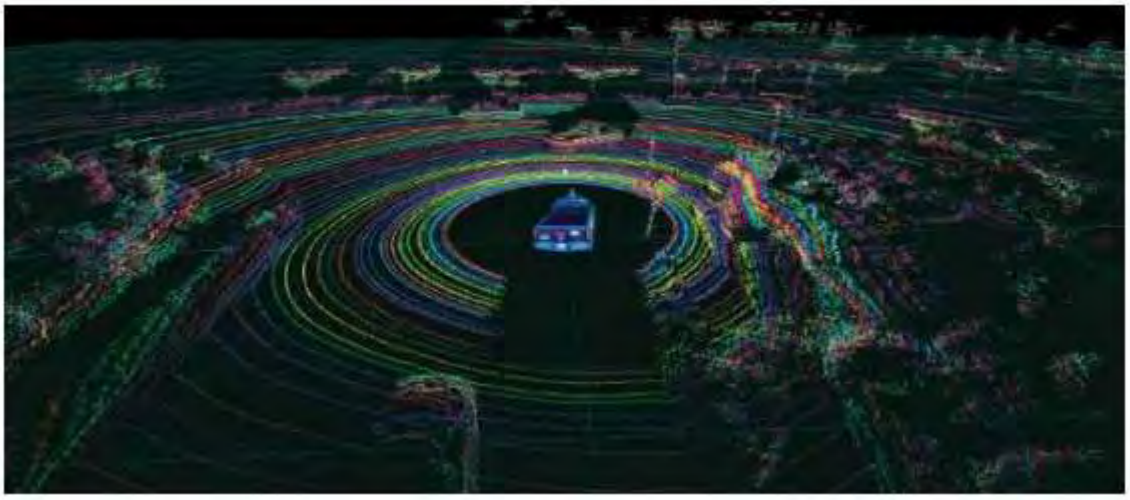


Fig. 35. Nube de puntos generada por lidar [28]

6.1.1.4 Cámara estéreo

La cámara estéreo es un dispositivo que tiene como característica principal, la detección de profundidad y rastreo de movimiento. Esta cámara es ideal para ser implementada en aplicaciones de robótica móvil. Este dispositivo de alta tecnología, se basa en la visión estero humana (ver Fig. 36). Gracias a su tecnología de odometría visual, hace seguimiento de movimiento en un espacio tridimensional, obteniendo una posición y orientación con precisión milimétrica. Además no se necesitan marcadores o sensores externos.

Fig. 36. Cámara estéreo zed [29]

Por último, para interiores se encuentra el RFID y la localización por visión artificial, los cuales requieren una infraestructura previa y un hardware de mayor capacidad, tanto en cámara estéreo, LIDAR como en la visión artificial.

6.1.1.5 Rfid

El sistema de identificación por radiofrecuencia (RFID), es una tecnología que consta de dos componentes, emisor y receptor. Este último, al ser estimulado por una onda de radiofrecuencia emite una señal que puede ser leída por un lector de datos (Fig. 37), este al obtener la señal de la etiqueta puede conseguir la

información almacenada de la misma, ya sea el precio del producto o características del producto.

Por otra parte, el RFID puede localizar objetos de manera precisa por medio de etiquetas las cuales pueden ir adheridas a múltiples superficies (ver Fig. 38). Gracias a esto, los sistemas móviles pueden obtener de manera precisa la ubicación o estado en tiempo real. Sin embargo, este es un gran problema, ya que para obtener la ubicación exacta del objeto en un lugar muy extenso se necesitan demasiadas etiquetas para su localización, por lo que genera un costo elevado.

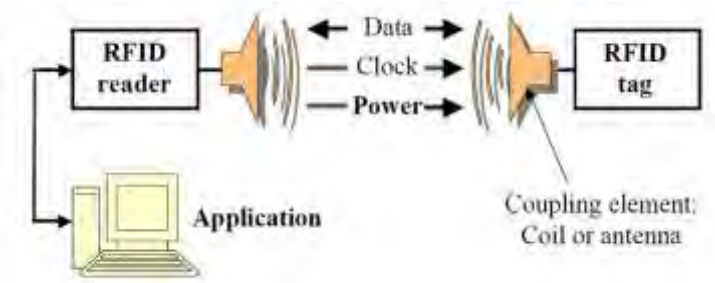


Fig. 37. Lector de RFID. [30]

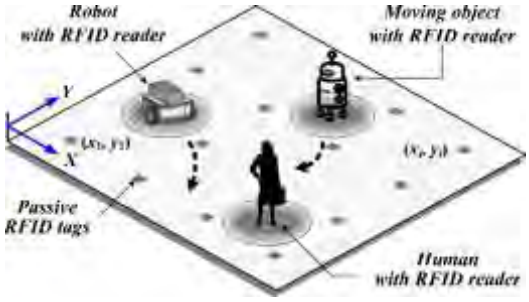


Fig. 38. Sistema de localización por RFID [31]

6.1.1.6 Visión artificial (sistema de percepción estático)

El sistema de percepción visual, es una tecnología que puede localizar objetos de manera precisa por medio de la estimación visual de la distancia que hay entre el objeto a localizar y un observador estático. Habitualmente una cámara con esta distancia puede determinar la posición bidimensional del objeto (marco local) con respecto a la cámara, lo que nos permite posteriormente realizar la transformación de esta ubicación dentro de un marco global. (ver Fig. 39)

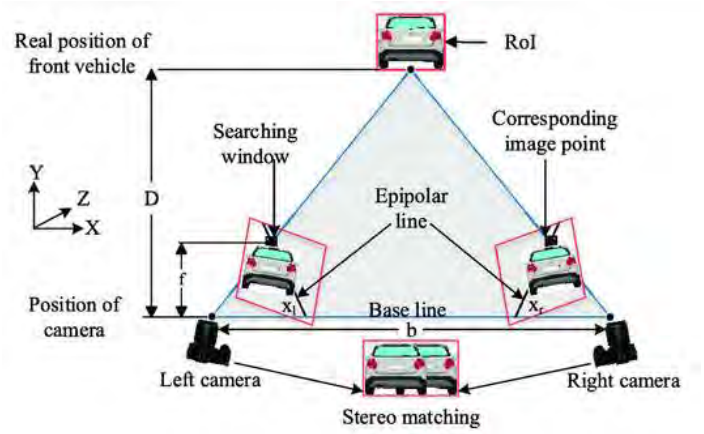


Fig. 39. Sistema de localización por visión artificial [32]

Los sistemas de estimación visual representan una alternativa válida. Sin embargo, los costos de implementación en términos de equipos de adquisición de imagen y video, son demasiado altos. Además, estos sistemas requieren de una gran capacidad computacional para poder procesar los datos adquiridos.

6.1.2 Actividad 2: integración de sistemas de localización más usados

6.1.2.1 Mapeo

Este consiste en la generación de un mapa digital del entorno en el que el robot se encuentra, con el fin de generar un sentido en la direccionalidad del robot. Con esta técnica, es posible la generación óptima de rutas y trayectorias que el robot seguirá acorde a lo planeado previamente.

Por otro lado, esta técnica se presenta de diversas maneras siendo capaz de realizar el planteamiento de mapas en dos dimensiones (2D) o en tres dimensiones (3D).

Para realizar este procedimiento existen varias aproximaciones basadas en diferentes técnicas. En este caso, se hablará dos de ellas: SLAM y Occupancy Grids.(ver Fig. 40)

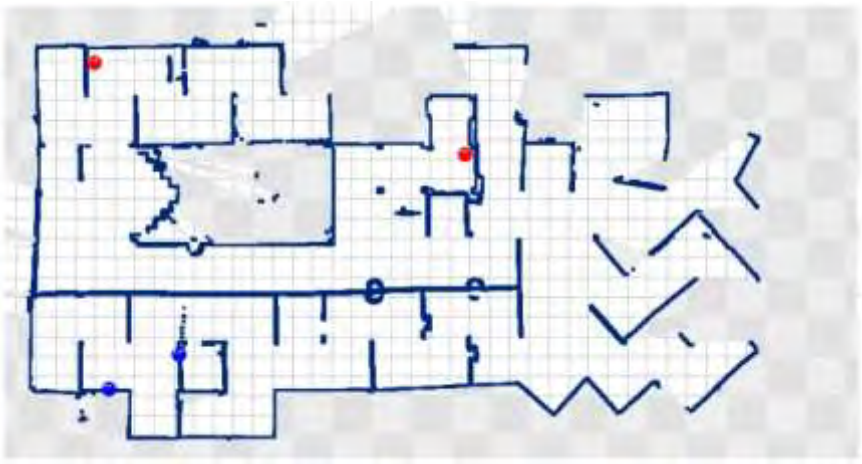


Fig. 40. Ejemplo de mapeo robótico [33]

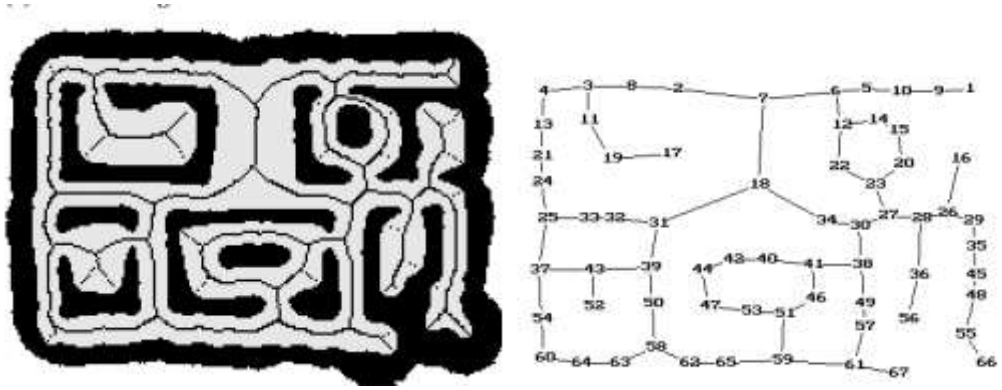
6.1.2.2 Occupancy grids

La idea básica de este método de mapeo, consiste en el desarrollo de mapas por medio de Grillas (o una cuadrícula), las cuales poseen un espacio equitativo y binario de variables aleatorias que representan la presencia de un obstáculo en una ubicación del ambiente en el que el robot se encuentra. De acuerdo a esto, el algoritmo realiza el cómputo respectivo para estimar estas variables en base a las condiciones iniciales y el input del (los) sensor(es). Por lo tanto, este método se refiere a la familia de algoritmos computacionales basados en probabilística.

Adicionalmente, este método fue creado por H. Moravec y A. Elfes en 1985 pensando en el problema de la generación de mapas por medio de sensores con ruido y de poca confiabilidad a la hora de realizar mediciones, tomando como requerimiento el hecho de que se conoce la pose inicial del robot.

Integración de grillas de ocupación y mapas topológicos para la navegación de la robótica móvil: este es un artículo presentado en la conferencia nacional de inteligencia artificial de Estados Unidos *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, desarrollado por Sebastian Thrun y Arno Bücken, el cual describe la aproximación que integra ambos paradigmas.

Ver Fig. 41 A la grilla de ocupación y en la Fig. 41 B el mapa topológico



A **B**

Fig. 41 Grillas de ocupación y mapas topológicos [34]

En este caso, los mapas basados en grillas son aprendidos usando redes neuronales artificiales e integración Bayesiana. Además, los mapas topológicos son generados encima de las grillas de manera que se haga una partición de estos en regiones coherentes. Por otro lado, combinando ambos paradigmas, la aproximación presentada gana lo mejor de ambos mundos: precisión, consistencia y eficiencia.

6.1.2.3 Slam

SLAM es el acrónimo que viene de *"Simultaneous localization and mapping"*, por sus siglas en inglés. Se refiere a la capacidad del robot de realizar un mapa de su entorno a la vez que este va identificando aspectos clave del ambiente en el que se encuentra, logrando de esta manera ubicarse con referencia a el lugar en donde empezó a desarrollarse el mapa. (ver Fig. 41)

En realidad, el SLAM no se refiere a un método como tal de uso para realizar un mapa digital, es más bien descrita como una técnica la cual puede ser implementada en robots móviles para que realicen un mapeo del entorno de forma autónoma pudiendo así volver a donde empezaron. Teniendo esto en cuenta, el SLAM es implementado en diferentes aplicaciones, siendo una de las principales el mapeo de entornos no conocidos previamente como lo puede ser la superficie terrestre, el océano, el interior del cuerpo humano o incluso puede ser usado en tecnologías que se encuentran en desarrollo para aplicaciones interplanetarias.

Por otra parte, dada la complejidad de esta técnica, existen numerosos algoritmos que tratan de resolverla. Algunas de los métodos de aproximación, consisten en la inclusión de filtros como el filtro de Kalman, el cual se basa en la unificación multisensorial para una estimación más acertada de la localización del robot y él mismo ha sido caso de varios estudios y avances a nivel matemático para el desarrollo de tecnologías que tenemos hoy en día. Otra aproximación, está basada en filtros de partículas la cual trata de usar el mismo concepto de Kalman, pero usando algunas opciones distintas, tratando así de realizar esta estimación por medio de cálculos más avanzados y robustos, pero igualmente más costosos a nivel computacional.

Los escaneos tomados en cada pose se alinean de acuerdo con su estimación de pose para formar un mapa global ver Fig. 42

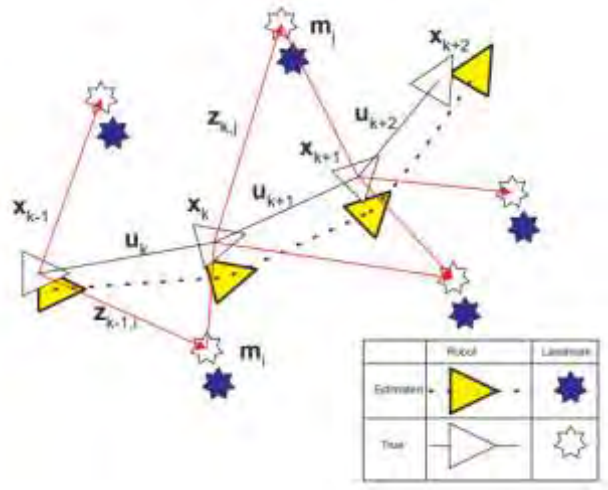


Fig. 42. SLAM basado en trayectoria. [35]

6.1.2.4 Representación basada en características

La visión por computador es un campo muy importante en la actualidad gracias a sus amplias aplicaciones en las que está siendo implementado, la representación del conocimiento y más específicamente el basado en características tiene como uno de sus objetivos, el facilitar la inferencia a partir de un conocimiento previo.

6.1.2.5 Odometría

Es la forma o mecanismo empleado para estimar (y no determinar) la forma aproximada de la posición del robot, la cual consiste en incorporar una serie de sensores que den como resultado información de su desplazamiento y orientación (posición relativa). Sin importar la metodología implementada se genera una acumulado de error, el cual se le aplican diferentes técnicas de procesamiento para reducirlo a su mínima expresión y obtener el valor más exacto posible, además de tener la idea fundamental de integrar información incremental a lo largo del tiempo. (ver Fig. 43)

Adicionalmente, los terrores que existen en la odometría son de dos tipos, uno de ellos es el sistemático, el cual se presenta cuando el mecanismo de la toma de datos presenta problemas, ya sea por la tasa de muestreo, mal alineamiento de ruedas entre muchas otras, y el segundo tipo es el no sistemático, el cual está relacionado directamente a hechos causales como deslizamiento de rueda en el terreno, fuerzas externas que secuestren o manipulen el robot y el más común como es el desnivel.

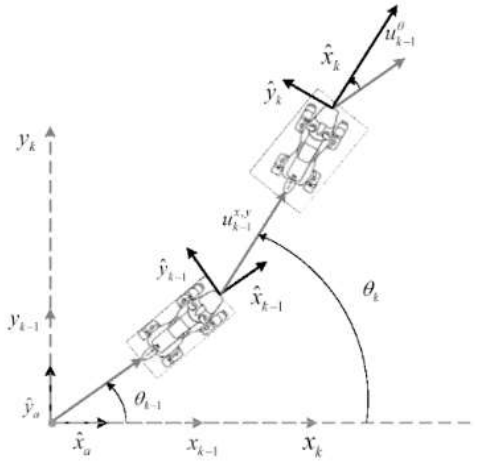


Fig. 43. Modelo de movimiento basado en odometría. [36]

6.2 ETAPA 2: SELECCIÓN DE LOS SISTEMAS DE LOCALIZACION METRICA Y TOPOLOGICA A IMPLEMENTAR

6.2.1 Actividad 3: sistemas de localización más adecuado a implementar

El plan de localización métrica a desarrollar en este proyecto, corresponde a un sistema de estimación 3D basado en mediciones de poses, el cual emplea un filtro extendido de Kalman que combina mediciones provenientes de diferentes fuentes como lo es el sensor IMU y la odometría del robot. Esto se puede observar a grandes rasgos en la Fig. 44.

Por otro lado, se seleccionó este sistema de localización debido a que se evidenció en varios proyectos de investigación, excelentes resultados al aplicarlo. En dichos proyectos se identificó el uso de conFig.ciones similares de sensores debido a que su operación era en ambientes semi estructurados como el sótano de la Universidad Autónoma de Occidente, de manera inicial. Cabe resaltar que dicho entorno de operación no presenta variaciones en el terreno.

Fig. 44. Fusión de sensores. [37]

El plan de localización topológica consiste en emplear un sistema de visión artificial basado en redes neuronales convolucionales, la cual se entrena con imágenes del entorno con su correspondiente clasificación de cada elemento que compone la zona donde opera el robot.

Una forma de entender gráficamente como es la red neuronal convolucional, es mediante la Fig. 45.

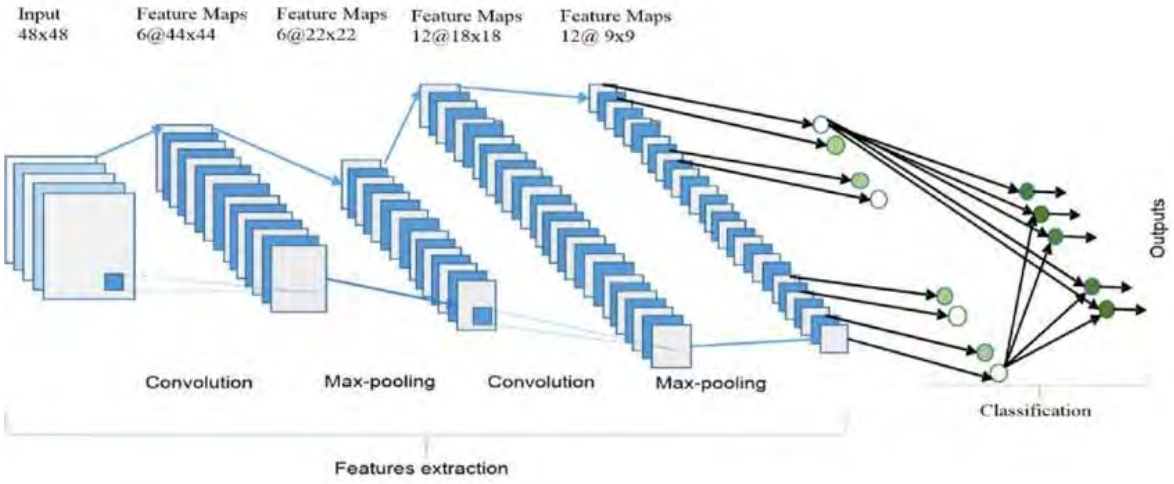


Fig. 45. Red neuronal artificial convolucional. [38]

Como se evidencia en la Fig. anterior (Fig. 45), se expone el funcionamiento de una red neuronal convolucional, la cual está constituida por capas de max-pooling (determina el valor máximo dentro de una sección de la imagen) y convoluciones (determina características específicas de la imagen).

6.2.2 Actividad 4: revisión y selección de sensores

Para el desarrollo del sistema de localización métrico, se requiere el uso de sensores que permiten la adquisición de variables como velocidad, aceleración y orientación. De acuerdo a esto, un sensor que cumple con dicho requerimiento es el IMU. Sin embargo, este sensor posee múltiples variantes en el mercado, las cuales se diferencian en la adquisición de variables adicionales. Por lo cual, se enlistan a continuación las alternativas del IMU y sus respectivas especificaciones.

Sensor de medición inercial. Para elegir de manera correcta el IMU, existen diferentes factores a tener en cuenta, entre los más importantes se encuentran el rendimiento, grados de libertad, resolución, consumo y costo. Además, en aplicaciones con situaciones adversas, las vibraciones pueden alcanzar un alto nivel y es necesario tener en cuenta esto al momento de su selección, ya que existen algunas alternativas que minimizan esto. Por esta razón, es necesario evaluar algunas alternativas de sensores comerciales en Colombia.

Tabla I

Selección de IMU

	VOLTAJE	REGULADOR DE VOLTAJE INTEGRADO	CONSUMO ENERGÉTICO	GRADOS DE LIBERTAD	RESOLUCIÓN	PRECIO
Shield imu 6dof DFR0209	3.3 o 5	Conversor de nivel lógico	<0.5mA	6	13 bits	150.000
Imu 9dof Gy-85	3.6 a 9	SI	<1mA	9	12 bits	52.000
Imu 10dof Gy-86	3.3 o 5	Conversor de nivel lógico	<0.5mA	10	13 bits	70.000
Imu 9dof MPU9250	3.3 a 5.5	SI	<0.5mA	9	13 bits	22.000
Imu 6dof MPU6050	3.3 a 5.5	SI	<1mA	6	10 bits	10.000
Imu 3dof ADXL345	3.3 o 5	Conversor de nivel lógico	<0.4mA	3	10 bits	11.000
Imu 10dof GY-91	3.3 a 5	SI	<1mA	10	16 bits	25.000

Nota: se presenta las características de los sensores de referencia IMU con sus respectivos valores.

Después de clasificar cuáles son los sensores IMU, se procede a la selección dependiendo de las características mostradas anteriormente (tabla I).

El regulador de voltaje integrado es un factor muy importante en el proceso de selección, ya que al momento de estar completamente operativo el robot, es necesario que este sensor cuente con una fuente de alimentación estable y así evite errores en la medida o que se pueda estropear.

El segundo factor importante es los grados de libertad, Ya que con 9 grados de libertad se puede integrar el sistema actual con múltiples proyectos a futuro. Finalmente, el precio también es relevante, ya que si se desea replicar el proyecto es conveniente que sus costos sean lo más razonables y ajustados posibles.

Dicho lo anterior, los sensores a clasificar son en primer lugar el Imu 10dof GY-91 por su alta resolución, bajo precio y los 10 grados de libertad, el cual lo hace un candidato muy bueno y en segundo lugar, el Imu 9dof MPU9250 con sus 9 grados de libertad, resolución adecuada y bajo precio. En definitiva, se optó por usar el sensor MPU9250 por la disponibilidad con la que se cuenta.

Por otro lado, el sensor encargado de medir la rotación de las ruedas es el encoder, este es un codificador rotatorio el cual se encarga de generar pulsos digitales que son interpretados para convertirlos en una posición angular; en el caso de robots móviles se utilizan motores que poseen en su interior un sensor de este tipo. Por lo cual, se debe seleccionar adecuadamente el motor según las características del robot y sus especificaciones preliminares.

Para la selección del motor de tracción, se debe tener en cuenta la fuente de alimentación necesaria, la carga que el motor debe mover, la velocidad que requiere y el par esperado de él. Además, con estos datos se puede ver qué corriente máxima puede pedir y qué eficiencia se puede obtener.

Datos:

Carga: 20 Kg -estimada

Ruedas de diámetro: 15,24 cm

Velocidad deseada: 1 a 2 m / s (caminata promedio de una persona)

Número de motores: 2 (movimiento del robot guía)

Para calcular el par instantáneo, la fórmula:

$$\textbf{Torque=Fuerza x Distancia=20Kgx7.62cm=152.4kgcm} \quad (14)$$

Este par se divide en cuatro apoyos obteniendo 38.1 kg * cm de par constante o máximo, se utilizará un par de 25 57.12 kg * cm, ya que solo hay dos motores para el desplazamiento.

Para la velocidad se obtiene que:

$$\textbf{Perímetro= 2 x Pi * 7.62 cm = 47.88 cm} \quad (15)$$

En este caso el perímetro es igual a la distancia que recorre una rueda al girar, ahora la Velocidad es igual a distancia entre tiempo

En los motores la velocidad está en RPM (Revoluciones por Minuto) y la velocidad que se requiere es de 1 m/s a 2 m/s, que es la velocidad promedio de una persona al caminar suavemente, por lo tanto, esta velocidad se convierte en RPM.

$$\textbf{Velocidad=2m/s * 60s/min=120m/min} \quad (16)$$

Ahora se calcula cuántas vueltas tiene que dar el neumático para cubrir 60 metros, para ello convierta los 47,88 cm a metros y divida:

$$\textbf{120m/0,4788m=250.62RPM} \quad (17)$$

Por lo tanto, se requiere una velocidad entre 125,31 RPM y 250,62 RPM

Se obtuvieron los dos datos clave Torque Constante de 57.12 kg * cm y Velocidad de 250.62 RPM, ahora se busca motores que tengan estas características siempre tratando de estar al menos un 10% por encima de los datos estimados.

Tabla II

Motor datasheet selection

	DS04-NFC Servo 360	i00600 Torxis Servo	HERCULEX 0602
Precio (USD)	10	280	320
Load (Kg-cm)	8	115	77
Feedback	Angle	Angle	Angle, Torque, Temperature, Current
Voltage (V)	5V	12V	14.8V
Interface	PWM	PWM	Serial
Current(A)	1A	3A	1.5A
RPM	50	30	180
Encoder	10 bits	-	14 bits
Control	-	-	PID, Fuzzy
Protection	-	-	Current, Voltage
Programmable	-	Yes	Yes
Visual Alerts	-	-	Leds Multicolor

Los motores mostrados en la tabla anterior (tabla 2), cuentan con las características como lo son, la carga que deben soportar, el consumo energético y la realimentación de los mismos, lo cual es muy útil en etapas posteriores de proyectos que se implementen sobre esta plataforma.

Por esa razón, se seleccionó el motor de referencia herculex 0602 que cumple con las características ya mencionadas, además de un encoder de 14 bits, el cual se considera con una buena resolución para la aplicación en la que se desea trabajar.

Adicionalmente, para la selección del motor de dirección se debe tener en cuenta la carga a la que estaría sometido el motor. En este caso, luego hacer algunas pruebas, se determinó que la carga oscila entre 2 y 3 libras, la distancia es 7.62 cm y un ángulo de funcionamiento de 120 grados.

$$\textbf{Torque=Fuerza x Distancia=1.5Kgx7.62cm=11.43kgcm} \qquad \textbf{(18)}$$

Tabla III

Motor datasheet selection

	HERCULEX 0201	HERCULEX 0101	MG945 metal
Precio (USD)	140	40	20
Load (Kg-cm)	24	12	12
Feedback	Angle, Torque, Temperature, Current	Angle, Torque, Temperature, Current	------
Voltage (V)	7.4V	7.4V	5.5
Interface	Serial	Serial	pwm
Current(A)	1A	1A	1A
RPM	180	120	120
Encoder	14 bits	10 bits	10 bits
Control	PID, Fuzzy	PID, Fuzzy	---------
Protection	yes	yes	----------
Programmable	Current, Voltage	Current, Voltage	----------
Visual Alerts	Yes	Yes	----------

Según la tabla 3, la selección que conlleva mayor costo beneficio es el motor de referencia herculex 0101, ya que tiene la capacidad de cumplir a cabalidad con las especificaciones técnicas.

Posteriormente, para continuar con la siguiente etapa del proyecto, fue de vital importancia realizar un proceso de reacondicionamiento de la plataforma robótica asignada, ya que se encuentra en mal estado y es necesario conocer su morfología antes de realizar el sistema de percepción robótica. A continuación, se va a mostrar someramente el proceso realizado.

Reacondicionamiento de la plataforma asignada

Diagnóstico. El diagnóstico de la plataforma asignada es el robot móvil chimuelo y se dividió en dos frentes que fueron intervenidos de manera simultánea. Sin embargo, por la contingencia, este proceso se vio afectado y prolongado.

Diagnóstico mecánico

Eje de ruedas de tracción. En esta parte el robot cuenta con un eje de 13cm de aluminio con holgura en la rueda de 2mm y un tornillo prisionero, que debido a la carga que debe soportar en el eje, este se encuentra con un desgaste muy significativo generando un grave inconveniente en la tracción del robot y en su estabilidad, además de la falta de dos rodamientos en el buje.

Motores y ejes de dirección. El robot cuenta con dos motores de dirección los cuales en sus pruebas de funcionamiento arrojaron resultados negativos, dado que su sistema de comunicación con el microcontrolador se ve interrumpido. Además, al indagar porqué ocurre esto, se encuentra que debido a que fueron sometidos a voltajes por encima del admisible, se comprometió el sistema de comunicación. También, se logra evidenciar un desgaste muy significativo en los ejes diseñados en aluminio que causa una holgura y además de cuentan con un rodamiento pegado.

Sistema de estabilización. El sistema de estabilización que posee el robot es un desarrollo que tiene varios años y posee amplia presencia de óxido en sus tornillos y le hace falta un rodamiento. Además, en el buje donde se apoya las barras laterales con el sistema de estabilización, se presenta una holgura de 3 mm a la izquierda y 2 mm a la derecha. (ver Fig. 46)

Carcasa. La falta de una carcasa hace difícil mantener elementos al interior del robot y su apariencia estética se ve afectada por esto. (ver Fig. 47)

Fig. 46. Sistema de estabilización del robot.

En la Fig. 46, se puede observar cómo algunas piezas del sistema de estabilización del robot se encuentran oxidadas y como está expuesta sin ninguna protección.

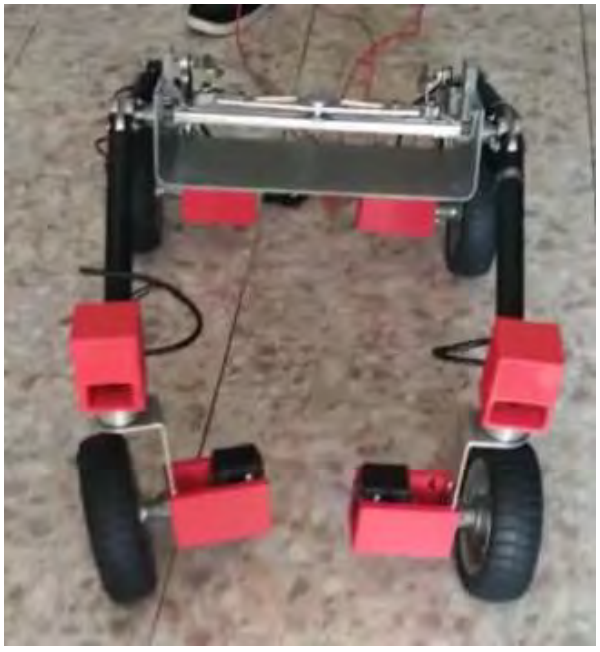


Fig. 47. Robot físico vista superior.

En la Fig. 47 se observa una vista del robot donde se evidencia que tiene defectos en la estructura debido a su curvatura.

Diagnóstico eléctrico. El robot no cuenta con una batería o fuente de alimentación, además, las conexiones eléctricas no tienen ningún manual que permita conocer cómo se llevaron a cabo estas, ya que no existe ninguna norma de colores o referencias. Adicionalmente, se encuentran expuestos los puertos de conexión de los motores. Finalmente, se cuenta con una caja donde se almacena el computador del robot y otros componentes electrónicos como lo son un conversor de nivel lógico, unos reguladores de voltaje y un Arduino, los cuales se encontraron quemados.

Fig. 48. Caja de circuitos expuestos .

En la Fig. 48 se puede observar cómo están expuestos sin ninguna protección los circuitos y cables del robot.

Conclusión. El robot se recibe en mal estado general, con tornillos rodados, partes oxidadas, una holgura en todas sus uniones, faltan rodamientos, los motores de dirección no funcionan, cables eléctricos expuestos sin protección alguna y una de las barras estaba torcida

Plan de acondicionamiento. El plan de mejoramiento consiste en tres fases, en las que se considera el uso del esqueleto actual, el diseño y la elaboración de las piezas mecánicas necesarias y el diseño de una carcasa más amigable con el usuario.

Fase 1: Correcciones mecánicas.

Recorte del eje de tracción. Con el fin de reducir el peso, se realiza el recorte del eje a solo 7cm, ya que se decide cambiar el material a acero por sus propiedades mecánicas, el cual le otorga una vida útil mayor. Por ende, se debe maquinar de nuevo.

Fig. 49. Caja de motores de tracción.

En la Fig. 49 se observa la caja de los motores de tracción (imagen izquierda) antes de recortar el eje y en la imagen derecha como quedo después de esto.

Sistema de protección contra sobrevoltajes. Para la protección de los componentes electrónicos se optó por utilizar una serie de reguladores de voltaje (ver Fig. 50)

Fig. 50 Regulador LM2596

Motores y ejes de dirección. Siguiendo el proceso de selección de motores llevado a cabo, se realiza el cambio de estos, lo que deriva en la inutilidad de los ejes de dirección actuales, así que con el objetivo de maximizar su vida útil se realiza el cambio de material a acero.

Fig. 51. Caja motores de dirección.

En la Fig. 51 se observa cómo es la parte interna de las nuevas cajas de dirección desarrolladas con el fin de mejorar el diseño del robot. Sin embargo, este problema puede ser solucionado definitivamente en un futuro con la llanta brushless hub motor (ver Fig. 52), la cual eliminaría la posibilidad de desgastes mecánicos en el eje de tracción.

Fig. 52. Motor brushless.

Sistema de estabilización. En este sistema se debe realizar el cambio de los tornillos, lubricado de algunos rodamientos y maquinado de dos anillos para los bujes que permitan asegurar los rodamientos y eliminar la holgura que posee actualmente.

Fig. 53. Sistema de estabilización.

En la Fig. 53, se observa el sistema de estabilización (Fig. izquierda) y cómo al interior de esta (Fig. derecha) existen holguras que afectan el desempeño del robot. Además, dado que los motores requieren cajas que los soporten y éstas quedan inútiles al haber recortado los ejes, se realiza el diseño CAD en el software SolidWorks de las nuevas cajas de los motores de tracción y motores de dirección.

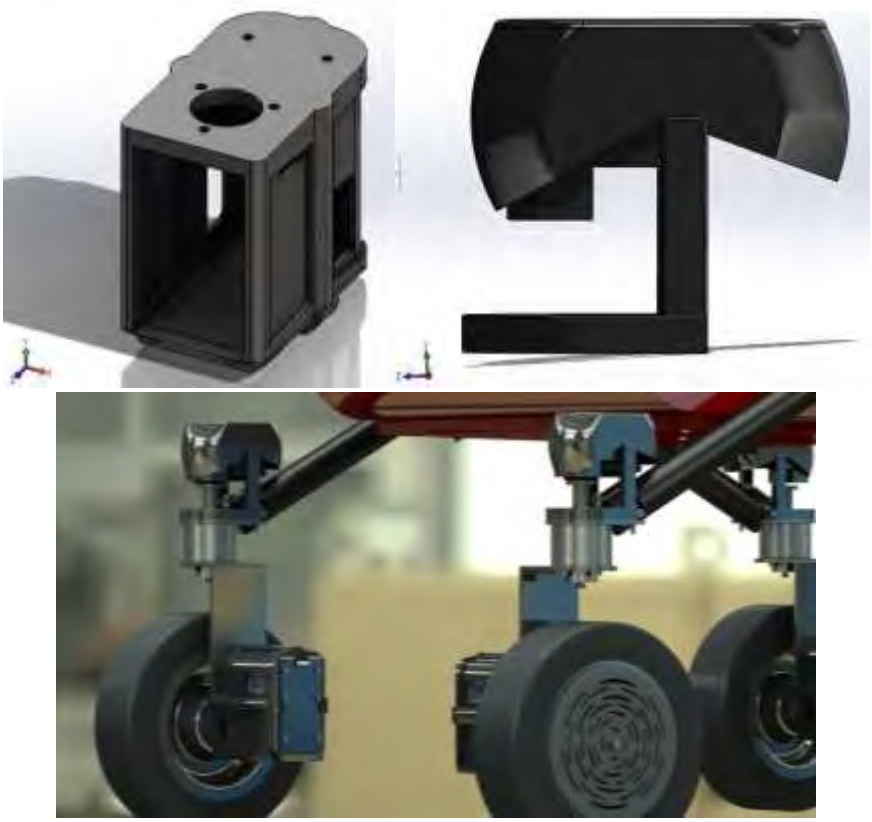


Fig. 54. Rediseño cajas de motores.

En la Fig. 54 (superior derecha), se puede observar el nuevo diseño correspondiente a las cajas de motor de tracción, luego se encontró un nuevo diseño de las cajas de motor de dirección (Fig. superior derecha) la cual debe ser evaluada después de realizar la impresión 3D y finalmente se estableció (Fig. inferior) como quedaría el modelo con las alternativas ya planteadas.

Fase 2: Correcciones eléctricas

Fuente de poder. Para calcular cuánto es la potencia requerida en la fuente de poder, se debe conocer cuánto es el consumo promedio de los elementos y así calcular el tiempo que se espera operar con el robot antes de tener que volver a conectarlo a un cargador.

Consumo a máxima potencia

Nvidia Jetson nano: 10W

Motores herculex 0601: 15W*2

Motores herculex 0201: 5W*2

Arduino due: 2W

Regulador bec pro: 3W

Regulador LM2595: 1W

Cámara Logitech c920: 3W

Total: 59W

Además, dado que la potencia máxima es 59W, se va a utilizar una batería de 6 celdas con capacidad de 10.000 mAh, lo que garantiza una operación a máxima potencia de 1 hora. Sin embargo, el consumo promedio de los elementos es menos de la mitad en las pruebas realizadas, por esta razón, el rango de operación esperado está entre 1 y 2 horas.

Cableado. En esta fase se diseña el sistema eléctrico según los requerimientos de operación del robot, gracias a que las barras laterales del robot están vacías al interior, se usaron como conducto para proteger los cables y evitar que se expongan o se vean.

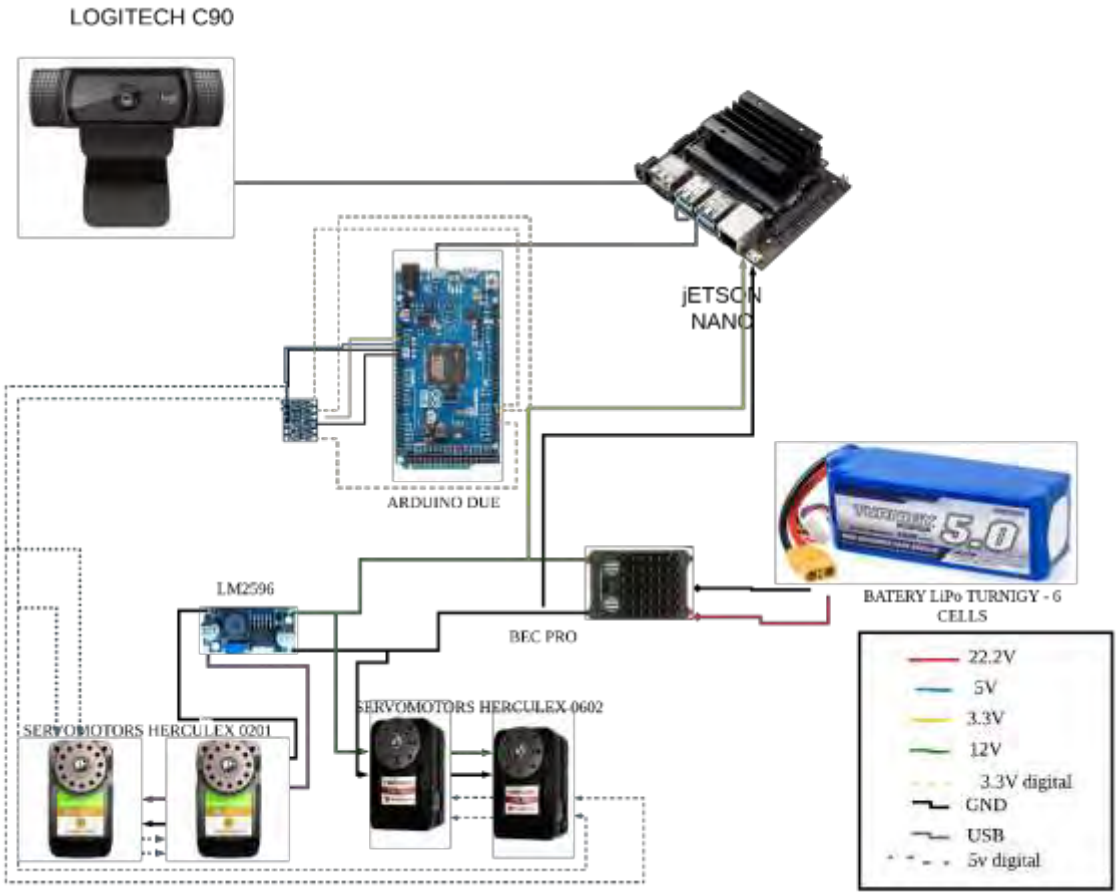


Fig. 55. Esquema de cableado.

En la Fig. 55 se encuentra el diagrama de conexiones eléctricas del robot y su conFig.ción en la parte inferior derecha.

Fase 3. Diseño de carcasa.

En el desarrollo de esta fase, se lleva a cabo una consulta de literatura sobre robots móviles y cómo es su morfología según su uso; dado que la futura aplicación del robot tiene mucha interacción con el público en general, se opta por un diseño que pueda generar empatía.

Adicionalmente, se realiza un análisis de diseño para crear una interfaz humana desde el robot hasta el usuario que se puede determinar tanto desde el punto de vista estético como funcional.

Estéticamente: El robot podría ofrecer una interacción hombre-máquina más fluida, amigable y directa, permitiendo al usuario sentir empatía con las expresiones mostradas a través de una matriz de leds blancos ubicados en la parte frontal del robot para darle personalidad mientras crea un punto de referencia local para denotar el sentido de movimiento y orientación que tendrá el robot móvil con respecto a un individuo y será reconocido por el usuario.

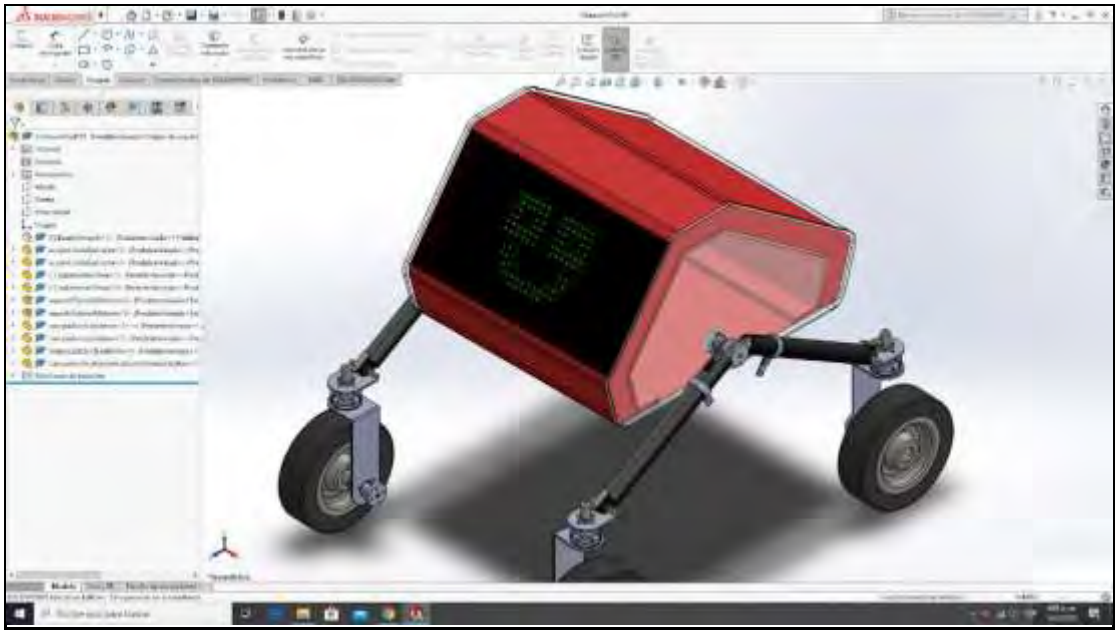


Fig. 56. CAD de robot terminado

Dentro de los acabados se utilizará material acrílico transparente de 5 mm, el cual se pintará en las caras internas para darle cuerpo a la carcasa, y la parte externa será traslúcida para dar efecto barrera y con esqueleto en acero inoxidable de 5 mm montado y atornillado a la estructura actual conservando las piezas del sistema de estabilización.

Los colores previstos son rojo mate para el exterior e interior de la carrocería, negro para la "cara" y translúcido en los laterales para observar el circuito interno. Cabe señalar que la pantalla negra será una película de vinilo negro, que permitirá que pase la luz de la matriz de LED.

Ver diseño exterior en interior en la Fig. 56, 57 y 58.

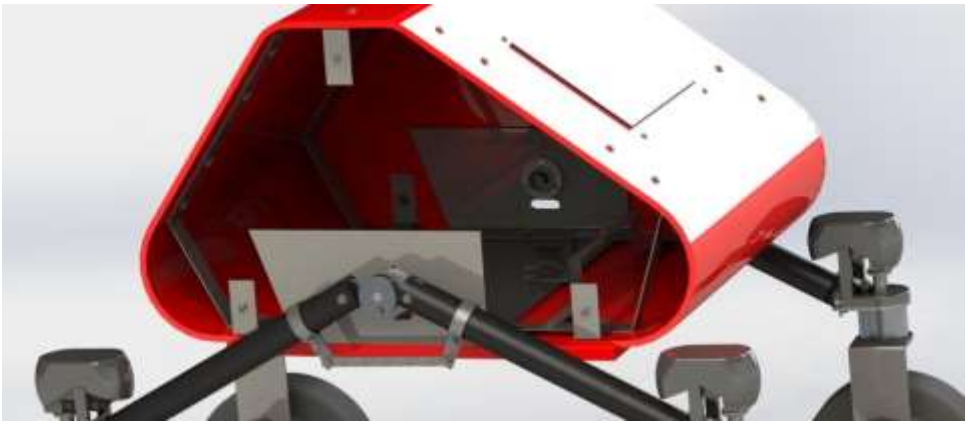


Fig. 57. CAD de interior del robot

Funcional: La ubicación espacial y direccional de la matriz LED para el robot guía móvil, se basa en los ángulos de movimiento que se realizan con la vista humana hacia un objeto de interés, obteniendo unos valores preliminares de las dimensiones y ángulos de visualización para la carcasa y matriz a utilizar. Como punto de partida, se estima los valores antropométricos promedio involucrados en los movimientos para enfocar la visión en un objeto y luego conocer los ángulos de confort de manera práctica usando SolidWorks, este dato es de mujeres y hombres jóvenes colombianos entre 20 y 39 años, esto se detalla en la tabla IV.

Tabla IV

Alturas promedio de personas jóvenes en Colombia

	Mujer		**Hombre**		**Promedios**
Medida (cm) / Edad (años)	***20 - 29***	***30 - 39***	***20 - 29***	***30 - 39***	
Altura	156.9	155.8	170.1	168.9	**162.9**
Altura ojos (de pie)	146.3	145.4	159.1	158.2	**152.3**
Altura acromial (de pie)	128.0	127.3	138.6	138.1	**133.0**

Medidas antropométricas promedio en personas jóvenes de Colombia. Source: Avila-Chaurand, Rosalio & Prado-León, Lilia & González-Muñoz, Elvia. (2007). Dimensiones antropométricas de la población latinoamericana : México, Cuba, Colombia, Chile / R. Avila Chaurand, L.R. Prado León, E.L. González Muñoz.

Por otra parte, basados en esta información, se diseñó la carcasa con unos ángulos que permitan su correcta visualización. (ver Fig. 58)

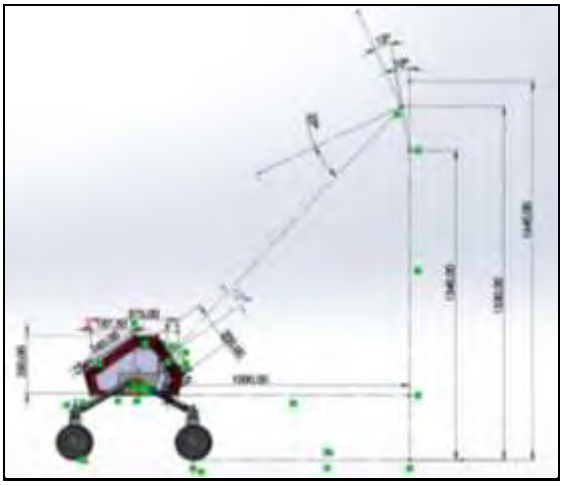


Fig. 58. CAD altura de visualización del robot.

Estructura física. La arquitectura física es muy importante porque permite definir los módulos del robot.

Primero, el robot percibe el entorno a través de una cámara y un sensor IMU. Además, la señal de la cámara y el IMU pasa al microprocesador que es el subsistema más importante en esta aplicación porque el robot necesita una buena capacidad de procesamiento. Finalmente, el movimiento lo dan los servomotores, estos motores solo necesitan un Arduino Mega 2560 conectado en forma serial. (ver Fig. 59)

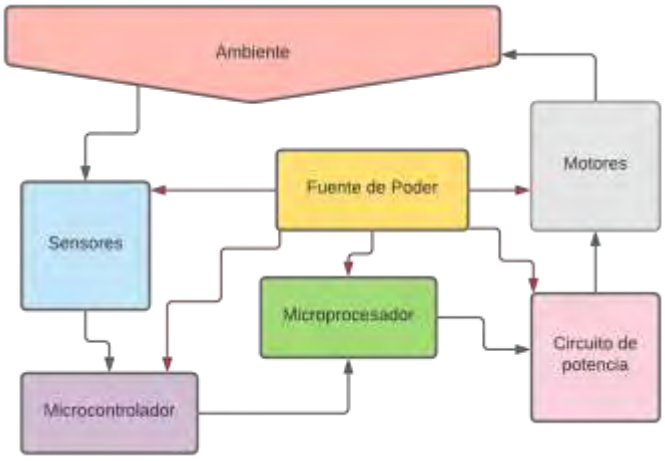


Fig. 59. Estructura física.

Estructura lógica. El robot interactúa con el entorno a través de sensores y actuadores. Primero, hay dos tipos de sensores que perciben los movimientos del robot para producir mediante un software el trazado de la ruta por donde ha transitado en un entorno de simulación. Además, cuenta con un tercer sensor que provee de imágenes una red neuronal que clasifica las imágenes según sus características y visualiza el resultado, todo este sistema es tele operado por el usuario. (ver Fig. 60)

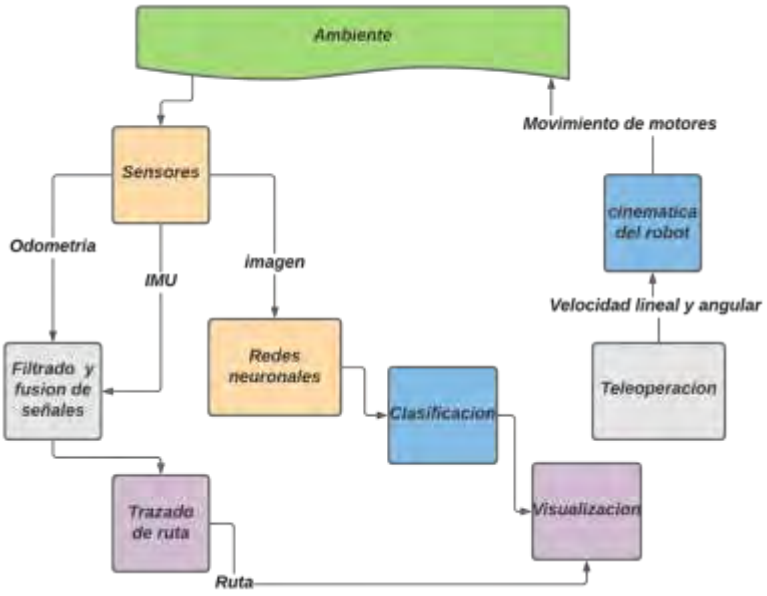


Fig. 60. Estructura lógica.

6.3 ETAPA 3: CONFIG.CION DEL ENTORNO DE SIMULACION

6.3.1 Actividad 5: ambiente de trabajo

Para la ejecución del proyecto en un entorno virtual, se empleó el simulador de ambientes virtual Gazebo y el visualizador de datos Rviz respectivamente.

6.3.1.1 Gazebo

Gazebo es un entorno capaz de simular uno o más robots en ambientes, tanto en exteriores como interiores de manera precisa y eficiente. Asimismo, permite ejecutar nodos de programación (ROS) para la generación de movimiento en articulaciones (joints)y diseñar robots. Este simulador cuenta con un motor robusto en cuanto a físicas y gráficos de alta calidad, lo cual permite el perfecto modelado de múltiples partes de distinto tamaño. Cabe resaltar, que Gazebo posee una gran ventaja, ya que es un software de licencia libre. Por esta y muchas otras razones, es el simulador de robots por excelencia en la actualidad y una herramienta muy importante en el mundo del desarrollo de robots. (ver logo en Fig. 61)

Fig. 61. Logo de gazebo.[39]

6.3.1.2 Rviz

Rviz es la herramienta de visualización 3D por excelencia en aplicaciones de robótica, ya que cuenta con la capacidad de mostrar modelos del robot, captura de información por sensores y reproducción datos. Este visualizador tiene la particularidad de integrarse perfectamente con entornos de simulación físicos (mundo real) y virtuales(Gazebo), que al implementarse con tópicos del sistema operativo robótico (ROS), se puede evidenciar toda la información contenida por estos entornos en RVIZ. (ver logo Fig. 62)

Fig. 62. Logo RVIZ [40]

6.3.2 Actividad 6: ruta a monitorear

El sótano dos de la Universidad Autónoma de Occidente fue el entorno de trabajo seleccionado a monitorear; a continuación se muestra un mapa del lugar. (ver Fig. 63)

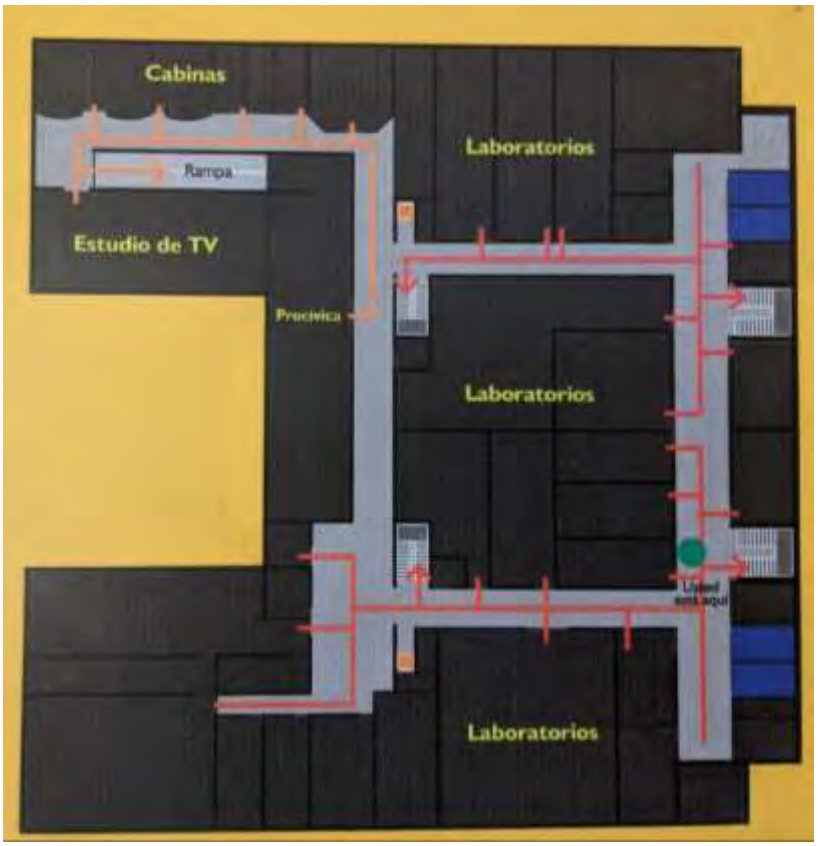


Fig. 63. Plano sótano dos Universidad Autónoma de Occidente.

Para la generación del entorno virtual en formato 3D (sótano dos), se utilizaron los planos suministrados por el arquitecto de planta de la Universidad Autónoma de Occidente. Estas representaciones graficas fueron esenciales para la creación del mapa 3D en la herramienta de modelado y simulación mecánica SolidWorks, el cual generó un paquete URDF que contenía el mapa 3D del sótano en código fuente. (ver Fig. 64)

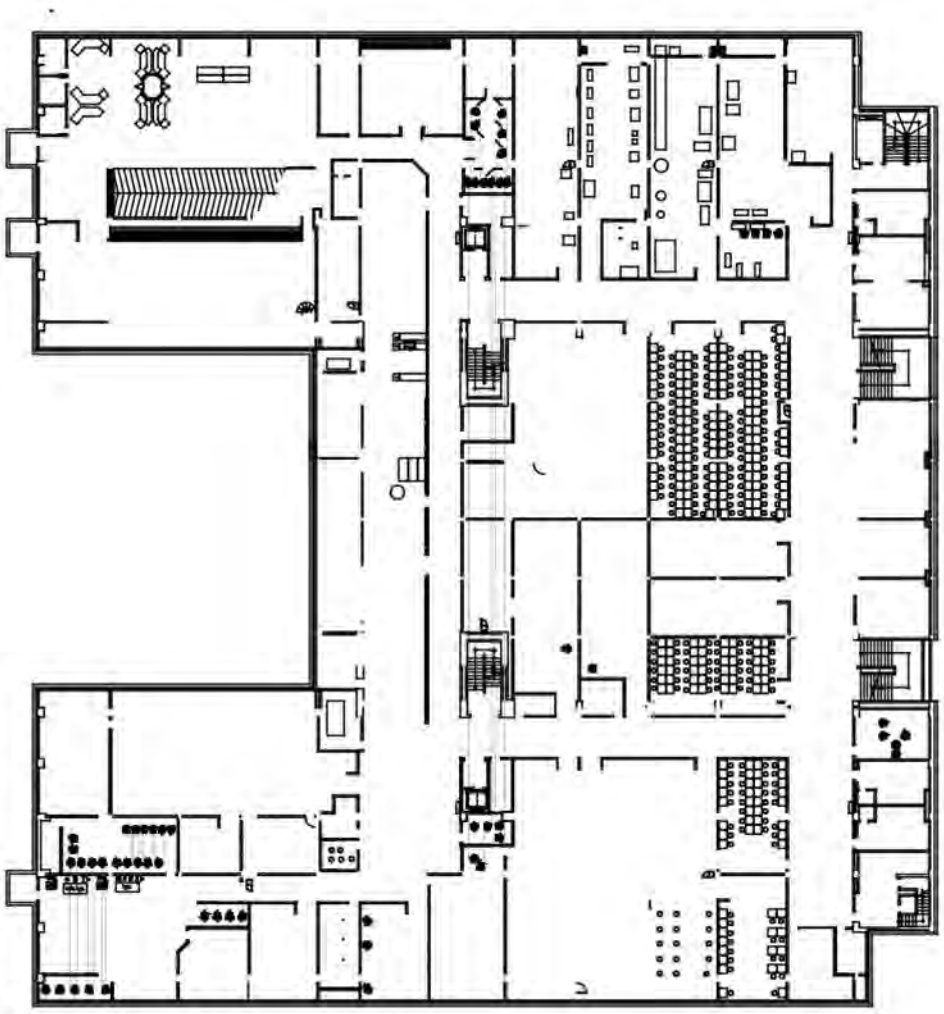


Fig. 64. Plano arquitectónico sótano dos Universidad Autónoma de Occidente

6.4 ETAPA 4: DESARROLLO DE LOS SISTEMAS DE LOCALIZACION

6.4.1 Actividad 7: sistema de localización métrica

Para el desarrollo del sistema de localización métrico, fue necesario emplear el simulador mecánico SolidWorks 3D para el diseño y creación del paquete URDF del modelo virtual del robot. Como ya se había mencionado, este simulador tiene la capacidad de generar dichos paquetes, los cuales contienen la estructura física y datos necesarios para el correcto funcionamiento en el mundo virtual Gazebo.

Después de validar el modelo virtual en Gazebo, se generó un archivo ejecutable denominado “fusión.launch”, el cual creó un mundo virtual vacío y la exposición del robot ante este mundo. De igual manera, este “script” ejecutó dos nodos del sistema operativo robótico (ROS), los cuales generaron la comunicación entre las articulaciones del robot virtual y los publicadores de estado. Por último, este archivo “.launch” lanzó el visualizador Rviz para comprobar los grados de libertad del modelo. Para mayor claridad del tema puede ver la tabla V.

Tabla V

Componentes de lanzamiento

Mundo Vacío Para Gazebo	Se inicializa un mundo vacío para ejecutar en el mundo virtual de gazebo
Modelo del Robot	Se almacena el modelo del robot en el parámetro denominado ”robot_description”
Joint state publisher y Robot State Publisher	Se ejecutan los nodos publicadores de estado
Interfaz	Se activa la opción de interfaz interactiva en Rviz para los joints del robot
Rviz	Se ejecuta el nodo de rviz para visualizar
Gazebo	Se ejecuta el nodo de gazebo con el parámetro de robot description, dando como resultado el spawn del robot en el mundo virtual.

Como se puede observar en la Fig. 65 la visualización en Rviz permitió comprobar el movimiento de cada articulación (joint) del robot, por medio de una interfaz denominada “gui”, la cual proporcionó valores interactivos para cada grado de libertad del modelo. Este sistema de “visualización-validación” permite evidenciar el funcionamiento de cada articulación del robot, por lo cual, si hay certeza de algún error en su estructura, este puede modificarse muy fácilmente en código fuente del paquete URDF.

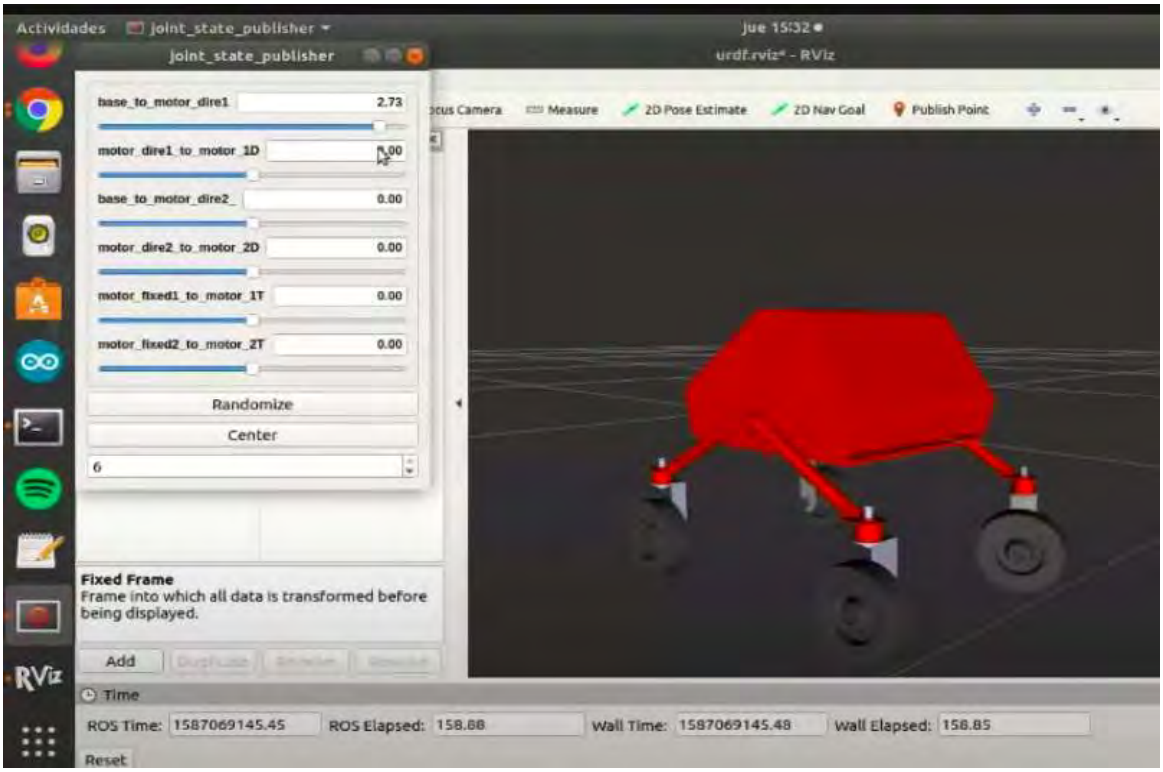


Fig. 65. Chimuelo en RVIZ (visualizador)

En la Fig. 66 se expone el modelo virtual dentro de un mundo vacío(Gazebo).

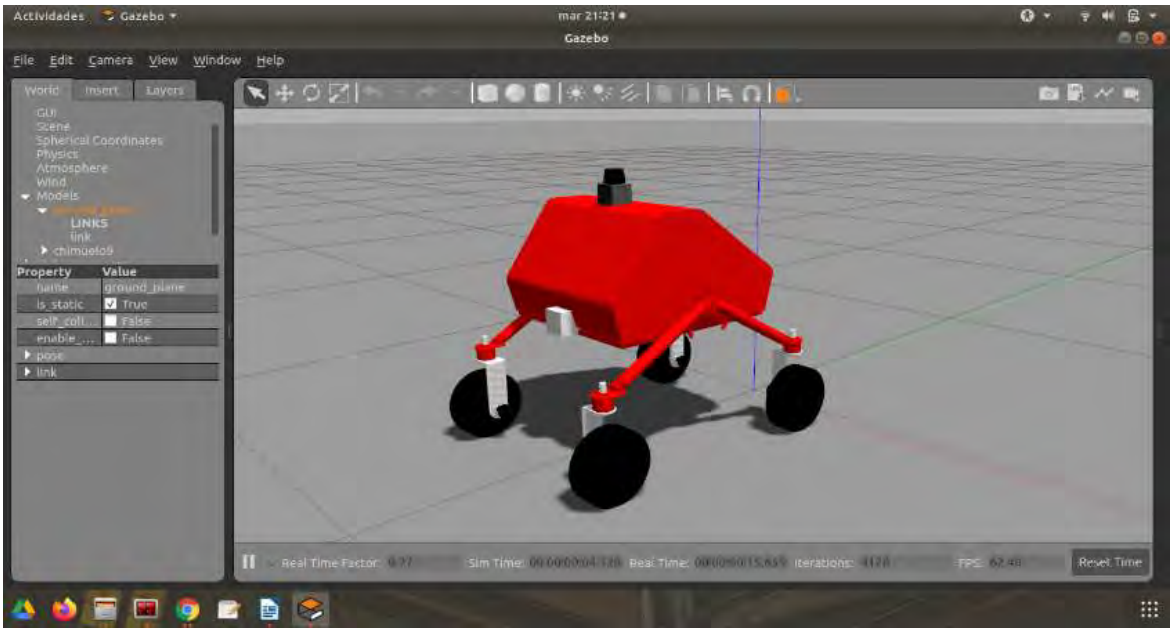


Fig. 66. Chimuelo en Gazebo (mundo virtual)

Posteriormente se transformó el modelo URDF a XACRO, esto con el objetivo de facilitar la escritura de código, considerando que gran parte de la conFig.ción de del modelo y parámetros de los enlaces (links) se repetía. Cabe resaltar que el formato XACRO tiene la particularidad, la cual es crear métodos o funciones que ejecutan instrucciones de programación cada vez que se invocan.

```
<robot  xmlns:xacro="http://www.ros.org/wiki/xacro"
        xmlns:sensor="http://playerstage.sourceforge.net/gazebo/xmlschema/#sensor"
        xmlns:controller="http://playerstage.sourceforge.net/gazebo/xmlschema/#controller"
        xmlns:interface="http://playerstage.sourceforge.net/gazebo/xmlschema/#interface"
        name="chimuelo9">
```

Fig. 67. Inicialización del modelo en .Xacro

Seguidamente se crearon dos tipos de funciones por medio de “xacro:macro”, las cuales establecieron transmisiones de dirección y velocidad de una forma rápida cada vez que se invocaban. Cabe señalar que estas funciones producen un puente de comunicación entre ROS y los valores de rotación de cada sistema de coordenadas (frame o joint) del robot.

Adicionalmente, los métodos de transmisión funcionaron de acuerdo al tipo de actuador (hardwareInterface) conFig.do, un claro ejemplo para el caso en detalle puede verse en las Fig.s 68 y 69, donde se emplearon dos tipos de actuadores, el VelocityJointInterface y el EffortJointInterface. El primer actuador permitió el

movimiento constante de la articulación, en tanto que el segundo permitió el control de posición de los marcos de referencia.

```
<!--////////////////////////// XACRO MACRO CON TRANSMISION DE VELOCIDAD ////////////////////////////// -->

    <xacro:macro name="transmission_block1" params="joint_name1">
        <transmission name="tran2">
            <type>transmission_interface/SimpleTransmission</type>
            <joint name="${joint_name1}">
                <hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface>
            </joint>
            <actuator name="foo_motor1">
                <hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface>
                <mechanicalReduction>1</mechanicalReduction>
            </actuator>
        </transmission>

    </xacro:macro>
```

Fig. 68. Función de transmission block(velocidad)

```
<!--////////////////////////// XACRO MACRO CON TRANSMISION DE POSICION ANGULAR(RADIANES) -->

    <xacro:macro name="transmission_block" params="joint_name">
        <transmission name="tran1">
            <type>transmission_interface/SimpleTransmission</type>
            <joint name="${joint_name}">
                <hardwareInterface>EffortJointInterface</hardwareInterface>
            </joint>
            <actuator name="foo_motor">
                <mechanicalReduction>1</mechanicalReduction>
                <hardwareInterface>EffortJointInterface</hardwareInterface>
            </actuator>
        </transmission>

    </xacro:macro>
```

Fig. 69. Función de transmission block(Posición)

Para mayor claridad se presentan las dos(2) funciones de transmisiones empleadas y sus respectivos actuadores (véase la tabla VI).

Tabla VI

Funciones de transmisión

Nombre de la función	Funcion	Tipo de Actuador
Xacro:transmission_block	Funcion de transmision para dirección	<hardwareInterface>**EffortJointInterface**</hardwareInterface>
Xacro:transmission_block1	Funcion de transmision para velocidad	<hardwareInterface>**hardware_Interface/VelocityJointInterface**</hardwareInterface>

En la tabla VI, se observa el funcionamiento de las transmisiones implementadas en el código principal del robot, las cuales se emplearon para enviar información desde un nodo ROS (generador de valores) hacia las articulaciones virtuales del robot.

Posteriormente se invocó el llamado de varias funciones, pasando como parámetro el nombre del “joint” o articulación de movimiento.

```
<xacro:transmission_block joint_name="base_to_motor_dire1"/>
<xacro:transmission_block1 joint_name1="motor_dire1_to_motor_1D"/>
<xacro:transmission_block joint_name="base_to_motor_dire2_"/>
<xacro:transmission_block1 joint_name1="motor_dire2_to_motor_2D"/>
<xacro:transmission_block1 joint_name1="motor_fixed1_to_motor_1T"/>
<xacro:transmission_block1 joint_name1="motor_fixed2_to_motor_2T"/>
```

Fig. 70. Archivo .xacro

Adicionalmente, se creó un archivo denominado ”joint_name.yaml, el cual se caracterizó por contener la conFig.ción PID de los controladores para cada articulación (joint) de velocidad y dirección. Estas conFig.ciones fueron de gran relevancia para limitar el torque de los actuadores contenidos en las transmisiones.

Fig. 71. Archivo joint_name.yaml(Parámetros de todos los controladores)

Tele operación: para el desarrollo de la tele operación del robot, se empleó un nodo del sistema operativo robótico (ROS), el cual incrementó o decrementó la velocidad lineal y angular por medio de la activación de teclas (keyboard). Este nodo utilizó varias librerías (véase la tabla VII) necesarias para el envío de información hacia nodos del sistema de localización métrico, tales como la cinemática del robot virtual y real.

Tabla VII

Librerías de Tele operación

Librerías	Función
import rospy	Compilar instrucciones de ROS en python.
from geometry_msgs.msg import Twist	Expresar velocidades en partes divididas
import sys, select, os	Acceder al sistema operativo, intérpretes y selección de datos.

En la tabla VII se evidencia las librerías de ROS utilizadas para el desarrollo de la tele operación.

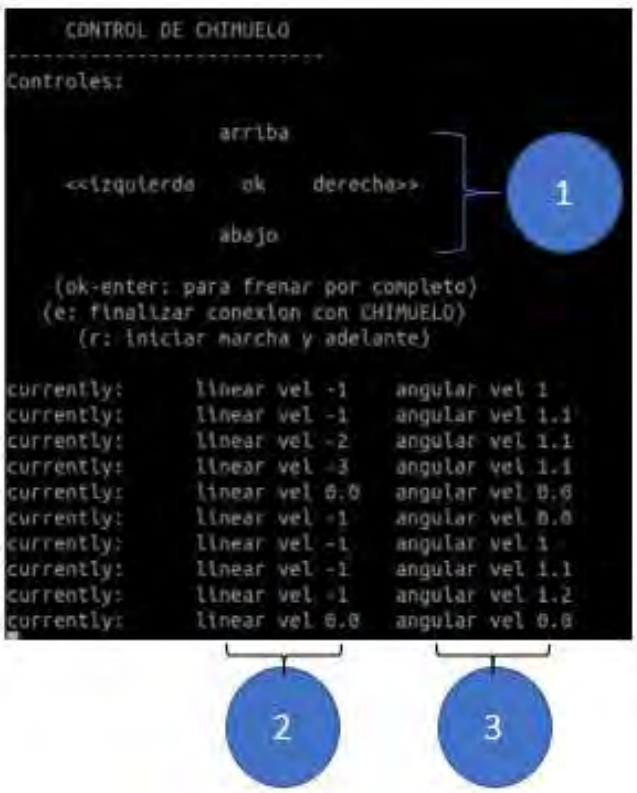


Fig. 72. Teleoperación en funcionamiento

Como se puede observar en la imagen 72 existen tres conjuntos de información principales en la terminal de control. El primero muestra la información relacionada a los controles de mando, el segundo expone el historial de la referencia llamada velocidad lineal y el tercero corresponde al historial de referencia denominada velocidad angular.

Para generar un movimiento adecuado al robot chimuelo, fue necesario comprender cómo su parte mecánica afectaba al desplazamiento del mismo, de modo que fue imprescindible realizar un análisis cinemático. Como ya se había mencionado, el robot móvil (chimuelo) hace parte del tipo Ackermann, por lo cual, necesita de dos ángulos de dirección en sus ruedas delanteras para efectuar un giro como se puede ver en la Fig. 73. Además, estos valores de rotación son necesarios para encontrar restricciones de rodamientos y desplazamiento del robot.

Adicionalmente, la conFig.ción tipo Ackermann permite en su análisis cinemático simplificar su observación en los ángulos de dirección, reduciendo los dos ángulos principales por uno solo como se observa en la Fig. 73. Este ángulo central permite reducir la complejidad del análisis cinemático, dando como resultado el estudio del movimiento de un triciclo.

Además, para que el robot pueda dar un rotación, este debe encontrar un radio de giro denominado ICR, el cual se halla por medio de las velocidades angulares y lineales estipuladas por el usuario al momento de usar la tele operación.

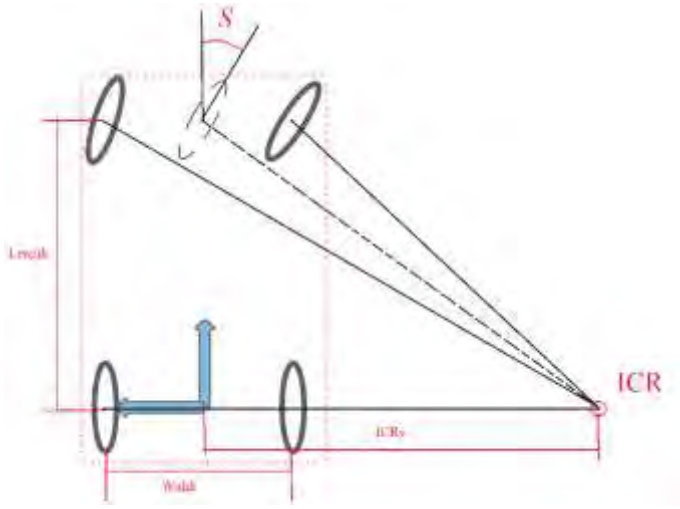


Fig. 73. ICR de robot tipo Ackermann.

$$ICRy = \frac{Velocidad\ lineal}{Velocidad\ angular} \quad \textbf{(14)}$$

Fig. 74. ICR del robot.

Luego de contar con el ICR (Fig. 74) fue necesario calcular el Angulo S (ver ecuación 15 o Fig. 75), el cual pertenece a la rueda virtual que se planteó, y a partir de allí determinar el ángulo que le corresponde a cada rueda delantera del robot (ver ecuaciones 16 y 17 o Fig.s 76,77). Además, en la obtención de los ángulos Sa y Sb se empleó trigonometría simple.

$$S = a\tan\left(\frac{length}{ICRy}\right) \quad \textbf{(15)}$$

Fig. 75. Angulo de la rueda virtual.

$$S\alpha = a\tan 2\left(\frac{Length}{ICRy - \frac{Width}{2}}\right) \quad (16)$$

Fig. 76. Angulo rueda a.

$$S\beta = a\tan 2\left(\frac{Length}{ICRy + \frac{Width}{2}}\right) \quad (17)$$

Fig. 77. Angulo rueda B.

6.4.1.1 Odometría virtual.

Para hallar la odometría virtual, se empleó un nodo del sistema operativo robótico (ROS), el cual permitió la obtención de datos del simulador virtual Gazebo, tales como coordenadas, orientación y aceleración del robot. Este sistema operativo robótico manipuló los datos para su posterior envío o muestra en terminales. Para mayor claridad sobre el contenido del nodo véase la tabla VIII.

Tabla VIII

Librerías para generar odometría

Librería	Función
import rospy	Compilar instrucciones de ROS
from std_msgs.msg import Header	librería de ROS para construir cabeceros
from nav_msgs.msg import Odometry	Librería de ROS para construir y compilar odometría.
from gazebo_msgs.srv import GetModelState, GetModelStateRequest	Librería de ROS para acceder a modelos activos dentro del simulador.

Este nodo de ROS creó dos cabeceros de identidad, los cuales almacenaban valores de la odometría y el modelo del robot que contenía el simulador gazebo. Asimismo, este script obtenía todos los modelos activos que estaban en el simulador virtual, con la finalidad de identificar y rechazar información ajena al robot chimuelo, como lo es el mapa 3D del sótano dos de la Universidad Autónoma de Occidente (información ajena al robot). Al momento de identificar el modelo del robot, este se almacenó en una variable denominada “result”, la cual se empleó para extraer las poses (poses) y los giros (Twist) del robot. Por último, el nodo contuvo toda la información dentro del cabecero denominado “odom”, el cual fue publicado por un tópico llamado “/my_odom”. Para mayor claridad véase la Fig. 78.

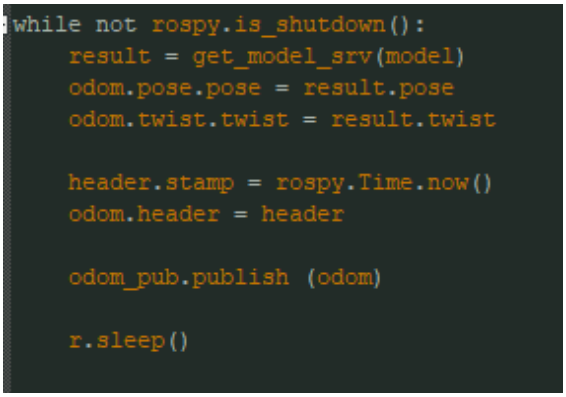


Fig. 78. Generación de odometría .

Para determinar la orientación y aceleración del modelo virtual, se generó un archivo de conFig.ción llamado “imu.yaml”, el cual tuvo como parámetro de entrada la odometría virtual publicada con el topico “/my_odom”, el cual contenía los valores necesarios para la estimación del IMU. De igual forma, este archivo inicializó la extensa conFig.ción de parámetros para el correcto funcionamiento del sensor virtual, como tiempo de espera del sensor, frecuencia de muestreo, tamaño de cola, límites de rechazos, aceleración gravitacional y covarianza de ruido. Por ultimo esta información se integró a la odometria mediante un tópico de ROS denominado “/imu”.

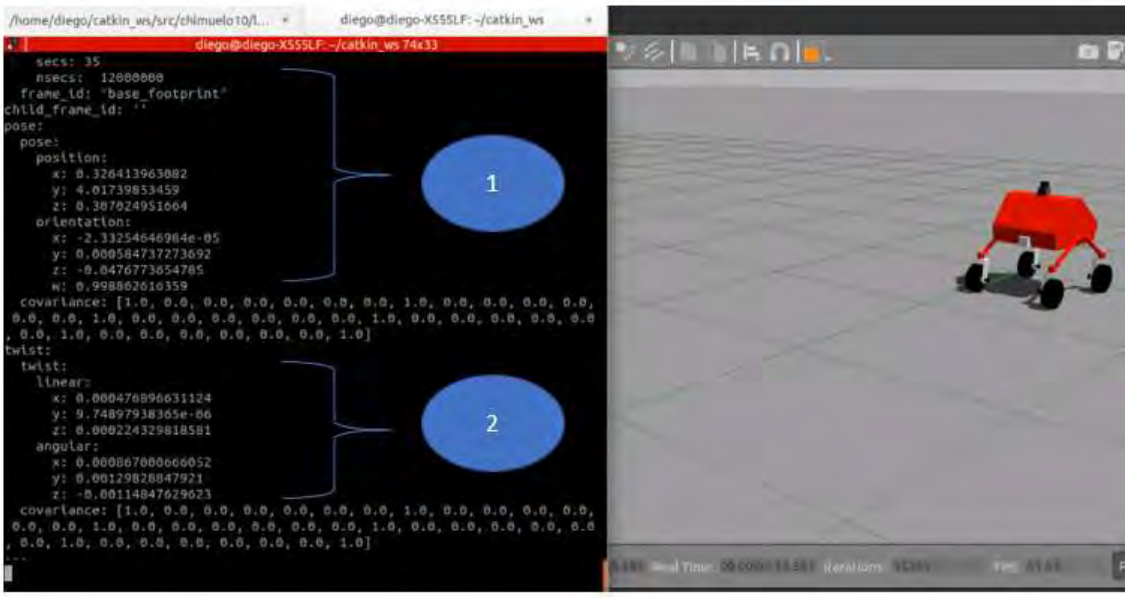


Fig. 79. Visualización de odometría(posición y orientación) en terminal.

Como se puede observar en la imagen 79 se encuentra dividida en dos conjuntos de datos. El primero corresponde a la localización métrica(posición y orientación) del robot virtual, mientras que el segundo expone la información respectiva a la aceleración lineal y angular.

6.4.1.2 Odometría real.

Para determinar la localización métrica real del robot chimuelo, se empleó un nodo de ROS en el IDE de Arduino, con el fin de encontrar y enviar las coordenadas 2D del modelo físico hacia el sistema operativo robótico. Para esto, se necesitó la velocidad lineal dada por el encoder de los actuadores, el ángulo de giro, hallado por sensores, y la velocidad angular, determinada por trigonometría básica. Para mayor claridad véase la Fig. 80 donde se aprecia la asignación de valores a las variables de interés.

```
// OBTENCION DE LA VELOCIDAD(SPEED)
velocidad_lineal = EncoderMotores();
// OBTENCION DEL ANGULO DE GIRO
angulo_de_giro = steering;
// OBTENCION DE LA VELOCIDAD ANGULAR
velocidad_angular = velocidad_lineal * tan(angulo_de_giro)/(distancia_entre_llantas);
```

Fig. 80. Adquisición de datos.

Después de encontrar y almacenar los valores de cada velocidad y ángulo, se emplearon las ecuaciones de actualización para las variables “X,Y” y ángulo de giro, como se puede observar en la Fig. 81. Estas ecuaciones estaban contenidas por el bucle “void loop” de Arduino, por lo cual, no fue necesario implementar una instrucción de repetición adicional para su actualización de datos.

```
x_actualizacion += cos(angulo_de_actualizacion)*velocidad_lineal;
y_actualizacion += sin(angulo_de_actualizacion)*velocidad_lineal;
angulo_de_actualizacion += angulo_de_giro*tiempo.toSec();
```

Fig. 81. Actualización de coordenadas.

Por último, este conjunto de datos fueron empaquetados(véase la Fig. 82) por la librería de ROS llamada “tf/transform_broadcaster”, la cual creó los datos esenciales de ROS tales como el tópico de publicación “/odom” y el marco de identidad “base_link”. Así mismo, esta librería realizó el envío de datos hacia el sistema operativo robótico (ROS) para ser manipulados por otro nodo que requirió la información y su posterior visualización de los datos en RVIZ(véase la Fig. 82).

```
t.header.frame_id = odom;
t.child_frame_id = base_link;

t.transform.translation.x = x_actualizacion;
t.transform.translation.y = y_actualizacion;

t.transform.rotation = tf::createQuaternionFromYaw(angulo_de_actualizacion);
t.header.stamp = nh.now();

broadcaster.sendTransform(t);
nh.spinOnce();
```

Fig. 82. Envío datos a ROS.

6.4.1.3 Imu real.

Para realizar la lectura de los valores de aceleración y orientación del IMU real, se necesitó de un nodo de ROS capaz de realizar dicha función. Este nodo fue implementado en el IDE de Arduino, con el objetivo de extraer información desde la unidad de medición inercial, empaquetar su información en una cadena de texto y enviarla en un tópico de publicación ROS, como se puede ver en la Fig. 83.

Además, este script de Arduino funcionó gracias a la implementación de la librería “Wire.h” ,la cual registró todos los caracteres que generaba el IMU físico de acuerdo a las direcciones de memoria de la tarjeta Arduino MEGA 2560 y los almacenó en variables de interés, como se observa en la Fig. 83.

```
void loop()
{
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr,14,true);

  AcX=Wire.read()<<8|Wire.read();
  AcY=Wire.read()<<8|Wire.read();
  AcZ=Wire.read()<<8|Wire.read();
  Tmp=Wire.read()<<8|Wire.read();
  GyX=Wire.read()<<8|Wire.read();
  GyY=Wire.read()<<8|Wire.read();
  GyZ=Wire.read()<<8|Wire.read();
  String AX = String(AcX);
  String AY = String(AcY);
  String AZ = String(AcZ);
  String GX = String(GyX);
  String GY = String(GyY);
  String GZ = String(GyZ);
  String tmp = String(Tmp);
```

Fig. 83. Lectura de datos IMU real.

Finalmente, estos datos fueron enviados por el tópico de publicación “imu117”, el cual fue suscrito por un nodo lector de datos que tuvo como objetivo la descompresión del conjunto de información y la envió a un tópico adicional para su posterior fusión (filtro de Kalman) y la visualización en RVIZ.

6.4.1.4 Fusión de la odometría real y virtual para la estimación de “pose” en 3D.

Para generar la localización métrica del robot por estimación, se necesitó fusionar la odometría y orientación del modelo tanto virtual como físico. Esto con el objetivo de poder crear una pose estimada 3D que tuviera un error pequeño respecto a las medidas reales. Por lo cual, se empleó un filtro de Kalman extendido, contenido en el paquete “robot_pose_ekf”. Este filtro toma como entrada dos conjuntos de información, como lo son odometría virtual (odometría con orientación

virtual) y odometría real (odometría con orientación real), el cual estimó nueva información en base a los datos tomados.

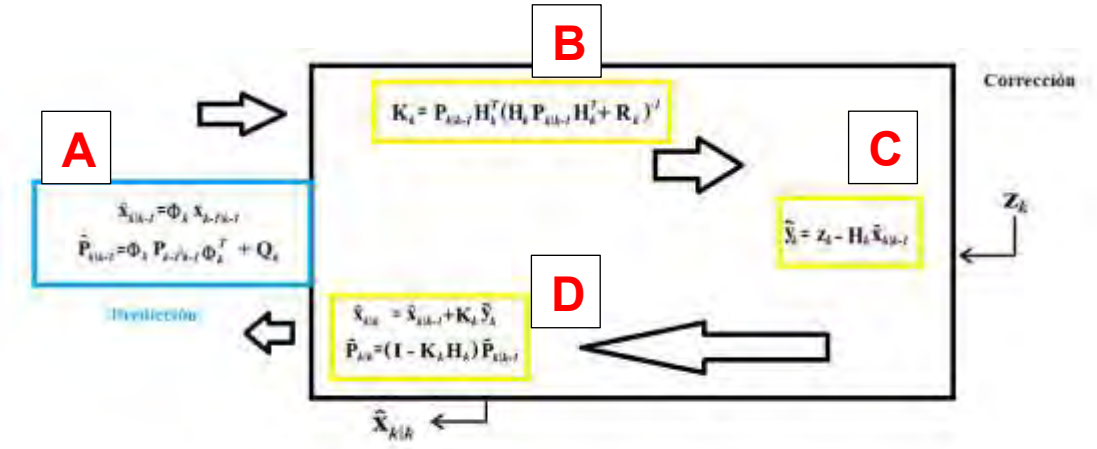


Fig. 84. Ecuaciones filtro de Kalman.

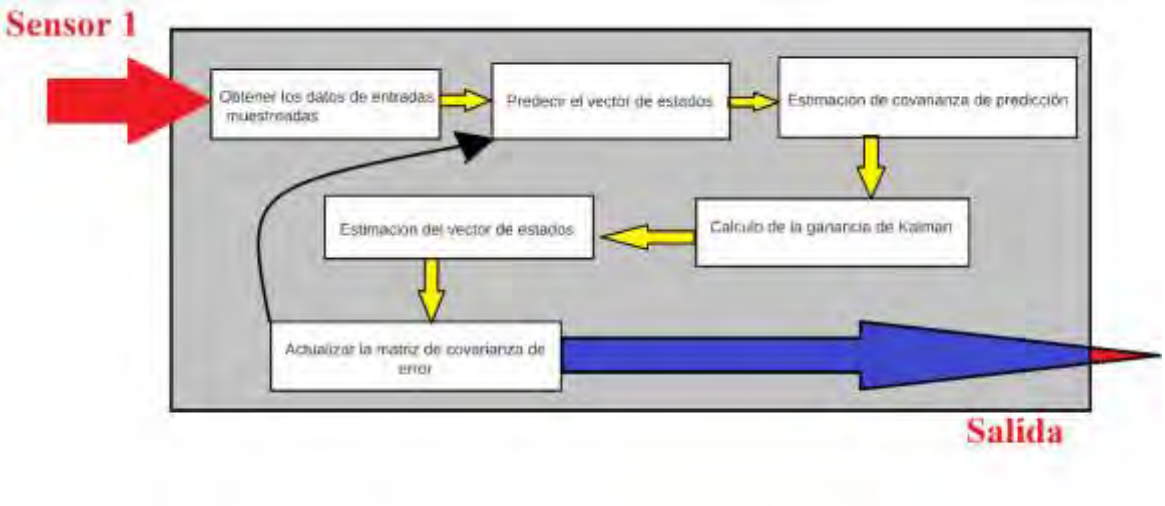


Fig. 85. Diagrama de flujo filtro de Kalman

El sistema con el que se está trabajando se denomina estocástico, ya que es posible conocer sus estados posteriores tanto por las acciones propias del sistema como por elementos aleatorios.

Una característica de este tipo de sistemas (estocásticos), es que su variable de estado contiene insuficiente información para predecir con exactitud los estados futuros.

Dado que el filtro de Kalman es un algoritmo recursivo, tiene la capacidad de correr en tiempo real usando únicamente las mediciones de entrada actuales, el estado calculado previamente y su matriz de incertidumbre. Esto se convierte en un gran beneficio, ya que no requiere más información para su funcionamiento.

Según el algoritmo del filtro discreto de Kalman (ver Fig. 84), es posible describir el proceso en dos pasos, el primero es la predicción y el segundo es la corrección. Sin embargo, se cuenta con una particularidad para la aplicación en la que se está implementando, ya que al ser no lineal, es necesario realizar un proceso de linealización del sistema en torno a $\hat{x}(t)$ mediante Taylor en la etapa de predicción; este paso adicional es lo que se denomina filtro extendido de Kalman.

En la Fig. 84 es posible observar cuatro cuadros denominados con las letras A,B,C y D, los cuales corresponden a:

Cuadro A. estimación a priori y la covarianza del error asociada a la estimación a priori.

Cuadro B. Ganancia de Kalman.

Cuadro C. Actualización del residuo de la medición.

Cuadro D. Estimación a posteriori y a la covarianza del error asociada a la estimación a posteriori.

En la Fig. 85, se observa las entradas y salidas del filtro de Kalman y su funcionamiento interior.

En la Fig. 79 es posible observar el formato como es suministrados los datos al filtro de Kalman, el cual se compone de pose (posición, orientación y su respectiva covarianza) y twist (lineal, angular y su respectiva covarianza). Estos datos primeramente son filtrados de manera independiente y posteriormente son fusionados.

Por otro lado, en la conFig.ción del paquete “Robot_Pose_Ekf” fue necesario la conFig.ción de una frecuencia de actualización y publicación del filtro, ya que de inicializarse con un valor considerablemente grande, este habría generado una sobrecarga del sistema y no funcionaría correctamente (una alta frecuencia no significa que puede obtener más información sobre la postura del robot en un tiempo determinado). El segundo parámetro conFig.do fue el tiempo de espera o actualización, ya que la información de los sensores es tomada en este rango de tiempo y luego es desestimada. Además, una de las ventajas que trae el paquete “Robot_Pose_Ekf” es el seguir operando si algún conjunto de información pierde conexión. Por último, se conFig. los datos que van a ingresar, en este caso aparece “odom_used” , "imu_used" y "vo_used", los cuales corresponden a la odometría, imu y odometría visual. En caso de este proyecto se deshabilita la odometría visual asignándole value="false”.

```
<launch>

  <node pkg="robot_pose_ekf" type="robot_pose_ekf" name="robot_pose_ekf">

    <param name="output_frame" value="odom"/>

    <param name="freq" value="30.0"/>

    <param name="sensor_timeout" value="1.0"/>

    <param name="odom_used" value="true"/>

    <param name="imu_used" value="true"/>

    <param name="vo_used" value="false"/>

    <param name="debug" value="false"/>

    <param name="self_diagnose" value="false"/>

  </node>
```

</launch>

Este paquete (robot_pose_ekf) suministra los siguientes datos:

Odometry message coniene. x, y and yaw angle

IMU message. contiene roll, pitch and yaw angles.

VO message contiene. x,y,z and roll, pitch, yaw angles.

Posteriormente, estos datos se publican en un nodo de ROS con el objetivo de ser enviados a otro paquete llamado “odom_to_trayectory”, el cual tiene la función de almacenar cada posición resultante del filtro de Kalman y dar la posibilidad de ser representada gráficamente en el visualizador rviz.

En el paquete “odom_to_trayectory” únicamente es necesario conFig.r el nodo al cual se desea subscribir, ya que cuenta con la posibilidad de ejecutar dos scrips independientes. El primero “path_odom_plotter”, el cual se suscribe a la odometría 2D sin filtrar, agrega las poses del robot y publica la trayectoria del robot sin filtrar. El segundo es “path_ekf_plotter”, se suscribe a la odometría filtrada en 3D, agrega las poses del robot y publica la trayectoria filtrada del robot. Finalmente, todos estos datos se visualizan en RVIZ (ver Fig.s 100, 105, 106, 107 y 108).

6.4.2 Actividad 8: sistema de localización topológica

Un Sistema de Localización Topológica o STL tiene como objetivo identificar lugares característicos como laboratorios o pasillos basados en la apariencia captada por la cámara en el sótano dos de la Universidad Autónoma de Occidente. Para el desarrollo de este sistema se necesitó un método de clasificación confiable que permitiera la identificación de lugares en forma rápida y precisa. Para ello se utilizaron las redes neuronales convolucionales o CNN, con las cuales se realizó una clasificación de diferentes imágenes tomadas por la cámara web con el fin de determinar la ubicación espacial del robot.

A) B) C)

Fig. 86. Puntos de referencia en sótano dos.

En la Fig. A se puede observar el logo del noticiero 90 min cercano a su entrada y al laboratorio de de producción audiovisual, en la Fig. B se observa el servidor cercano a el laboratorio de industrial y simulación mecánica. Finalmente se observa en la Fig. C las mesas ubicadas al interior del laboratorio de automática y cerca del laboratorio de multimedia.

Como se puede observar en las Fig. 86 los objetos y lugares que fueron seleccionados están cerca a laboratorios y pasillos frecuentemente utilizados por los estudiantes de la universidad, actuando como nodos digitales para localizar zonas de gran interés.

Para la creación del dataset se seleccionaron objetos y lugares representativos de las zonas de interés. En total se definieron 13 puntos claves para localizar laboratorios, pasillos y zonas de evacuación. Estos puntos actuaron como clases, las cuales almacenaron el nombre del objeto o lugar a reconocer. Cabe resaltar que se tomaron 50 imágenes por cada punto clave o clases, las cuales se redimensionaron a un tamaño de “512x512x3” px con el fin de cumplir el tamaño de entrada de la CNN, sin embargo, este número de imágenes no fue suficiente para distribuir entre datos de entrenamiento y de validación. En consecuencia se necesitó una técnica que incrementara el dataset; una de estas técnicas fue el “data augmentation”, la cual permitió el incremento de todo el dataset de entrenamiento y de validación por medio del preprocesamiento de imágenes, generando así, múltiples representaciones gráficas con transformaciones matriciales (rotación de varios ángulos, ruido y zoom).

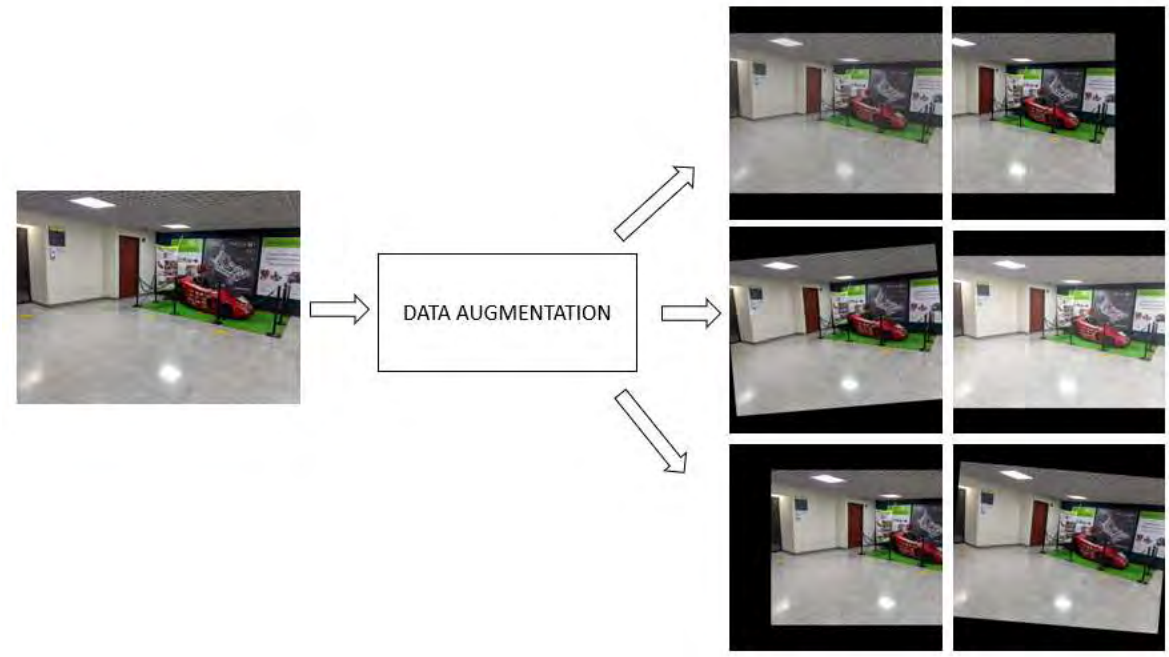


Fig. 87. Esquema de preprocesamiento y data augmentation

En la imagen 87, se muestra el funcionamiento del “Data Augmentation”, el cual consiste en el preprocesamiento de una imagen(Fig. 87-a) que es expuesta al proceso central, y posteriormente genera múltiples imágenes del tratamiento aplicado(Fig. 87-b, Fig. 87-c, Fig. 87-d) con rotación de varios ángulos y aplicación ruido.

Para el desarrollo del “Data Augmentation” se empleó un bucle “for”, el cual permitió leer cada imagen que contenía la lista file_list(lista contenedora de imágenes) y generar el procesamiento de imagen con varias conFig.ciones. Después de finalizar dicho proceso se almacenaron todos los resultados en la nube “drive”, para mayor claridad véase la Fig. 88.

```
for image in file_list:
    img = cv2.imread(image)

    #rotacion 90 grados
    rows,cols,depth = img.shape
    M = cv2.getRotationMatrix2D((cols/2,rows/2),90,1)
    dst = cv2.warpAffine(img,M,(cols,rows))
    cv2.imwrite('images_aug\image_' + str(c) + '.jpg',dst)
    c = c+1
    #rotacion 45 grados
    N = cv2.getRotationMatrix2D((cols/2,rows/2),45,1)
    dstN = cv2.warpAffine(img,N,(cols,rows))
    cv2.imwrite('images_aug\image_' + str(c) + '.jpg',dstN)
    c = c+1
    #rotacion a 180 grados
    P = cv2.getRotationMatrix2D((cols/2,rows/2),180,1)
    dstP = cv2.warpAffine(img,P,(cols,rows))
    cv2.imwrite('images_aug\image_' + str(c) + '.jpg',dstP)
    c = c+1
    #rotacion a 25 grados
    Q = cv2.getRotationMatrix2D((cols/2,rows/2),25,1)
    dstQ= cv2.warpAffine(img,Q,(cols,rows))
    cv2.imwrite('images_aug\image_' + str(c) + '.jpg',dstQ)
    c = c+1
```

Fig. 88. Código de data augmentation.

Fig.. 89. Resultado del data aumentation.

Después de generar el dataset, se llevó a cabo el proceso de entrenamiento de la red neuronal convolucional, para esto se utilizó un método llamado transfer learning, el cual consistió en la utilización de un modelo pre entrenado y el entrenamiento de la capa final. Este método se empleó con el fin de reducir la carga computacional y el tiempo de procesamiento del ordenador. Asimismo se utilizó Google colab para

realizar el entrenamiento de la red neuronal, con el objetivo de aprovechar los recursos gratuitos GPU de la nube.

Los datos generados por el proceso “Data Augmentation” que fueron almacenados en Drive se utilizaron con el fin de alimentar la red neuronal contenida en un script de Google Colab , por lo cual se estableció una comunicación entre la nube y este. Para ello se utilizó el comando “ ! cp -r ”, el cual permitió clonar archivos de la nube hacia el código de la red. Esto se puede observar en la Fig. 90, donde se muestra el copiado de múltiples archivos hacia la carpeta contenedora de Google Colab.

El dataset previamente clonado, estuvo conformado por cuatro archivos: imágenes de entrenamiento, validación, regiones de interés y las clases a identificar. Para mayor claridad, véase la tabla IX.

Tabla IX

Dataset

Carpeta llamada “**FUSIÓN**”	Contiene todas las imágenes y regiones de interés.
Archivo **annotations.csv**	Contiene una lista de direcciones de todas las imágenes de entrenamiento.
Archivo **annotations_test.cvs**	Contiene una lista de direcciones de todas las imágenes de validación.
Archivo **classes.csv**	Contiene los nombres de las clases.

```
# COPIADO DE ARCHIVOS DESDE EL DRIVE A LA CARPETA CONTENEDORA DE GOOGLE COLAB.

!cp -r "/content/drive/My Drive/CHIMUELO 2020/DATOS/FUSION" "/content/keras-retinanet"
!cp -r "/content/drive/My Drive/CHIMUELO 2020/DATOS/annotations.csv" "/content/keras-retinanet"
!cp -r "/content/drive/My Drive/CHIMUELO 2020/DATOS/annotations_test.csv" "/content/keras-retinanet"
!cp -r "/content/drive/My Drive/CHIMUELO 2020/DATOS/classes.csv" "/content/keras-retinanet"
```

Fig. 90. Copiado de archivos

La red neuronal pre entrenada denominada “coco”, cuenta con una arquitectura Resnet 50, la cual posee como característica principal las conexiones residuales.

Estas “saltan” o ignoran dos capas convolucionales(véase Fig. 91) y se integran a la ponderación del proceso.

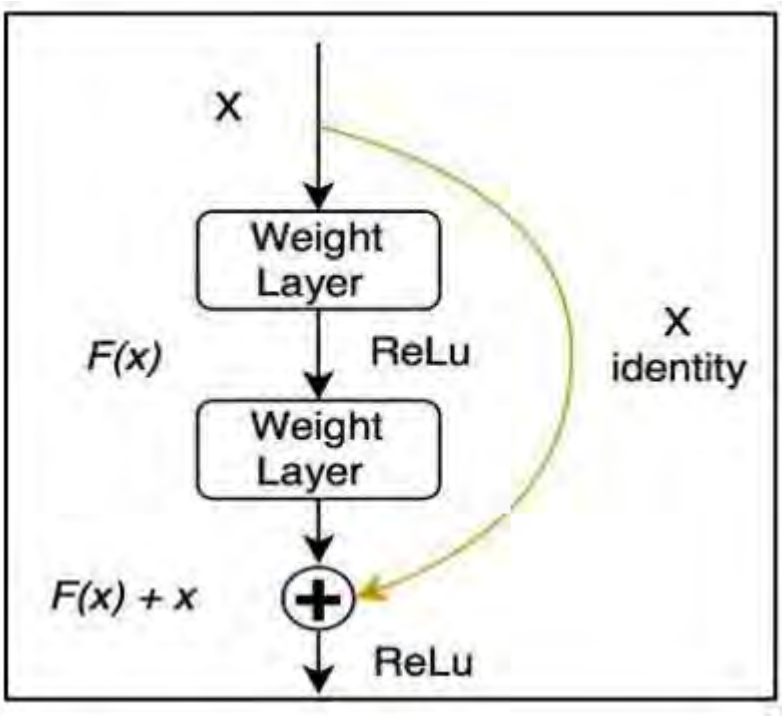


Fig. 91. Conexión residual. [41]

Las redes neuronales convolucionales residuales, representan una mayor precisión respecto a otras arquitecturas, puesto que permiten incrementar las capas convolucionales sin que el error aumente(sobreajuste), para dar más claridad sobre esto, véase la Fig. 92, donde se compara la Resnet de 18 y 34 capas vs plain de 18 y 34 capas. Estas gráficas evidencian como las conexiones residuales de la resnet permite aumentar las capas convolucionales para disminuir el error, suceso que no ocurre con la plain. En la arquitectura plain no permite disminuir el error con la adición de más capas convolucionales, si no que aumente el mismo.

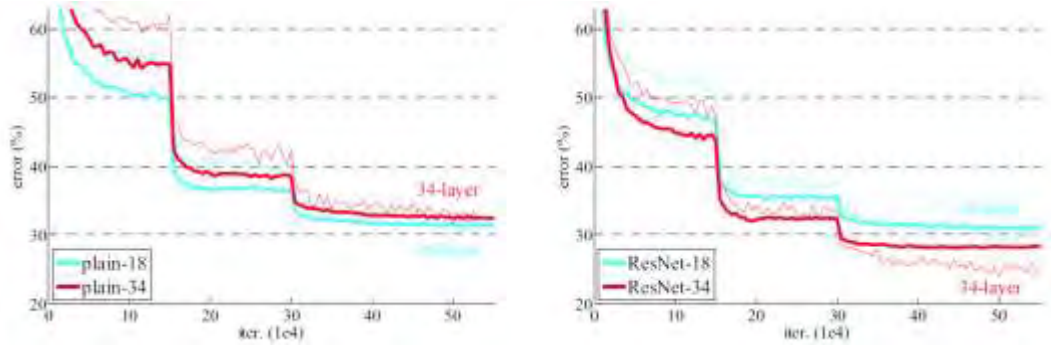


Fig. 92. resnet50 vs plain18 [42]

La Resnet50 se caracteriza por tener varios bloques de convolución repartidos por toda su arquitectura, a continuación se expone su organización en forma de bloques convolucionales, véase la Fig. 93.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fig. 93. Arquitecturas RESNET [43]

Como se puede observar, la Resnet50 está constituida por 50 capas, las cuales se pueden detallar en la tabla X.

Tabla X

Arquitectura resnet50

Convolución 1	7x7, 64 con stride 2
Max Pool	3x3, con stride 2
Bloque Convolucional 1	([1x1,64],[3x3,64],[1x1,256])x3 , se repite tres(3) veces, por lo tanto, da como resultado nueve(9) capas
Bloque Convolucional 2	([1x1,128],[3x3,128],[1x1,512])x4 , se repite cuatro veces, por lo tanto, da como resultado doce(12) capas
Bloque Convolucional 3	([1x1,256],[3x3,256],[1x1,1024])x6 , se repite seis(6) veces, por lo tanto, da como resultado dieciocho(18) capas
Bloque Convolucional 4	([1x1,512],[3x3,512],[1x1,2048])x3 , se repite tres(3) veces, por lo tanto, da como resultado nueve(9) capas
Bloque Full Conectado	capa full conectada con mil(1000) nodos

La suma de todas las capas se puede observar en la tabla XI

Tabla XI

Capas por bloque

Convolución 1	1 Capa
Bloque Convolucional 1	9 Capas
Bloque Convolucional 2	12 Capas
Bloque Convolucional 3	18 Capas
Bloque Convolucional 4	9 Capas
Bloque Full conectado	1 Capa

Luego, se desarrolló el entrenamiento de la red neuronal con un “batch size” de 8, “steps” de 200 y el número de épocas igual a 100, asimismo se pasó como parámetro el nombre de las clases con las imágenes de entrenamiento utilizando la arquitectura Resnet.

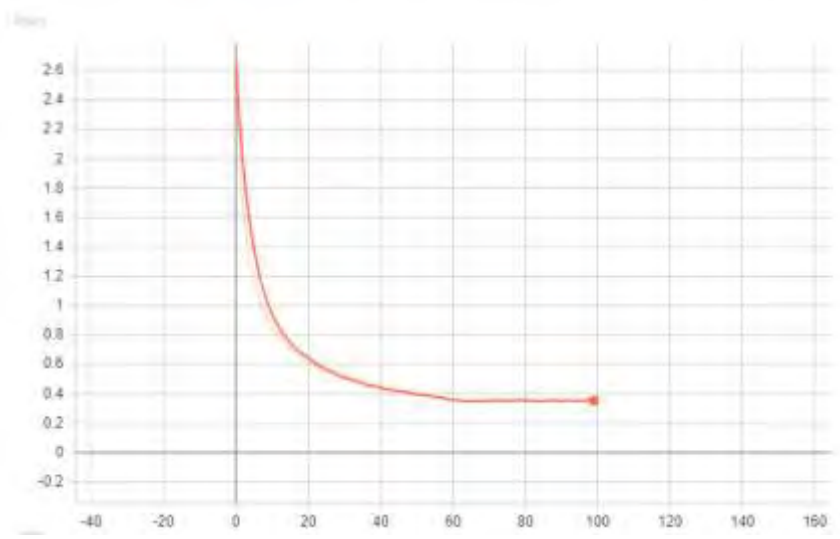


Fig. 94. Perdida de la Red Neuronal.

En la Fig. 94, se observa el comportamiento de la perdida(loss) durante el proceso de entrenamiento. Cabe resaltar que el eje Y corresponde al valor de la perdida, mientras que el eje X pertenece al número de épocas entrenadas por la red, las cuales fueron 100, Para mayor información se pueden visualizar los anexos X y XI sobre el comportamiento de la red en perdida de clasificación y regresión.

El código de entrenamiento de la red neuronal empleó un auto ajuste en el parámetro de aprendizaje (LR) que al finalizar cada época de entrenamiento se adaptó el LR con un valor inferior al de la época terminada, con el objetivo de reducir la perdida(loss).

Posteriormente, se cargó el modelo entrenado en la variable “model”(véase Fig. 95), la cual se transformó en modelo compatible con el paquete contenedor de la red neuronal.

```
model = models.load_model(modelo_entrenado, backbone_name='resnet50')
model = models.convert_model(model)
```

Fig. 95. Generación del modelo neuronal

Después se utilizó una función que predijo las regiones de interés, los porcentajes de detección y las coordenadas del recuadro también llamadas “boxes” en imágenes de validación. El funcionamiento de este método consistió en la utilización del modelo entrenado para efectuar predicciones. Para esto se usó el comando “model.predict_on_batch(véase Fig. 96), el cual recibió como parámetro de entrada la imagen de validación con preprocesamiento Keras. Luego se conFig.uró el umbral de confianza en 0,8 para generar el contorno de dibujo.

```
import skimage.io as io

def prediccion_imagen(image):

  imagen = preprocess_image(image.copy())
  imagen, scala = resize_image(image)

  #BOX      SCORES      LABELS
  cajas, puntuaciones, etiquetas = model.predict_on_batch(
    np.expand_dims(image, axis=0)
  )

  cajas /= scala

  return cajas, puntuaciones, etiquetas
```

Fig. 96. Función de predicción.

Seguidamente se utilizó otra función, la cual dibujó los contornos de detección en base a los resultados obtenidos por la predicción de la red. Este método empleó un condicional “if” (véase Fig. 97), el cual verificó el valor del “score” con respecto al umbral de detección. Si el condicional era verdadero, todo el proceso de detección era finalizado. No obstante, si era falsa, la función dibujaba el contorno en la región de interés con el comando “draw_caption”(véase Fig. 97) , el cual recibía como parámetro la imagen analizada, contorno de detección en forma de caja y la información de etiqueta y porcentaje de predicción.

```
def dibujo_contorno(image, cajas, puntuaciones, etiquetas):
  for box, score, label in zip(cajas[0], puntuaciones[0], etiquetas[0]):

    # CONDICIONAL QUE VERFICA SI LA FUNCION EFECTUA EL DIBUJO
    if score < umbralScore:
        break

    color = label_color(label)

    b = box.astype(int)
    draw_box(image, b, color=color)

    caption = "{} {:.3f}".format(etiquetas_nombre[label], score)
    draw_caption(image, b, caption)
```

Fig. 97. Función de detección.

Por último se utilizó una instrucción de repetición “for”(ver Fig. 98), la cual recorrió todo el archivo de validación denominado “test_df”(véase Fig. 98). Este archivo contuvo todas las imágenes de validación y cada etiqueta de las mismas. No obstante, como se puede observar en la Fig. 98 solo se utilizaron las imágenes de validación con el uso de la variable “row”, en la cual se almacenaron las representaciones graficas con cada iteración del bucle, asimismo estas imágenes eran del formato “.Png”, por lo cual se realizó el acotado de canales de color(cambio de 4 a 3 canales) para generar la predicción. Por ultimo este método empleó el rol de unificar todas las funciones anteriormente vistas, generando las predicciones, dibujo del contorno y la exposición de los resultados.

```
# BUCLE QUE RECORRE LAS IMAGENES DE VALIDACION Y POSTERIOR PREDICCION, DIBUJO CONTORNO

for indice, row in test_df.iterrows():
  print(row[0], indice)

  image = io.imread(row[0])
  image = image[:,:,0:3]

  # BOX   SCORES         LABELS
  cajas, puntuaciones, etiquetas = prediccion_imagen(image)

  draw = image.copy()
  dibujo_contorno(draw, cajas, puntuaciones, etiquetas)

  plt.axis('off')
  plt.imshow(draw)
  plt.show()
```

Fig. 98. Predicción y detección de validación.

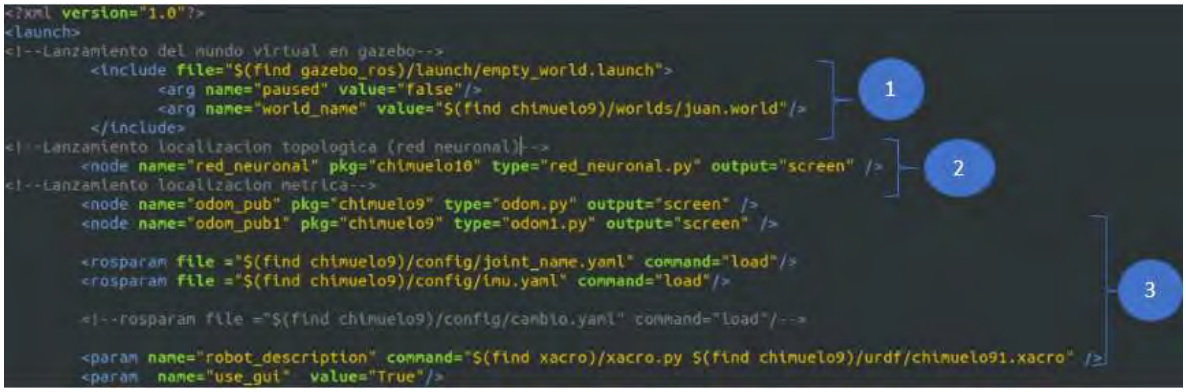


Fig. 99. Integración de los sistemas de localización

Como se puede observar en la imagen 99 existen 3 secciones de código, las cuales ejecutaron la localización métrica y topológica en conjunto. La primera sección lanzó el mundo virtual en gazebo, la segunda ejecutó el sistema de localización topológica y la tercera ejecutó la localización métrica. Cabe resaltar que todo el sistema de localización (métrico y topológico) funcionó mediante el sistema meta operativo robótico ROS, el cual por medio de tópicos contenidos en los nodos de lanzamiento (véase Fig. 99) envió información a terminales y visualizadores Rviz.

Además, al momento de ejecutar el archivo ".launch" de la Fig. 99, se activan múltiples nodos de programación de ROS, los cuales envían información de la odometría real y virtual al filtro de Kalman. Finalmente, el resultado del filtro se envía al visualizador Rviz para su exposición(localización métrica). De igual forma, la localización topológica es generada por medio de la activación de dos nodos de programación de ROS, el cual ejecuta la imagen en tiempo real de la cámara web y pasa como parámetro de entrada las imágenes(frame) a la red neuronal convolucional para realizar la predicción, posteriormente se muestran las representaciones graficas con la predicción hecha por la red en una ventana del monitor, tal cual como se muestra en las Fig.s 105,106,107 y 108.

La ejecución y visualización simultanea de ambos sistemas de localización, no solo permitió clasificar los objetos que componen el lugar donde se encuentra el robot, sino también conocer el punto en el mapa donde está ubicado. La integración de estos sistemas permitirá en un futuro la creación e implementación de un método que pueda generar rutas de navegación para la guía de personas al interior del sótano dos de la Universidad Autónoma de Occidente.

6.5 ETAPA 5: VALIDACION

6.5.1 Actividad 9 y 10: test de funcionamiento resultados finales

Visualización del recorrido del robot en un ambiente simulado. La visualización de la fusión sensorial compuesta por odometría virtual y real, fue generada por un paquete adicional llamado “odom_to_trajectory”, el cual es un complemento de ROS. Este paquete tomó como entrada los resultados del filtro de Kalman (RobotPoseEkf), los cuales se enviaron por medio de tópicos hacia el visualizador RVIZ, como se puede ver en la Fig. 100.

En la Fig. 100, se observa la trayectoria del robot en línea recta, con la particularidad de evidenciar la odometría real, virtual y estimada. La odometría real del robot se representa con el color rojo, la cual fue extraída mediante la tarjeta electrónica embebida Arduino. Por otro lado, la odometría virtual, en este caso visible con el color amarillo, es generada por el simulador virtual gazebo. Con estas dos fuentes de información, se realizó el cálculo de la odometría estimada mediante uso del filtro de Kalman(RobotPoseEkf). Esta nueva odometría, se puede identificar con el color verde.

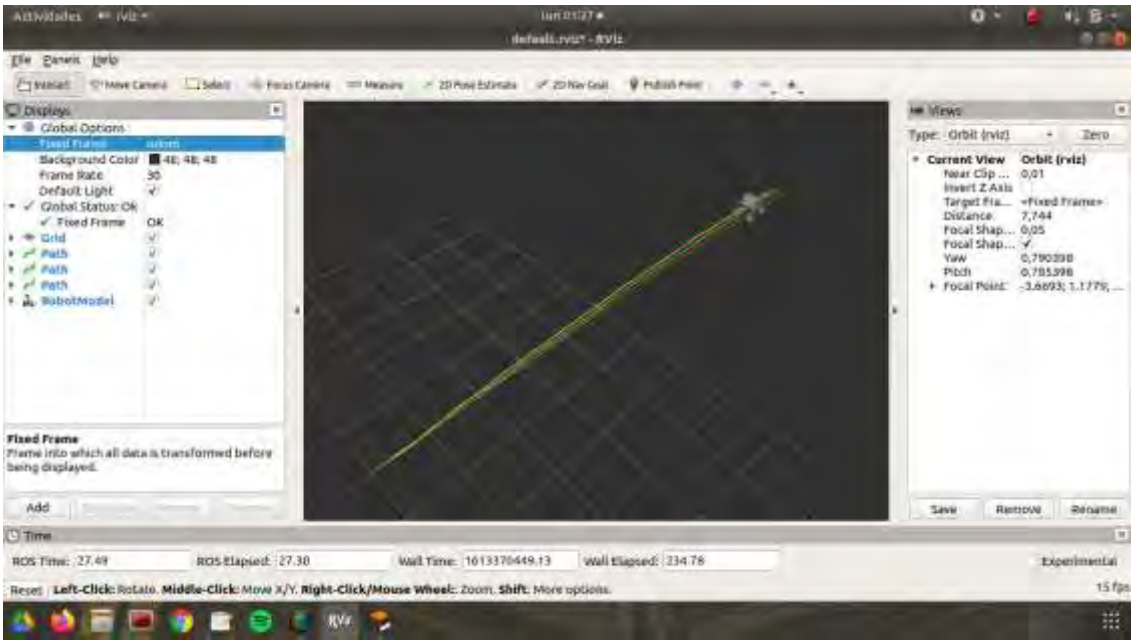


Fig. 100. Línea de odometría fusionada.

Fig. 101. Línea de prueba 6Mtrs

Fig. 102. Línea de prueba 15.92Mtrs

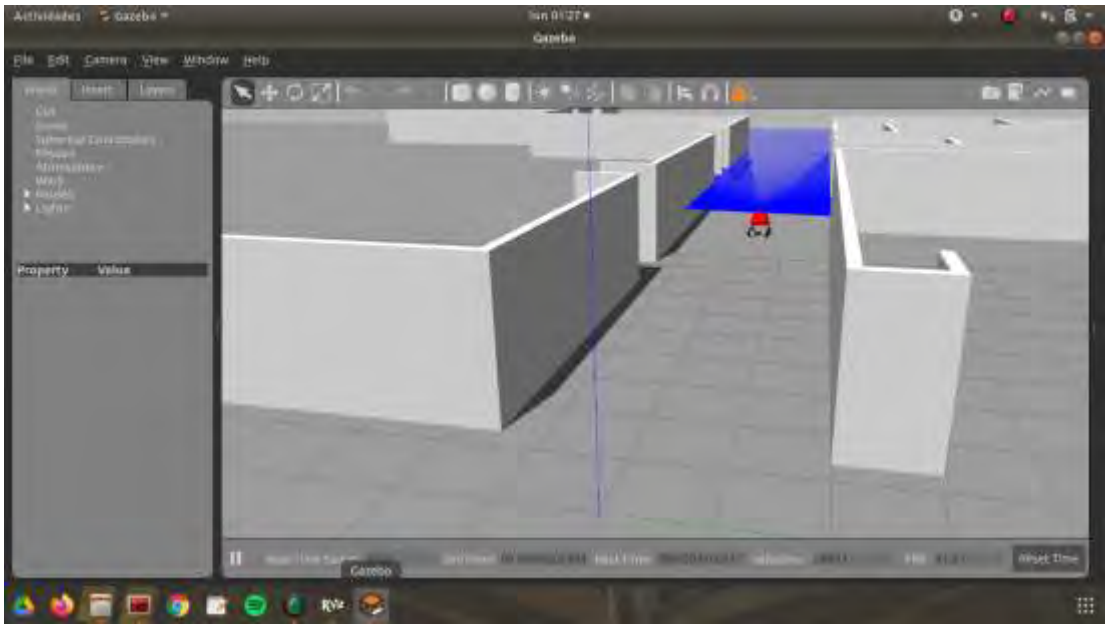


Fig. 103. Simulador de gazebo con robot y sótano 2 en 3D.

En el tabla de anexo A,B y C, se puede observar los datos de una prueba realizada en línea a lo largo del pasillo de electrónica (ver Fig.s 101 y 102). Además, se puede observar la simulación del robot en la Fig. 103, el cual tiene una medida de 27.99 metros de longitud y su respectivo error absoluto y relativo

En la tabla de anexo D,E,F, G, H e I, se puede evidenciar las pruebas realizadas a cuatro distancias diferentes (1Mt , 6Mts ,15.92Mts y 27.9Mts) con el objetivo de verificar cuanto seria el error en una posición determinada del robot.

Resultados del sistema de localización Topológica

Como se puede observar, el desempeño de la red es consistente con lo esperado, está CNN identifica de manera exitosa diferentes puntos o localizaciones a lo largo del sótano 2 de la Universidad Autónoma de Occidente, identificando estas ubicaciones mediante objetos o características claves, gracias a las cuales la seguridad que tiene el robot de estar en un lugar determinado por la SLT es muy alta, ya que siempre se encuentra en una probabilidad de detección superior al 89%(ver Fig. 104)

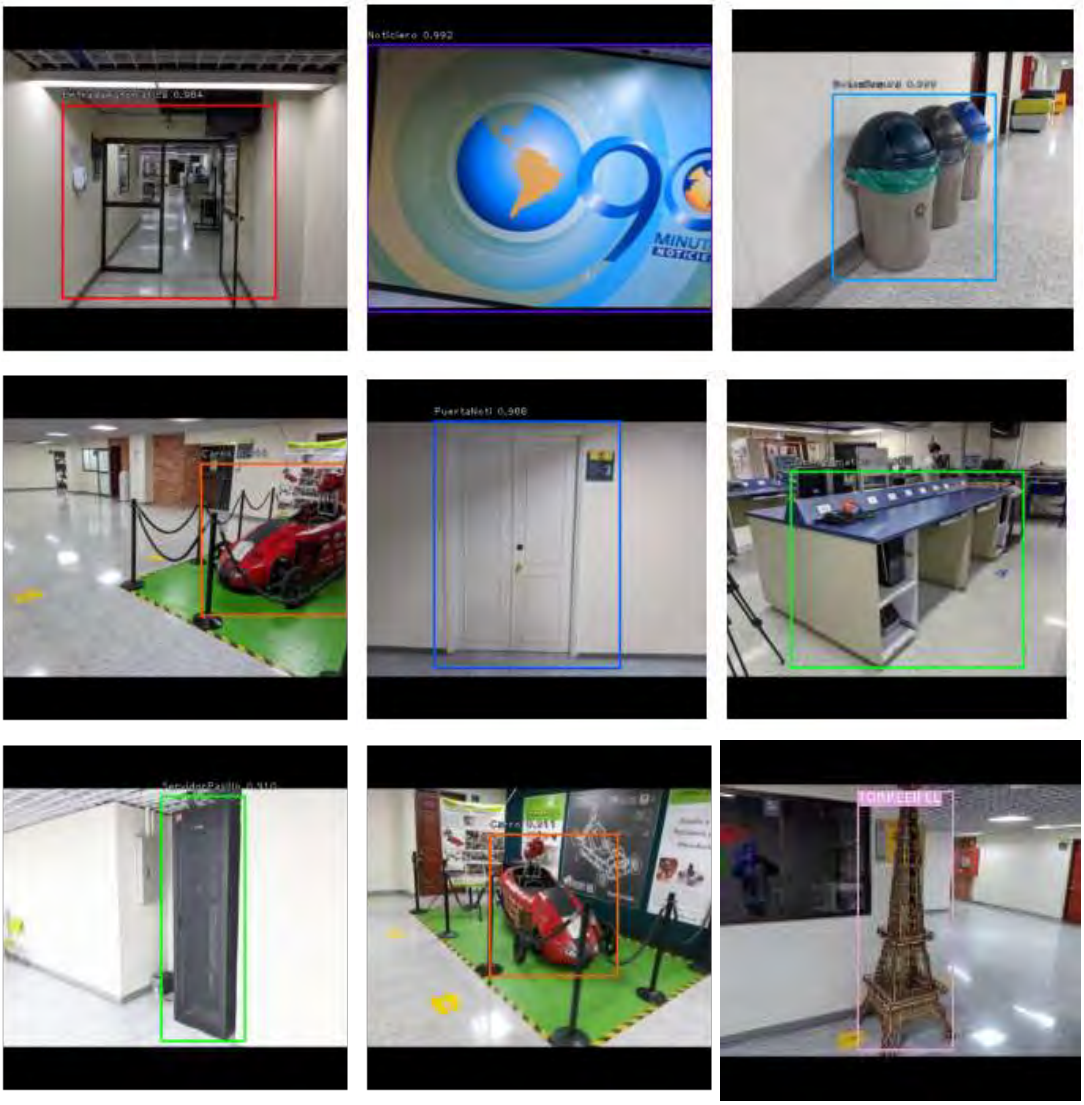


Fig. 104. Resultados de red neuronal.

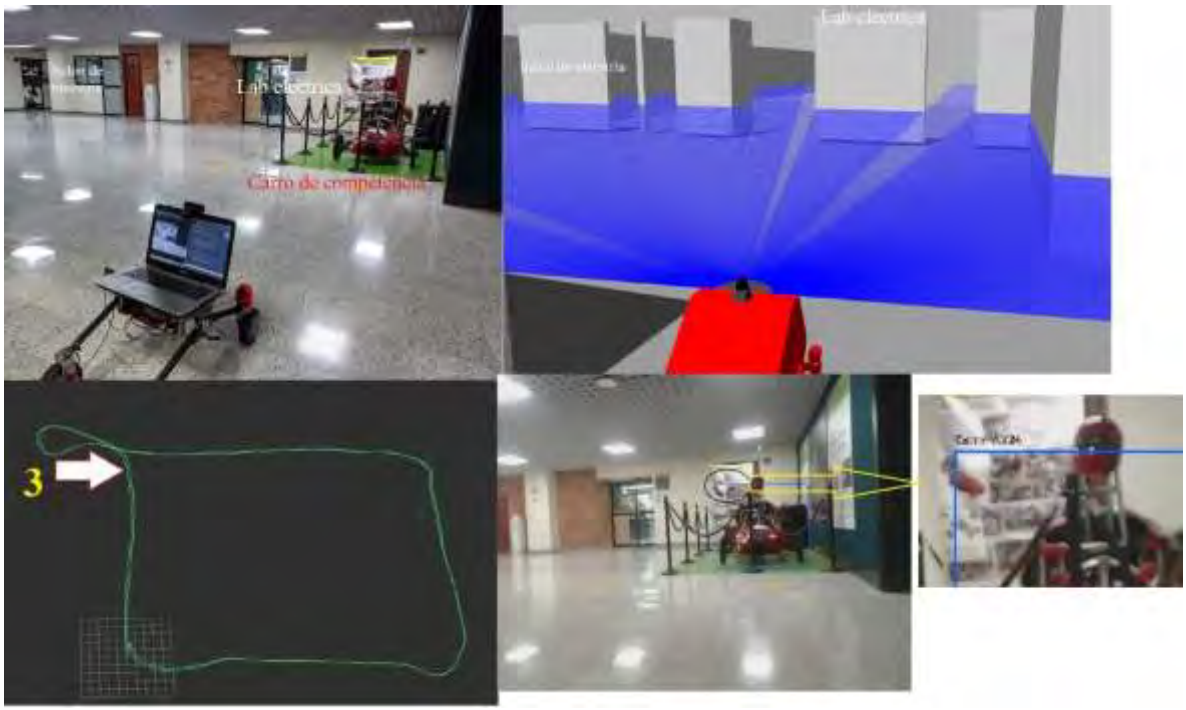
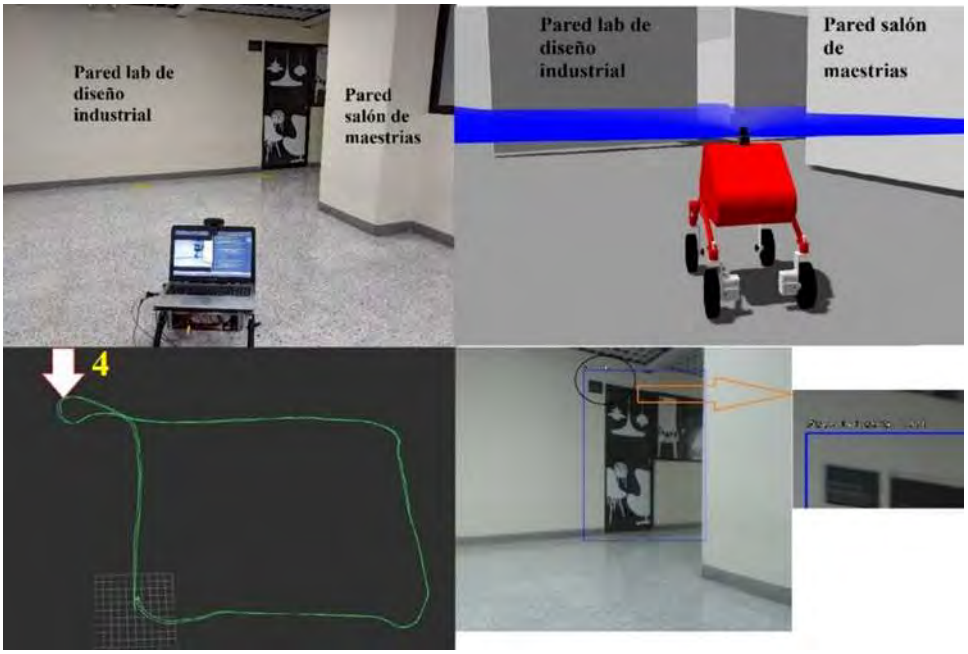
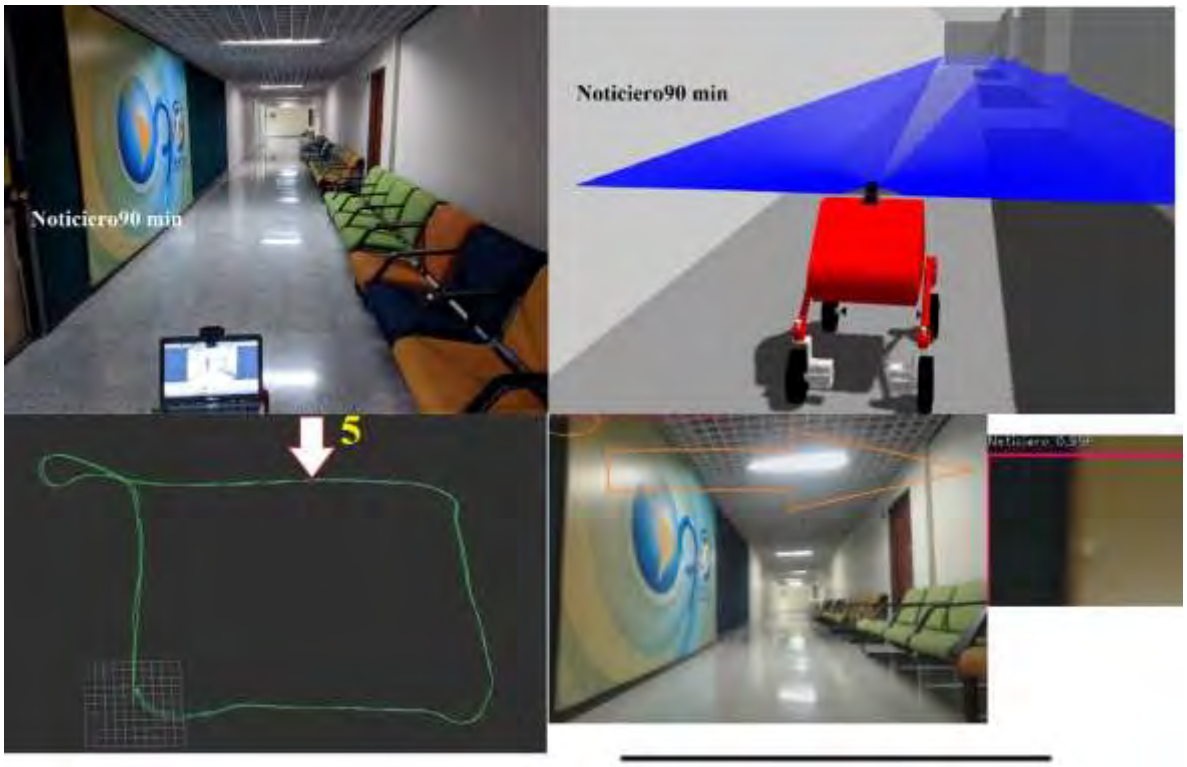


Fig. 105. Localización métrica y topológica sótano dos(1)



En la Fig. 105, 106 y 107 se puede observar en la parte superior izquierda una foto real de un punto de identificación y a su derecha el simulado; en la parte inferior, el punto de recorrido (localización métrica) y en la izquierda la identificación semántica (localización topológica)

Fig. 107. Localización métrica y topológica(3)



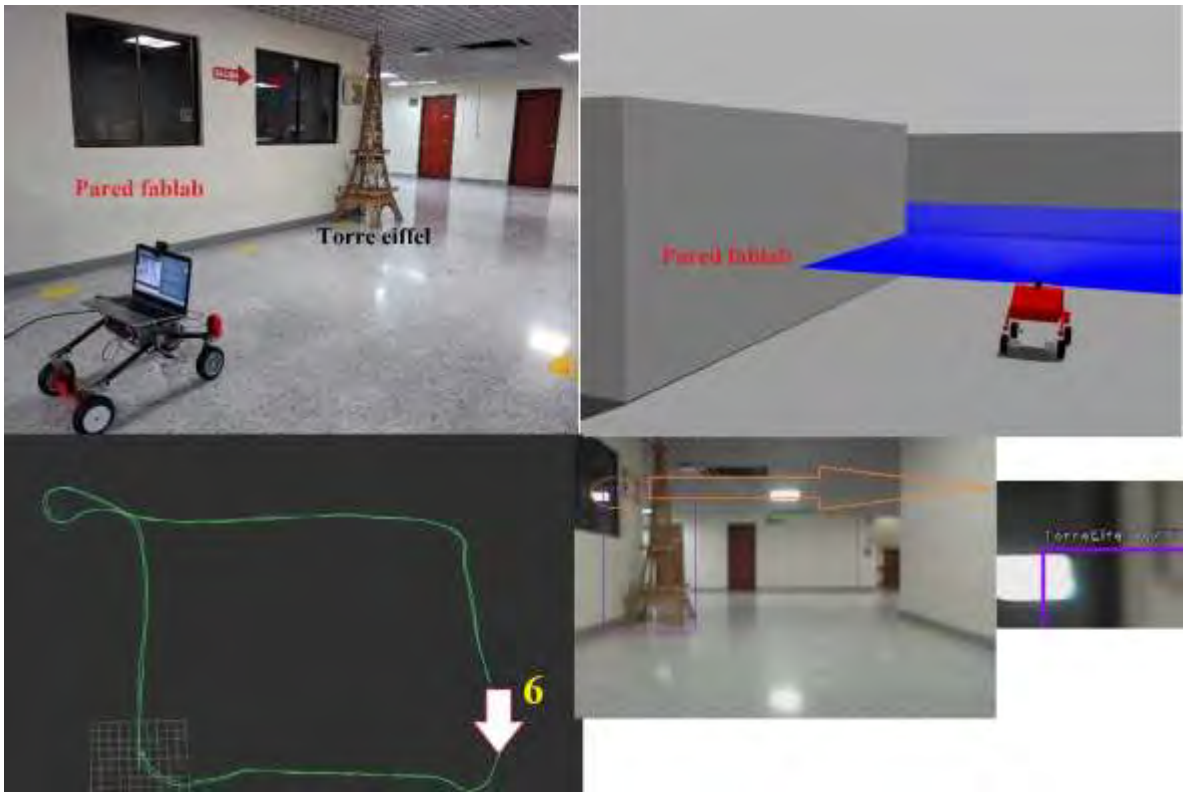


Fig. 108. Localización métrica y topológica(4)

7. CONCLUSIONES

Gracias a las increíbles ventajas que nos ofrece el software libre, fue posible llevar a cabo el modelado, simulación, desarrollo y puesta en marcha del sistema de localización métrica y topológica en ambientes semiestructurados. Para el primer sistema (métrico) fue necesario seleccionar la plataforma robótica en la que se iba a implementar. En este caso fue en chimuelo, un robot móvil tipo Ackermann propio de la Universidad Autónoma de Occidente, el cual fue dotado de una serie de sensores propioceptivos como lo son el IMU y los encoders, ya que son necesarios por el sistema desarrollado, tanto por sus ventajas o por los buenos resultados que ofrecen al ser fusionarlos mediante el filtro de Kalman.

Además, se encontró diversos inconvenientes en la instrumentación de los sensores a utilizar, tales como el formato de envió de información de estos o como se debía fusionar para obtener el resultado esperado, ya que este proceso es clave en el desarrollo del proyecto. Este fue uno de los principales retos, ya que fue necesario convertir la información y los formatos adaptándolo a las necesidades del proyecto.

También, para la realización de las pruebas del sistema de localización métrico, fue necesario los planos arquitectónicos del sótano de la universidad, con el fin de llevar a cabo el modelado 3D de este y finalmente, en la evaluación del desempeño se obtuvo un error que oscila entre el 0.2% y el 6.8%.

Adicionalmente, el sistema de localización topológica (SLT), se diseñó e implementó de manera exitosa en un computador con el sistema meta-operativo ROS, de acuerdo a las condiciones del lugar de trabajo (sótano dos (2) de la Universidad Autónoma de Occidente), tales como intensidad de luz, número de objetos a identificar y restricciones de movilidad del robot chimuelo. Para esta caracterización se examinó cada pasillo del entorno, y se encontraron varios objetos de interés para ser identificados con el SLT. De igual manera, se evidenció que varias zonas del entorno carecían de luz y algunos objetos de interés estaban inclinados, lo que representaba un problema para el entrenamiento de la red neuronal convolucional, ya que las detecciones de objetos podrían verse comprometidas por un cambio de intensidad de luz o por la inclinación del objeto a identificar. No obstante, este problema se soluciona con un preprocesamiento del dataset de entrenamiento llamado “data Augmentation”. Mediante este método se generaron nuevas imágenes con las condiciones del entorno (cambios de luz e inclinaciones de imágenes), esto con el fin de entrenar a la CNN con cualquier estado de los objetos que pudiesen generar un problema.

Por otro lado, la red neuronal convolucional inicialmente presentó múltiples fallas en su entrenamiento, evidenciándose en los errores de identificación de múltiples objetos, lo cual no permitía ubicar de manera adecuada el lugar en el que se encontraba. Este problema se dio en gran parte por mala conFig.ción de los hiper-parámetros de entrenamiento, tales como LR por sus siglas en inglés(learning rate) y el número de imágenes de entrenamiento. Por lo tanto, se amplió el dataset de entrenamiento mediante la toma de nuevas imágenes y así mismo la conFig.ción del LR con un valor muy pequeño, lo que llevó un entrenamiento mucho más lento. Esta solución permitió una considerable mejora en el rendimiento de la red y de esta manera identificar correctamente el lugar en el que se encuentra con un error mínimo.

Adicionalmente, una de las ventajas de nuestro SLT, es la fácil comunicación entre sistemas externos, ya que trabaja con el sistema operativo robótico (ROS), el cual permite la emisión y recepción de datos de cualquier tipo. Por lo cual, el SLT facilita el envío de todas las predicciones de las imágenes de entrada, tales como región, porcentaje y etiqueta de la detección hacia otros sistemas por medio de tópicos de ROS. Esto en un futuro desarrollo, permitirá la integración con múltiples sistemas, un ejemplo de esto podría ser la guía de personas dentro de las instalaciones de la universidad, más específicamente en el sótano dos(2) de la Universidad Autónoma de Occidente. Además, el SLT podría conectarse a un sistema de sonido por altavoz, el cual al detectar objetos claves en las imágenes de entrada procederá con el envío de información relacionada con los lugares cercanos reproduciéndose por un altavoz, siendo una capacidad muy útil para la orientación o guía de personas con discapacidad visual.

Finalmente se puede decir que el desarrollo del proyecto le agrega valor y sirve como una base para futuros proyectos.

REFERENCIAS

[1] Q. Lin, X. Liu and Z. Zhang, "Mobile Robot Self-LocalizationUsing Visual Odometry Based on Ceiling Vision," 2019 IEEE Symposium Series on Computational Intelligence (SSCI), 2019, pp. 1435-1439, doi: 10.1109/SSCI44817.2019.9003092.

[2] C. Lee, H. Lee, I. Hwang and B. Zhang, "Spatial Perception by Object-Aware Visual Scene Representation", Openaccess.thecvf.com, 2019. .[En línea]. Disponible en: https://openaccess.thecvf.com/content_ICCVW_2019/html/DL4VSLAM/Lee_Spatial_Perception_by_Object-Aware_Visual_Scene_Representation_ICCVW_2019_paper.html.

[3] A. Davison, "FutureMapping: The Computational Structure of Spatial AI Systems", arXiv.org, 2018. .[En línea]. Disponible en: https://arxiv.org/abs/1803.11288.

[4] J. L. Schonberger, J.-M. Frahm, "Structure-from-motion revisited", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4104-4113, 2016.

[5] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, J. J. Leonard, "Past present and future of simultaneous localization and mapping: Toward the robust-perception age", IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1309-1332, 2016.

[6] H. Durrant-Whyte, T. Bailey, "Simultaneous localization and mapping (SLAM): Part I", IEEE Robotics & Automation Magazine, vol. 13, no. 2, pp. 99-110, 2006.

[7] Pileun Kim, Jingdao Chen, Yong K. Cho,SLAM-driven robotic mapping and registration of 3D point clouds,Automation in Construction,Volume 89,2018,Pages 38-48,ISSN 0926-5805,https://doi.org/10.1016/j.autcon.2018.01.009.

[8] M. Olivares Ávila and J. Gallardo Arancibia, "sistema de localización autónoma para robots móviles basado en fusión de sensores propioceptivos", Revista Politécnica ISSN 2256-5353 (En línea), 2015. Disponible en: http://revistas.elpoli.edu.co/index.php/pol/article/download/621/597.

[9] L. Enciso Salas, "Diseño de un sistema de navegación autónomo para robots móviles usando fusión de sensores y controladores neuro difusos", Tesis.pucp.edu.pe, 2015. .[En línea]. Disponible en: http://tesis.pucp.edu.pe/repositorio/handle/20.500.12404/6191.

[10] C. Rojas Cardenas, "Precalculo matematicas para el calculo", Academia.edu, 2010. .[En línea]. Disponible en:https://www.academia.edu/38431347/Precalculo_matematicas_para_el_calculo

[11] K. Academy, "Integrales triples en coordenadas esféricas (artículo) | Khan Academy", Khan Academy, 2021. .[En línea]. Disponible en: https://es.khanacademy.org/math/multivariable-calculus/integrating-multivariable-functions/x786f2022:polar-spherical-cylindrical-coordinates/a/triple-integrals-in-spherical-coordinates.

[12] D. Fisica aplicada III, Universidad de sevilla, "Coordenadas cilíndricas. Definición", Laplace.us.es, 2007. .[En línea]. Disponible en: http://laplace.us.es/wiki/index.php/Coordenadas_cil%C3%ADndricas._Definici%C3%B3n.

[13] SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

[14] SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

[15] SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004.

[16] SIEGWART Roland, Nourbakhsh Illah R. Introduction to Autonomous Mobile Robots. 1 ed. Cambridge, London: The MIT Press, 2004. p25

[17] V. Lidar, "HDL-64E Durable Surround Lidar Sensor | Velodyne Lidar", Velodyne Lidar, 2021. .[En línea]. Disponible en: https://velodynelidar.com/products/hdl-64e/.

[18] Encoder ¿Como funciona? Y sus tipos: Como funciona un encoder [en linea] ingenieria mecafenix. .[En línea]. Disponible en: https://www.ingmecafenix.com/automatizacion/encoder/.

[19] "CSCI 2150 -- Digital Signals and Binary Numbers", Faculty.etsu.edu, 2021. .[En línea]. Disponible en: https://faculty.etsu.edu/tarnoff/ntes2150/dig_bin/dig_bin.htm.

[20] Amaris Gonzalez, Marcos. (2009). Aplicacion en Scilab para el tratamiento de señales digitales con la transformación de Fourier fraccionaria: Filtro de Wiener fraccionario. 10.13140/RG.2.2.19520.94727.

[21]M. Cervantes, "Señales", Programas.cuaed.unam.mx, 2017. .[En línea]. Disponible en: https://programas.cuaed.unam.mx/repositorio/moodle/pluginfile.php/824/mod_resource/content/5/contenido/index.html.

[22]J. Rojas, "servomotores para arduino uno", Es.slideshare.net, 2018. [Online]. Available: https://es.slideshare.net/NelsonRojasGonzales/servomotores-para-arduino-uno.

[23] A.M Karadeniz, M. Alkayyali y P.T Szemes. “Modelling and Simulation of Stepper Motor For Position Control Using LabVIEW,”, Department of Mechatronics, University of Debrecen, Debrecen, Hungary, No. #1, En ,7,2018.[En línea]. Disponible en: https://www.researchgate.net/Fig.ure/Schematic-view-of-a-stepper-motor-with-three-teeth4_Fig.1_324261648

[24] Robotic Operative System(Ros), (2020,Jul.10). ”Ros.org| Powering the world’s robots ”. [Internet]. Disponible en: https://www.ros.org/

[25] AristaSur, (2020, Jul, 11). “Cómo funciona el Sistema de posicionamiento GPS”.[Internet]. Disponible en: https://www.aristasur.com/contenido/como-funciona-el-sistema-de-posicionamiento-gps

[26] M. Olinski, A. Gronowicz, M. Ceccarelli y D. Cafolla, “Human motion characterization using wireless inertial sensors,” Departamento de Biomedica, Mecatronica y teoria de mecanismos, Laboratorio de mecatrónica y robotica, Wroclaw University of Science and Technology and University of Cassino and South Latium, Wroclaw and Cassino , Poland and Italy, 2015. p. 329. Disponible en: https://www.researchgate.net/Fig.ure/IMU-sensor-with-a-scheme-of-the-measured-Euler-angles_Fig.5_308860567

[27] G. Verhoeven, M. Wieser, C. Briese y M. Doneus, “Positioning in Time and Space – Cost-Effective exterior orientation for airborne archaeological photographs”, vias – Vienna Institute for Archaeological Science, LBI for Archaeological Prospection and Virtual Archaeology and Department of Geodesy and Geoinformation, University of Vienna and Vienna University of Technology, Vienna and Wien, Austria, 2013. p. 314. Disponible en: https://www.researchgate.net/Fig.ure/Three-axes-and-rotations-of-a-digital-still-camera_Fig.1_256325054

[28] M. Hirz and B. Walzel, “Sensor and object recognition technologies for self-driving cars ”, Graz University of Technology, Graz, Estiria, Austria, 2018. p. 6. Disponible en: https://www.researchgate.net/Fig.ure/Point-cloud-of-a-traffic-situation-delivered-by-LIDAR-10_Fig.8_322368082

[29] stereo labs, (2020,Oct, 15). “ZED Stereo Camera| Stereolabs”. [Internet]. Disponible en: https://www.stereolabs.com/zed/

[30] N. Saparkhojayev and S. Guvercin, “Attendance Control System based on RFID-technology”, Department of Systems Analysis and Management, Engineering Faculty, L.N.Gumilyov Eurasian National University and Suleyman Demirel University, Nur-sultán, Kazakhstan, 2012. p. 227. Disponible en: https://www.researchgate.net/Fig.ure/The-workflow-of-RFID-technology_Fig.1_267404585

[31] B. Choi, J. Lee, Ju. Lee and K. Park, “Distributed Sensor Network Based on RFID System for Localization of Multiple Mobile Agents”, Department of Electrical Engineering and Korea Institute of Machinery and Materials, KAIST, Daejeon, Korea (South), 2011. p. 3. Disponible en: https://www.researchgate.net/Fig.ure/Concept-of-the-sensor-network-based-on-the-RFID-system-for-localization-of-multiple_Fig.3_220279110

[32] A. Islam, T. Hossan and Y. Min Jang, “Convolutional neural network scheme-based optical camera communication system for intelligent Internet of vehicles”, Department of Electronics Engineering, Kookmin University, Seoul, Korea(South), 2018. p. 11. Disponible en: https://www.researchgate.net/Fig.ure/Distance-calculation-using-a-stereo-image-from-a-stereo-camera-and-b-system-platform_Fig.8_323782015

[33] J. Ziegler, J. Gleichauf and C. Pfitzner, “RobotCup Rescue 2018 Team Description Paper AutonOHM”, Department of Electrical Engineering, TH Nuremberg Georg Simon Ohm, . Nuremberg, Bavaria, Germany, 2018. p. 4. Disponible en: https://www.researchgate.net/Fig.ure/Map-generated-by-the-ohm-tsd-slam_Fig.2_326580040

[34] S. Thrun, A. Bucken, W. Burgard, D. Fox, T. Frohlinghaus, D. Hennig, T. Hofmann, M. Krell and T. Schmidt, “Map Learning and High-Speed Navigation in RHINO”, Institut fur Informatik III and Computer Science Department, Universitat Bonn and Carnegie Mellon University, Bonn and Pittsburgh, North Rhine-Westphalia and Pensilvania, Germany and United Stated, 1996. P. 8. Disponible en: https://www.researchgate.net/publication/2436687_Map_Learning_and_High-Speed_Navigation_in_RHINO

[35] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II," in *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108-117, Sept. 2006. Disponible en: https://ieeexplore.ieee.org/document/1678144

[36] F. Martin, V. Matellán, P. Barrera y J. M. Cañas, “Localización basada en lógica y filtros de Kalman para robots con patas”, Grupo de Robótica, Universidad Rey Juan Carlos, Móstoles, Madrid, España, 2014. p. 7. Disponible en: https://www.researchgate.net/Fig.ure/Fig.-5-Modelo-de-movimiento-basado-en-la-odometria_Fig.5_233792295

[37] github,Udacity(2020, Oct, 20). “robot_pose_ekf package”. [Internet]. Disponible en: https://github.com/udacity/robot_pose_ekf

[38] Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, S. Nasrin, M. Hasan, B. C. Van Essen, A. S. Awwal and V. K. Asari, “A State-of-the-Art Survey on Deep Learning Theory and Architectures”, Department of Electrical and Computer Engineering, Department of Earth and Atmospheric Sciences, Comcast Labs and Lawrence Livermore National Laboratory, University of Dayton, Saint Louis University, Dayton, washington dc, Livermore, Ohio, District of Columbia and California, United States, 2019. p. 11. Disponible en: https://www.researchgate.net/Fig.ure/The-overall-architecture-of-the-Convolutional-Neural-Network-CNN-includes-an-input_Fig.4_331540139

[39] gazebosim, (2020, Nov, 1). “Gazebo| Robot simulation made easy”. [Internet]. Disponible en: http://gazebosim.org/

[40] github, noetic-devel(2020, Nov, 5). “ros-visualization/rviz: ROS 3D Robot Visualizer”. [Internet]. Disponible en: https://github.com/ros-visualization/rviz

[41] H. A. Al-Barazanchi, H. Qassim and A. Verma, “Residual CNDS”, Department of Computer Science, University California, Fullerton, California, United States,2016, p. 2. Disponible en: https://www.researchgate.net/Fig.ure/Residual-Connection-22_Fig.2_305994981

[42] "ResNet (34, 50, 101): Residual CNNs for Image Classification Tasks", Neurohive.io,Saint-Petersburg, Vyborgskaya nab, Russia, Jan, 23, 2019. [En línea]. Disponible en: https://neurohive.io/en/popular-networks/resnet/.

[43] "ResNet (34, 50, 101): Residual CNNs for Image Classification Tasks", Neurohive.io,Saint-Petersburg, Vyborgskaya nab, Russia, Jan, 23, 2019. [En línea]. Disponible en: https://neurohive.io/en/popular-networks/resnet/.

ANEXOS

Anexo A. Prueba en eje X

EJE X					
# DE MUESTRAS	ODOMETRIA ESTIMADA	ODOMETRIA VIRTUAL	ODOMETRIA REAL	ERROR ABSOLUTO	ERROR RELATIVO%
1	0,00022	0,000224	0,000224	0,000004	1,785714286
2	0,60995	0,71158	0,610817	0,000867	0,1419410396
3	0,812122	0,9148	0,809875	0,002247	0,2774502238
4	1,109063	1,113034	1,08730	0,021763	2,001563506
5	1,31044	1,315185	1,27712	0,03332	2,608995239
6	1,61192	1,61814	1,5083	0,10362	6,869986077
7	2,6154	2,598	2,605	0,0104	0,3992322457
8	2,8145	2,8205	2,7134	0,1011	3,725952679
9	3,1163	3,127	3,1	0,0163	0,5258064516
10	4,4232	4,438	4,3178	0,1054	2,441057946
11	4,6257	4,6409	4,5999	0,0258	0,5608817583
12	4,9261	4,9423	4,8215	0,1046	2,169449341
13	5,12665	5,2044	5,089	0,03765	0,7398310081
14	5,4279	5,4469	5,3239	0,104	1,953455174
15	5,9523	5,9956	6,022	0,0697	1,157422783
16	6,4352	6,4	6,33113	0,10407	1,64378239
17	6,94099	6,9611	6,8334	0,10759	1,574472444
18	8,24608	8,27332	8,2	0,04608	0,5619512195
19	8,75188	8,7799	8,6981	0,05378	0,6182959497
20	10,8647	10,89926	10,7986	0,0661	0,6121163855
21	11,167	11,2018	11,1623	0,0047	0,04210601758
22	13,182	13,2182	13,277	0,095	0,715523085
23	15,5012	15,5381	15,5	0,0012	0,007741935484
24	17,0105	17,0545	16,9092	0,1013	0,5990821565
25	19,8359	19,8865	19,63	0,2059	1,048904738
26	21,4738	21,532	21,4726	0,0012	0,005588517459
27	26,55	26,8935	26,95	0,4	1,484230056
	X	X			

Anexo B. Prueba en eje Y

EJE Y					
# DE MUESTRAS	ODOMETRIA ESTIMADA	ODOMETRIA VIRTUAL	ODOMETRIA REAL	ERROR ABSOLUTO	ERROR RELATIVO
1	0,0000658	0,0000681	0,0000634	0,0000024	3,785488959
2	0,0144	0,0106	0,0149	0,0005	3,355704698
3	0,01647	0,01211	0,016999	0,000529	3,111947762
4	0,044690	0,013410	0,04594	0,00125	2,720940357
5	0,05340	0,01431	0,05498	0,00158	2,873772281
6	0,05857	0,01678	0,06149	0,00292	4,748739632
7	0,04434	0,000727	0,0465	0,00216	4,64516129
8	0,05501	0,02094	0,0576	0,00259	4,496527778
9	0,06798	0,02021	0,071	0,00302	4,253521127
10	0,06769	0,01338	0,0711	0,00341	4,796061885
11	0,06282	0,01933	0,0645	0,00168	2,604651163
12	0,06	0,0081	0,0631	0,0031	4,912836767
13	0,08193	0,0035	0,085	0,00307	3,611764706
14	0,1047	0,0001035	0,102	0,0027	2,647058824
15	0,08503	0,02091	0,0881	0,00307	3,484676504
16	0,07493	0,02129	0,07706	0,00213	2,764079938
17	0,056	0,42	0,05871	0,00271	4,615908704
18	0,0412	0,09144	0,04323	0,00203	4,695813093
19	0,01704	0,1141	0,0179	0,00086	4,804469274
20	0,042	0,2326	0,0441	0,0021	4,761904762
21	0,05624	0,2527	0,059	0,00276	4,677966102
22	0,1741	0,3991	0,1821	0,008	4,393190555
23	0,31242	0,5875	0,3224	0,00998	3,095533499
24	0,40092	0,64792	0,42	0,01908	4,542857143
25	0,4	0,5	0,39221	0,00779	1,986180872
26	0,4059	0,66	0,3967	0,0092	2,319132846
27	0,37	0,47	0,36	0,01	2,777777778
28	0,19	0,36	0,189	0,001	0,5291005291

Anexo C. Prueba eje Z de orientacion

EJE Z (ORIENTACION)					
# DE MUESTRAS	ESTIMADA	VIRTUAL	REAL	ERROR ABSOLUTO	ERROR RELATIVO
1	-0,0054	-0,0004	-0,0056	0,0002	-3,571428571
2	-0,0106	-0,0047	-0,0133	0,0027	-20,30075188
3	-0,023	-0,0046	-0,0315	0,0085	-26,98412698
4	-0,0631	-0,0044	-0,0848	0,0217	-25,58962264
5	-0,0033	-0,0037	-0,0367	0,0334	-91,00817439
6	-0,0221	-0,0032	-0,044	0,0219	-49,77272727
7	-0,0247	-0,0004	-0,0315	0,0068	-21,58730159
8	-0,09	-0,0002	-0,063	0,027	-42,85714286
9	-0,0298	-0,0002	-0,0543	0,0245	-45,11970534
10	-0,0152	-0,009	-0,018	0,0028	-15,55555556
11	-0,0313	-0,0058	-0,0602	0,0289	-48,00664452
12	-0,0845	-0,0072	-0,0533	0,0312	-58,53658537
13	-0,0045	-0,0085	-0,0074	0,0029	-39,18918919
14	-0,0428	-0,0096	-0,09	0,0472	-52,44444444
15	-0,0262	-0,0145	-0,0397	0,0135	-34,00503778
16	-0,0112	-0,0169	-0,0213	0,0101	-47,41784038
17	-0,0023	-0,0213	-0,0059	0,0036	-61,01694915
18	-0,0606	-0,0233	-0,0631	0,0025	-3,961965135

19	-0,0679	-0,0247	-0,0476	0,0203	-42,64705882
20	-0,022	-0,0279	-0,036	0,014	-38,88888889
21	-0,0163	-0,0322	-0,0239	0,0076	-31,79916318
22	-0,0769	-0,0333	-0,0557	0,0212	-38,06104129
23	-0,0123	-0,0373	-0,0176	0,0053	-30,11363636
24	-0,1233	-0,0417	-0,171	0,0477	-27,89473684
25	-0,0112	-0,001	-0,0387	0,0275	-71,05943152
26	-0,0036	-0,0023	-0,0422	0,0386	-91,46919431
27	-0,0406	-0,0063	-0,1079	0,0673	-62,37256719
28	-0,0364	-0,0028	-0,0235	0,0129	-54,89361702

Anexo D. Prueba a 1Metro eje X

EJE X						
# DE MUESTRAS	ODOM ESTIMADA	MEDIDA REAL	ERROR ABSOLUTO	ERROR RELATIVO %		DESVIACION ESTANDAR
1	1,0491	1	0,0491	4,9063	8,12E-06	0,009826233304
2	1,0543	1	0,0543	5,4300	5,70E-06	
3	1,0310	1	0,0310	3,1000	4,37E-04	
4	1,0418	1	0,0418	4,1800	1,02E-04	
5	1,0834	1	0,0834	8,3400	9,91E-04	
MEDIA ARIT.	1,0519					

Anexo E. Prueba a 6Metro eje X

EJE X						
# DE MUESTRAS	ODOM ESTIMADA	MEDIDA REAL	ERROR ABSOLUTO	ERROR RELATIVO %		DESVIACION ESTANDAR
1	6,0993	6	0,0993	1,6550	0,0006739216	0,08586400876
2	6,1338	6	0,1338	2,2300	0,0036554116	
3	5,9794	6	0,0206	0,3433	0,0088247236	
4	5,9868	6	0,0132	0,2200	0,0074891716	
5	6,1674	6	0,1674	2,7900	0,0088472836	
MEDIA ARIT.	6,07334					

Anexo F. Prueba a 15.92Mteros eje X

EJE X						
# DE MUESTRAS	ODOM ESTIMADA	MEDIDA REAL	ERROR ABSOLUTO	ERROR RELATIVO %		DESVIACION ESTANDAR
1	15,8198	15,92	0,1002	0,6293969849	0,0335402596	0,1905664136
2	15,8630	15,92	0,057	0,358040201	0,0196	
3	16,1990	15,92	0,279	1,752512563	0,0384	
4	15,9154	15,92	0,0046	0,02889447236	0,0076632516	
5	16,2175	15,92	0,2975	1,868718593	0,0460	
MEDIA ARIT.	16,00294					

Anexo G. Prueba a 27.99Mteros eje X

EJE X						
# DE MUESTRAS	ODOM ESTIMADA	MEDIDA REAL	ERROR ABSOLUTO	ERROR RELATIVO %		DESVIACION ESTANDAR
1	27,4394	27,9	0,4606	1,650896057	0,0362978704	0,4557455288
2	26,8795	27,9	1,0205	3,657706093	0,1364415844	
3	27,1228	27,9	0,7772	2,785663082	0,0158961664	
4	27,9471	27,9	0,0471	0,1688172043	0,4875111684	
5	26,8556	27,9	1,0444	3,743369176	0,1546691584	
MEDIA ARIT.	27,24888					

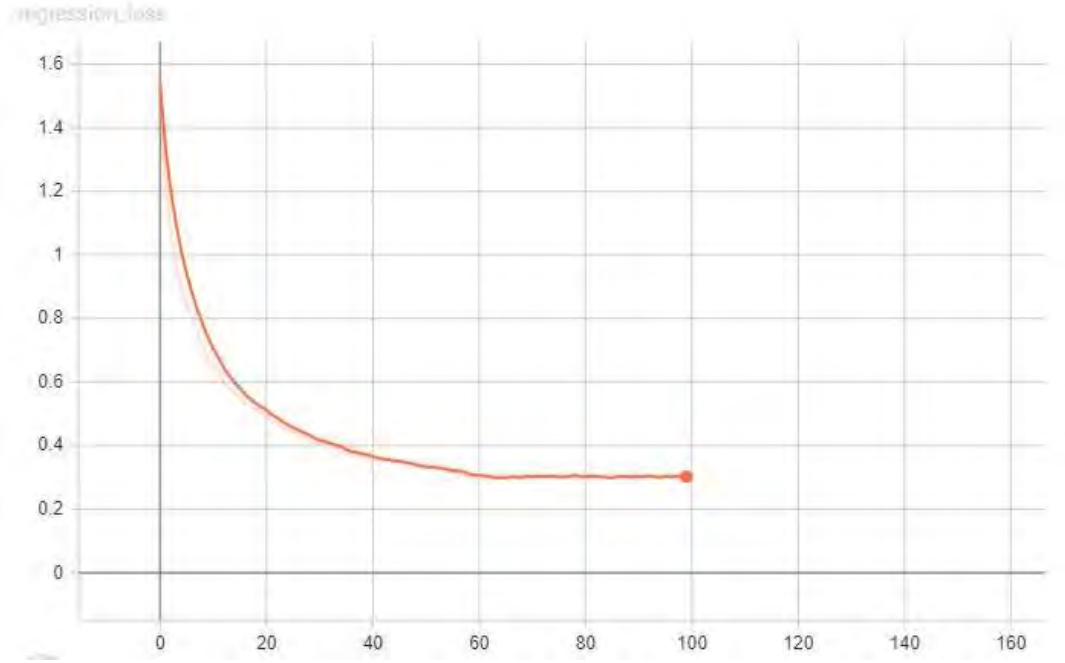
Anexo H. Prueba a 0Mteros de eje Y

EJE Y					
# DE MUESTRAS	ODOM ESTIMADA	MEDIDA REAL	ERROR ABSOLUTO		DESVIACION ESTANDAR
1	0,073900	0	0,0739	0,000311	0,04385908937
2	0,01304	0	0,01304	0,001868141284	
3	0,1241	0	0,1241	0,004601994244	
4	0,03193	0	0,03193	0,000592046224	
5	0,03834	0	0,03834	0,000321198084	
MEDIA ARIT.	0,056262				

Anexo I. Prueba a 0° de eje Z de orientacion

EJE Z ORIENTACION					
# DE MUESTRAS	ODOM ESTIMADA	MEDIDA REAL	ERROR ABSOLUTO		DESVIACION ESTANDAR
1	-0,0051	0	0,0051	0,0001085764	0,02198843332
2	-0,00918	0	0,00918	0,0000401956	
3	-0,05462	0	0,05462	0,00152881	
4	-0,0062	0	0,0062	0,0000868624	
5	-0,0025	0	0,0025	0,0001695204	
MEDIA ARIT.	-0,01552				

Anexo J. Regression_loss



Anexo K. Classification_loss