Implementación de Algoritmos Filtro de Kalman y Filtro de Partículas para Localización y Mapeo Simultáneo Aplicado a un Robot Móvil en Ambientes Interiores con Variaciones de Iluminación

Nicolás Álvarez Casadiego
Mario Armando Segura Albarracín

UNIVERSIDAD SANTO TOMÁS
FACULTAD DE INGENIERÍA ELECTRÓNICA
BOGOTÁ D.C.
2020

IMPLEMENTACIÓN DE ALGORITMOS FILTRO DE KALMAN Y FILTRO DE PARTÍCULAS PARA LOCALIZACIÓN Y MAPEO SIMULTÁNEO APLICADO A UN ROBOT MÓVIL EN AMBIENTES INTERIORES CON VARIACIONES DE ILUMINACIÓN

Nicolás Álvarez Casadiego
Mario Armando Segura Albarracín

Proyecto de grado presentado como requisito para optar al título de INGENIERO ELECTRÓNICO

Director: José Guillermo Guarnizo Marín

Codirector: Sindy Paola Amaya

UNIVERSIDAD SANTO TOMÁS DE AQUINO
FACULTAD DE INGENIERÍA ELECTRÓNICA
BOGOTÁ D.C.

2020

Dedicatoria

Nuestro trabajo está dedicado a Dios, quien supo darnos la sabiduría y voluntad para llegar a este punto tan importante acompañándonos en todo momento; En segunda instancia a nuestros padres por hacer con amor todo lo que estuvo a su alcance para facilitar nuestro trabajo; por último, a nuestros maestros por enseñarnos lo necesario en cada paso que dimos y a nosotros mismos por mantener la fe y persistencia en los momentos difíciles del largo trayecto que recorrimos.

AGRADECIMIENTOS

En primera instancia a Dios por permitirnos estar presentes para llevar a cabo nuestra formación académica y por darnos la sabiduría necesaria para escuchar nuestros corazones en los momentos de más complejidad, en segunda instancia a nuestros padres por apoyarnos a creer en cada uno de nosotros y brindarnos ese motivo de felicidad día a día para llegar hasta este punto.

Sencillo no ha sido el proceso, pero gracias a nuestros formadores, personas con gran comprensión, conocimiento y dedicación, acompañado del valioso tiempo dedicado en cada paso de este camino, se logró culminar nuestro proyecto de grado con éxito, pasión, amor y satisfacción propia.

Tabla de contenido

Índice de figuras

Índice de tablas

GLOSARIO

Algoritmo: Conjunto de instrucciones sistemáticas para la resolución de un problema, procesamiento de datos y resolución de tareas, mediante cálculos numéricos y metodológicos basados en variables tanto físicas como lógicas. Se obtiene una solución a partir de una entrada (estado inicial) llegando a una salida (estado final).

Error Cuadrático: Estimador del promedio de los errores al cuadrado, es decir, la diferencia entre el estimador y lo que se estima.

Filtro de Kalman: Método que permite estimar el estado de un proceso lineal discreto no observable a partir de un conjunto de medidas.

Filtro de Partículas: Algoritmo empleado comúnmente en visión artificial para estimar el estado de un sistema que cambia a lo largo del tiempo.

Iluminación en ambientes interiores: Partiendo del concepto nato la iluminación es la acción de iluminar y se desarrolla en efectos naturales (Proveniente del sol) como artificiales (proveniente de medios creados por el hombre); Para este proyecto de grado es de gran importancia resaltar la iluminación en ambientes interiores donde se presentan variaciones en la misma.

Localización y mapeo simultáneo: Técnica usada por robots y vehículos autónomos que parte de la construcción de mapas de un entorno y estimación de su trayectoria a partir de su desplazamiento dentro del mismo. La detección de la posición respecto a su entorno es tomada por el robot utilizando sus respectivos sensores según la aplicación destinada.

Mapeo Robótico: Rama de la cartografía con fin de estudio en aplicaciones de robótica para la construcción de mapas o planos, así mismo se denota que según la estrategia de navegación y las capacidades del robot hace variar la creación de mapas por parte de su percepción.

Robot móvil: Máquina de transporte automático capaz de trasladarse de un lugar a otro, partiendo de un punto inicial a un punto final. A través de su sistema programado permite realizar acciones autónomas o por medio de un usuario, usado ampliamente en aplicaciones de riesgo para la humanidad y guía de personal.

ROS: Por sus siglas en inglés (Robot Operating System) es un *framework* para el desarrollo de software para robots.

Sensor: Dispositivo capaz de transformar magnitudes físicas o químicas en magnitudes eléctricas. Las variables de instrumentación (magnitudes físicas y químicas) hacen referencia a distancia, presión, aceleración, temperatura, PH, entre otras. Las magnitudes eléctricas son aquellas que se pueden medir o interpretar y se refieren a dispositivos como resistencias eléctricas (RTD), tensión eléctrica (termopar), una corriente eléctrica (fototransistor) [1].

SLAM: Por sus siglas en inglés (mapeo y localización simultáneos) es una técnica usada por robots y vehículos autónomos para construir un mapa de un entorno desconocido y a su vez estimar su trayectoria al desplazarse dentro del mismo.

Parámetro: Un parámetro es cualquier elemento o dato importante que se utiliza para medir comparativamente algún dato o hecho de la realidad. [2]

RESUMEN

Este proyecto evalúa los parámetros influyentes ante variaciones de iluminación en el funcionamiento de los algoritmos Filtro de Kalman y Filtro de partículas aplicados a SLAM (Mapeo y Localización Simultanea) llevando a cabo experimentos en entornos cerrados que permiten realizar pruebas con distintas intensidades de luz. Para esto se utiliza el robot móvil diferencial Turtlebot3 Burger acompañado del sensor Kinect V1, ya que posee un sensor de profundidad y una cámara RGB, esto permite llevar a cabo construcción de mapas 3D y evidenciar variaciones en los procesos de mapeo dependiendo de la iluminación, lo cual es un requerimiento de este trabajo. El filtro de partículas (PF) es uno de los algoritmos de estimación más adaptados para SLAM, al igual que el filtro de Kalman que interactúa con estados pasados, presentes y estimaciones futuras del sistema, siendo empleado para seguir sistemas estocásticos dinámicos mediante sensores ruidosos. Los algoritmos emplean un método de predicción y corrección, es decir, pronostica un nuevo estado a partir de su estimación previa agregando un factor de corrección de la misma proporción al error de predicción, de tal manera que este error se minimiza con cada iteración, lo cual lo hace adecuado para trabajar ante variaciones de iluminación ya que esta técnica demanda que se pueda reconstruir y localizar en el menor tiempo posible cualquier sistema autónomo bajo condiciones naturales.

1. INTRODUCCIÓN

En la actualidad el problema SLAM es una de las aplicaciones más destacadas en la robótica móvil, esto principalmente debido a los grandes avances que presentan los sensores que proporcionan la información en entornos 3D y 2D [3]. El presente trabajo desarrolla el estudio de dos técnicas de SLAM, “Filtro de Kalman” y “Filtro de partículas”, aplicadas al SLAM, estos algoritmos fueron seleccionados ya que son los utilizados con más frecuencia en la problemática SLAM y así mismo existen gran variedad de configuraciones que se derivan de ellos [4], [5], [6].

Es de primordial importancia seleccionar el sensor o los sensores adecuados para las aplicaciones específicas que se desarrollarán en el proyecto, en este caso principalmente se busca un dispositivo que presente facilidad experimental en variaciones de iluminación, por esto se indaga entre distintos sensores existentes en el mercado, como el sensor laser, el cual emite un pulso láser de corta duración, y calcula la distancia hasta el objetivo basándose en el tiempo en que el láser vuelve al punto donde se generó, aunque presentando una limitación, este sensor solo es para implementaciones 2D [7]; o el sensor sonar, que utiliza la misma técnica que los sensores láser para detectar distancias y presenta ventajas en cuanto a costo y precisión, sin embargo, posee limitaciones en su velocidad, ya que está ligado directamente a la velocidad del sonido, y no presenta mayores interferencias cuando se habla en términos de iluminación, por lo que no sería un sensor acorde a los requerimientos del proyecto [7]. El dispositivo seleccionado es el sensor Kinect V1, inicialmente por las características que posee particularmente un sensor RGB, un sensor de profundidad 3D y micrófonos multi-array. El sensor de profundidad produce imágenes de 640 x 480 pixeles (tamaño VGA), el dispositivo tiene un campo de visión de 58° horizontalmente y 45° verticalmente y el campo óptimo de operación del sensor oscila entre los 0.8 y 3.5 m [8]. A partir de esto se encontraron antecedentes con resultados favorables como en [9], que realizan una comparación entre las dos versiones existentes de este dispositivo, evidenciando que no se presentan grandes diferencias entre ellos, excepto en aplicaciones muy específicas, o en [8], en donde exponen los resultados satisfactorios al implementar un método para convertir datos de profundidad 3D de Kinect en un mapa 2D.

Los entornos sociales presentan de una u otra forma cambios en la intensidad de iluminación, ya sea por las variaciones de iluminación natural a lo largo del día o por el cambio que se presenta en la iluminación artificial de los entornos sociales. Por medio del trabajo conjunto del dispositivo Kinect V1 con el robot móvil diferencial turtlebot3 Burger, se busca la implementación de los algoritmos ya mencionados, “Filtro de Kalman” y “Filtro de partículas”, aplicadas al SLAM. Por

medio de la aplicación de estas técnicas se busca realizar un análisis completo de su funcionamiento en relación con las distintas intensidades de iluminación que se pueden llegar a encontrar en el entorno de implementación y su influencia en el desarrollo de la técnica SLAM, con el fin de detectar los parámetros que, si llegasen a modificarse proporcionarían resultados más satisfactorios en la recolección de imágenes y por consiguiente el diseño del mapa para la localización del robot móvil diferencial.

2. PROBLEMA

El uso de la robótica en distintos tipos de entornos donde anteriormente no estaba involucrada, llevará a la ejecución de procesos donde la interacción de robots con las personas se hará más cotidiana, ya que la mayor parte de las tareas impartidas por el ser humano van a ser suplidas por máquinas capaces de efectuar dichos procesos.

Uno de los retos que debe enfrentar la robótica, si se quiere implementar de forma más presencial en distintos lugares y aspectos en los que las personas viven día a día, es el movimiento autónomo de los robots [10]. Estas habilidades se han ido perfeccionando gracias a la mejora que ha habido tanto en los algoritmos de localización y mapeo como en dispositivos de procesamiento, sensores y actuadores en robótica. La evolución que se presenta hoy en día en cuestión de hardware y software para la robótica convencional, permite que esta se desarrolle cada vez más en entornos sociales; a partir de ello surgen muchos campos de investigación como el uso de los algoritmos SLAM (Simultaneous Localization and Mapping o localización y mapeo simultáneo), el cual presenta diferentes problemáticas como son las variaciones de iluminación en ambientes interiores [11].

Sin duda alguna la iluminación no controlada en espacios interiores compete un desafío en los sistemas de mapeo y localización simultánea [12]. Las condiciones de luz en estos ambientes donde las personas no familiarizadas en robótica interactúan es importante, ya que las variaciones en la iluminación que se encuentran en diferentes horas del día influyen considerablemente en el mapeo del sensor [13], siendo un aspecto importante a trabajar, teniendo en cuenta que en la robótica social no se sugiere implementar restricciones como la luz, ya que esto podría desestimular el uso de robots.

Con este trabajo, se busca determinar las características que debe poseer un algoritmo de mapeo y localización simultánea (SLAM) aplicado en ambientes interiores con variaciones en la iluminación en un robot móvil, basado en resultados obtenidos en pruebas experimentales con algoritmos Filtro de Kalman y Filtro de partículas, mediante el uso de un sensor Kinect.

Teniendo en cuenta el problema hasta acá planteado, se propone la siguiente pregunta de investigación: ¿Qué parámetros son influyentes ante variaciones de iluminación en los algoritmos Filtro de Kalman y Filtro de partículas de localización y mapeo simultáneo (SLAM) en la implementación en un robot móvil diferencial en ambientes interiores?

3. ANTECEDENTES

En [14] se revisa en primer lugar qué tipo de sensores son los indicados a la hora de diseñar un modelo de SLAM, el cual tiene como objetivo analizar el funcionamiento del sensor, reconocer los diversos materiales de desarrollo que se pueden encontrar hoy en día y llegar a la correcta implementación de un algoritmo en SLAM. Las herramientas analizadas son el sensor láser, evidenciando cuáles son sus ventajas y desventajas; finalmente se concluye que el sensor es ideal para extraer propiedades de superficies planas, como las paredes y para detectar marcos de puertas, así mismo se implementa el sensor sonar, realizando el mismo análisis y se destaca que en contraste con otros sensores la velocidad del sonido supone una limitación en comparación con los sensores de visión, confirmando que es más elemental interpretar los datos adquiridos con otros sensores y que no es una alternativa comúnmente utilizada en el SLAM. Se resalta el uso de la odometría, la cual utiliza ecuaciones que dirigen al uso de encoders de las ruedas de un robot móvil, este traduce las revoluciones de las mismas a un desplazamiento lineal relativo al suelo; independientemente de las limitaciones que surjan en el uso de la odometría, en la investigación es una herramienta de suma importancia para el sistema de navegación de un robot.

En el siguiente artículo [15] se presenta una nueva metodología en el desarrollo del SLAM, en la cual se emplea un sensor de visión monocular con visión hacia adelante, y está orientado a disminuir el costo total de implementación explorando un método para ser aplicable en tiempo real. El método requerido consiste en analizar la dirección del robot utilizando en esta un punto de fuga especifico, a partir de entonces los modelos de estimación para la postura del robot y el punto de referencia se derivan como ecuaciones lineales simples; cabe resaltar que mediante estos modelos las posturas de la cámara, y las posiciones históricas se corrigen de manera más eficiente. Se revela que la eficacia del robot se expone en entornos retadores y que además el sensor monocular resulta ser llamativo en cuanto a su costo, peso y bajo consumo de energía; finalmente se puede concluir que para la manipulación de un sistema SLAM se deben tener en cuenta dos factores los cuales son el requisito a nivel computacional del algoritmo debe estar diseñado para ejecutarse en un procesador de bajo costo y realizar un correcto estudio del entorno y relacionarlo a medida con el algoritmo SLAM.

En [16] se incluye al lector en las dificultades de posicionamiento automático que sufren los robots para funcionar en interiores desconocidos, y se indagan soluciones tales como auto-localización por coordenadas las cuales pueden ser facilitadas por el uso del GPS y la implementación de algoritmos de SLAM (Localización y mapeo simultáneo). Así mismo se analiza el tipo de sensores más efectivos para utilizar con esta técnica el uso de cámaras, unidades inerciales y láser, haciendo énfasis en el grupo de sensores de visión como visión monocular y visión estereoscópica y su respectiva configuración. Se indica los ejemplares de

robot de bajo costo existentes en la industria, algunos de ellos utilizados para la diversión, los cuales cuentan con cámaras, hélices y demás. Las dificultades que presenta el SLAM monocular, según el proyecto, consisten en la estimación de la profundidad de los objetos y características, teniendo en cuenta que solo cuenta con una cámara como sensor; para ello se proponen dos algoritmos, que a partir de las pruebas experimentales realizadas se concluye que ambos algoritmos funcionan de manera aceptable.

El proyecto [17] empieza resaltando uno de los grandes problemas que se le presentan a la robótica autónoma, un mundo variante, por esta razón se debe utilizar un hardware que sea capaz de percibir estas variaciones del entorno. Se evidencian 3 factores a tener en cuenta para que un robot pueda ejecutar el aprendizaje de un mapa los cuales son el modelado, la localización y la planificación de la ruta. El siguiente procedimiento es indagar acerca de la instalación del software ROS y los paquetes necesarios para la realización de pruebas; se puede concluir que a partir de ellas se obtuvieron resultados adecuados, ya que el algoritmo funcionó de manera correcta junto con el hardware seleccionado y el robot consiguió realizar un mapa virtual. Se resalta el uso de Upboard, la plataforma que permite llevar a cabo simulaciones y reconocer objetos específicos para aplicaciones y una Raspberry pi para la construcción. Finalmente se tiene en cuenta ciertos agregados los cuales le darían más exactitud al robot, como añadir más sensores para la construcción 3D del entorno, implementar nuevas cámaras de desarrollo para aumentar la eficacia de la Upboard y emplear las versiones mejoradas de ROS para trabajar con sus sistemas operativos más actualizados.

En este artículo [18] se evidencia el incremento en el uso de sensores visuales en implementaciones de SLAM, todo esto gracias a su bajo costo y ahorro de energía y resaltando su superioridad en contraste de sensores de rango, que proveen información angular. La cámara al ser un sensor que calcula el ángulo con respecto a los elementos de la imagen, indica que la profundidad no puede ser obtenida mediante una sola medición; por lo siguiente se introduce a un nuevo manejo de SLAM, el cual está basado en sensores angulares y la técnica a utilizar es la triangulación estocástica que define una hipótesis en la medida de profundidad inicial. Mediante los experimentos elaborados en el entorno interior designado, se puede afirmar que el uso de una parametrización inversa para la profundidad de las imágenes permite el manejo adecuado y directo de un esquema estándar EKF en el proceso de estimación, por medio del cual se puede linealizar un sistema no lineal respecto a la estimación actual y de esta manera simplificar la implementación final. Cabe resaltar las ventajas de los métodos SLAM sin aplazamiento, los cuales en la implementación del proyecto proporcionan información de orientación y genera una mayor eficacia.

En el siguiente trabajo [19] los autores comienzan introduciendo al lector en las dificultades que presenta el SLAM al desempeñarse en ambientes interiores los cuales cuentan con variables que intervienen en el momento de recolectar datos y explicando la técnica utilizada, la cual consiste en una implementación de SLAM utilizando un filtro de Kalman ampliado, el robot utilizado fue un Seekur Jr. y la tarea de mapeo fue llevada a cabo por un sensor láser. Se proponen específicamente las dificultades que pueden presentarse en el sistema SLAM EKF, principalmente, la complejidad computacional aumenta, junto con los puntos de referencia del diseño. Después de ejecutar dicho proyecto en el entorno seleccionado, con las características mencionadas, los autores afirman que el rendimiento del sistema es adecuado, el cual cuenta con un algoritmo rápido en ejecución, y precisión en el mapeo, a través de los cuales se obtuvieron resultados esperados, muy similar a los que se presentan en simulaciones; finalmente los autores señalan que a pesar de lo anterior, el sistema puede mejorar combinando el funcionamiento diseñado previamente por ellos con técnicas de SLAM jerárquica, permitiendo extender su uso.

En [20] realizan el proceso de SLAM utilizando un filtro de partículas Raowellwellized el cual ayuda a mejorar la eficacia computacional, algunos de los casos más relevantes surgen en la aparición de una subestructura gaussiana lineal, la cual logra ser manejada de manera eficiente por los filtros de Kalman, esta es la formulación estándar del filtro de partículas Rao-Blackwellize acompañado de un segmento de línea como hito. Con el fin de reducir el costo computacional señalan los autores que se utilizaron solo dos puntos finales de un segmento de línea. En experimentos con datos distinguidos, el método propuesto permite tener una función de SLAM adecuada y un mapa compacto incluso en un entorno alterno, lo cual se denomina como un ambiente desordenado; finalmente se concluye que todas las modificaciones menores al RBPF-SLAM general propuestas en este documento, como la asociación de línea y el esquema de cálculo de peso, se integraron eficientemente en el marco RBPF-SLAM del proyecto; se sugiere un progreso a largo plazo para el proyecto, este se basa en generación de mapas locales y las relaciones topológicas fundamentadas en la descomposición del mapa de segmento de línea global.

En el artículo [21] se introduce a una implementación diferente del SLAM, mapeando el interior con brújula magnética digital, una de las técnicas implementadas por los investigadores está basada en aplicar un método de sonar visual y luego relaciona estos datos a través de un algoritmo basado en *Iterative Closed Point* y produce el mapa, la otra técnica consiste en utilizar una extensión de SLAM a un algoritmo de localización de cámara la cual se encuentra basada en el filtrado de partículas que proporciona resistencia al movimiento errático. El entorno seleccionado en el cual el robot llevará a cabo sus funciones es un entorno interior ideal, aun así, los autores resaltan que para un entorno no controlado requiere modificar los dispositivos de mapeo utilizados, mediante la utilización de escáneres láser o sistemas de visión. Un atributo particular en el

proyecto fue la innovación al incluir una función de detección de secuestro, que consistió en tomar el robot mientras realizaba el mapeo de un entorno y ubicarlo en posiciones distintas, los autores relatan que, aunque hacen falta modificaciones, el algoritmo funciona de manera eficiente.

El trabajo [22] se refiere a un novedoso método de seguimiento inercial-visual modificado, el cual se ejecuta constantemente para contrarrestar la complejidad computacional. Este documento indaga un dilema diferente, en lugar de mapear, mediante el seguimiento inercial visual con la información 3D disponible como base, se agregan mediciones de flujo óptico y la cinemática para permitir una estimación precisa de la pose de la cámara. En conclusión, esta investigación incrementa el sistema de seguimiento visual-inercial, con cálculos de flujo óptico, revelando que la adición de mediciones de flujo óptico reduce la cantidad requerida y la frecuencia de las características de observación con profundidad conocida.

El objetivo principal de este artículo [23] consiste en ubicar y realizar mapa en un interior. La asignación se realiza durante el movimiento de un sensor 3D en el interior de un edificio con el método de localización y el mapeo simultáneo (SLAM). El desarrollo de este experimento fue posible mediante la implementación de un Microsoft Kinect para Windows v1, la localización fue realizado por parte del algoritmo RGBDSLAM (ejecutado sobre ROS) la cual permite obtener rápidamente modelos tridimensionales en color de objetos y escenas interiores, generalmente con una cámara de estilo Kinect portátil; para obtener el procesamiento en línea, la imagen actual solo se compara con un subconjunto de las imágenes anteriores, el siguiente paso es construir un gráfico cuyos nodos corresponden a las vistas de la cámara y cuyos bordes corresponden a las transformaciones 3D.

El trabajo [24] tiene como objetivo suministrar algoritmos logrando mejorar la detección de objetos en movimiento, así como la eliminación de características de imagen en sistemas (SLAM) visuales con sensores de visión de mano, enfocadas en los problemas de estimación del estado del robot y MOD en entornos dinámicos. En esta investigación se desarrolla un algoritmo SLAM en línea con un detector de objetos en movimiento fundamentados en la restricción de correspondencia para la matriz esencial de características de la imagen, siendo calculada a partir de las características de la imagen seleccionadas mediante el método SURF.

En [25] se describirá una solución adoptada (modelo difuso (TG) Takagi-Sugeno) para el problema de localización y mapeo simultáneo (SLAM) con el método de asociación de datos de dos sensores (TSDA); para ello se introduce dicho método en la navegación de un robot móvil cuyo funcionamiento está basado en una programación lineal de punto interior, revelando que el método ETSDA, presenta

baja complejidad computacional y es de mayor precisión que el JPDA. Como conclusión este artículo demuestra que el método ETSDA supera el método JPDA, proporcionando una mayor asociación de datos y la capacidad de comenzar nuevas cualidades que hacen parte del método ETSDA.

En el artículo [26] se estudia un método de localización 3D para un IGS (Information Gathering System) para la generación de nubes de puntos interiores de edificios. El método en el cual se basaron fue en la introducción de una mochila IGS, la cual es utilizada por un operador humano y recopila información sobre el medio ambiente a lo largo de los pasillos. A continuación, se propone un método de estimación de postura 3D para la localización del IGS basado en el método SLAM compuesto por dos tipos de métodos SLAM, que extraen las poses más confiables de los sensores en IGS. Para la localización del eje z perpendicular al piso, la distancia de recorrido del IGS se definió como una nueva variable de dimensión, el espacio con la distancia de recorrido y la información de distancia a lo largo del eje z se incorporaron al sistema coordenadas; concluyendo que la estimación de posición muy precisa es realizable en el ambiente interior sin odometría basada en decodificadores.

En el documento [27] se define el inconveniente SLAM en un entorno desconocido en 2D basado en esquinas naturales, estas se escogen como puntos de referencia al encontrar los puntos finales de los segmentos de línea extraídos de los datos del sensor. El siguiente paso consistió en llevar a cabo el diseño de SLAM basado en el filtro de Kalman ampliado mejorado (IEKF), este se analiza para demostrar el menor costo de cálculo que el algoritmo EKF-SLAM estándar. En la parte experimental se lleva a cabo el experimento de extracción de líneas, extracción de esquinas y localización y mapeo de robots, el robot utilizado en el proyecto es el robot móvil P3-DX y el sensor elegido para el mapeo es el láser HOKUYO. Los sensores láser y sensores de visión estiman la posición del robot basados en señales hechas por el hombre, sin embargo, los autores afirman que los puntos de referencia artificiales se pueden aplicar en un rango muy estrecho con respecto a las características naturales del entorno real porque los hechos por el hombre no pueden colocarse en el área deseada donde la gente no puede llegar.

El artículo [28] expone una aplicación SLAM en línea para auto-localización de robots móviles en entorno artificiales basado en puntos de referencia. Con ello se busca explorar las características del filtro de Kalman extendido (EKF) con el fin de reducir el error acumulativo de la estimación de posición del robot móvil. Se expresa que para la aplicación de métodos como el expuesto en el trabajo la precisión del mapa es de gran importancia, sin embargo, se requiere mayor precisión en la estimación de posición del robot. Para llevar a cabo el experimento se hace uso de un robot real. Los autores aclaran que el método utilizado suprime el error acumulativo de auto-localización al usar los puntos de referencia. Las principales contribuciones del documento son las descripciones de un método

para extraer planos de la nube de puntos como normales y de la configuración de la condición para evitar falsas coincidencias.

En [29] se expone un trabajo implementado con base a un robot desarrollado internamente, el cuál realiza procesos de mapeo e implementación de detección de objetos. El desarrollo se lleva a cabo sobre la plataforma ROS (Robot Operating System) y se busca la obtención de un mapa 2D etiquetado de un entorno interior estático desconocido utilizando la técnica de localización y mapeo simultáneo (SLAM), para la creación de un mapa y así mismo la detección de objetos dentro del mismo. Esta solución se implementa por medio del paquete GMAPPING de código abierto. La herramienta de visualización es RViz, siendo esta la misma utilizada en el documento actual. Se resalta el uso de hardware de bajo costo que mantiene la precisión del mapa obtenido. Se concluyen resultados precisos y verificables de detección y etiquetado de objetos.

El articulo [30] abarca un proyecto basado en UKF SLAM monocular para la localización de un robot móvil, el cual presenta un esquema de SLAM visual en un robot móvil con ruedas aprovechando la información de odometría para ajustar la escala a unidades métricas. A través de una simulación del esquema propuesto se muestra que el filtro UKF resulta más consistente en la estimación de posición comparada con el tradicionalmente usado EKF asumiendo conocidos los parámetros por defecto del sistema robot/cámara. Lo que permite concluir finalmente que el rendimiento se ve reflejado tanto en la consistencia de la estimación de la localización del robot, como en el error de estimación de la profundidad de los landmarks del mapa.

4. JUSTIFICACIÓN

La habilidad que tienen los seres humanos de crear herramientas con los recursos que se encuentran en su entorno evoluciona rápidamente con el pasar del tiempo, esto conlleva a que las tareas que la maquinaria moderna es capaz de efectuar sean cada vez más complejas y se realicen con más precisión. Esto, en el campo de la robótica, permite un incremento en la implementación social de robots, la cual se define en prestar un servicio de atención para los seres humanos, ya sea desempeñando tareas en un entorno doméstico, o llevando a cabo funciones en entornos empresariales tales como guía de personas en las instalaciones y apoyo al personal de trabajo.

Teniendo en cuenta que la implementación de la robótica autónoma en entornos sociales va de la mano con el hecho de que el robot lleva a cabo su funcionamiento en ambientes interiores, conlleva a la necesidad de identificar y analizar los factores clave a estudiar para obtener un buen funcionamiento de la máquina a la hora de desempeñar sus tareas, a partir de resultados obtenidos de pruebas ejecutadas con distintos algoritmos de mapeo y localización simultanea (SLAM). Así mismo es necesario que el robot tenga conocimiento de su entorno y sea capaz de desplazarse y localizarse a lo largo de este, de esa manera el robot tendrá completa autonomía de sus movimientos cuando salga con el objetivo de identificar un lugar específico.

Uno de los parámetros de variación que se presenta en los ambientes interiores e influye en el funcionamiento de sensores visuales es la iluminación. Si el robot llegase a desempeñar sus funciones en un entorno interior, se debe considerar que la luz que percibe el sensor de mapeo no será igual en todo momento, resaltando la importancia de una correcta selección de los sensores que formarán parte del sistema [31].

Actualmente existe una amplia gama de sensores en el mercado tales como, sensores ultrasónicos que calculan la distancia a partir del uso de un único oscilador que cumple las funciones de emisión y recepción de las ondas ultrasónicas, sin embargo, estos se enfocan en ambientes bajo el agua [32]. Así mismo se encuentran sensores ópticos que a diferencia de los sensores de ultrasonido requieren de un transmisor y un receptor independiente para el cálculo de distancia a partir de la reflectividad con el objeto. De igual forma, se hallan los sensores láser, que se pueden utilizar para detectar puntos de referencia invariantes en la dirección de los rayos láser, como las esquinas de paredes, y campos que se encuentran en el entorno donde será su movimiento, a pesar de ello poseen ciertas limitantes como el rango de alcance [33] . En el campo de los sensores visuales, el rango de aplicaciones de un sensor como el Kinect es mayor a comparación de los sensores láser y los ultrasónicos ya que permite un mapeo de objetos en 3D, aunque presenta alteraciones ante cambios de iluminación [31], adaptándose al problema planteado en este documento.

Con este proyecto se pretende encontrar que variaciones se presentan en los resultados obtenidos a través de las aplicaciones SLAM (Localización y mapeo simultáneos) 2D y 3D de los algoritmos filtro de Kalman y filtro de partículas ante variaciones de iluminación en un robot móvil diferencial mediante un sensor Kinect. El trabajo que se va a realizar en SLAM es en ambientes interiores con variaciones de luz, así mismo, se sabe que una vez el mapa sea construido el robot lo utilizará para desplazarse en el entorno, previendo que tendrá que interactuar con objetos a su alrededor tales como personas circundantes.

5. OBJETIVOS

5.1 Objetivo general

Determinar los parámetros influyentes en la técnica SLAM ante variaciones de iluminación en los algoritmos Filtro de Kalman y Filtro de partículas, en un robot móvil diferencial en ambientes interiores a partir de un sensor Kinect.

5.2 Objetivos específicos

- Identificar los parámetros relacionados a la técnica SLAM ante variaciones de iluminación, al aplicar los algoritmos Filtro de Kalman y Filtro de partículas en un robot móvil diferencial en ambientes interiores.
- Implementar dos algoritmos para SLAM basados uno en Filtro de Kalman y el otro en Filtro de partículas en un robot móvil diferencial en ambientes interiores con variaciones de iluminación, mediante el uso de un sensor Kinect.
- Seleccionar la técnica SLAM más adecuada teniendo en cuenta la variación de la luz del entorno, a partir de pruebas experimentales con los algoritmos Filtro de Kalman y Filtro de partículas.

6. MARCO TEÓRICO

La implementación de sistemas en el campo de la navegación de robots móviles cada vez se ha tornado más frecuente en la robótica social, para ello se hace necesario emplear técnicas SLAM.

En [34] el Mapeo y Localización Simultánea SLAM aborda el problema de percepción de información de un robot que navega en un entorno desconocido, donde realiza un mapa de su trayectoria, y al mismo tiempo logra localizarse dentro de su mapa. Aunque se han desarrollado varias técnicas de mapeo, partiendo que cada una de ellas tiene sus méritos y deficiencias. A partir de este artículo se analizan distintas técnicas como Gmapping, Cartographer, Hector, Karto, Frontier Exploration, Kalman y Partículas con el fin de exponer la diferencia en cada una de ellas, resaltando el Filtro de Kalman y el Filtro de Partículas.

SLAM, del inglés *simultaneous localization and mapping* y en español como mapeo y localización simultáneos, es un proceso por el cual un robot móvil puede construir un mapa al recorrer un entorno y al mismo tiempo usar este mapa para estimar su ubicación.

En [35] a partir del SLAM, la trayectoria de la plataforma y la ubicación de todos los puntos de referencia son estimados sin la necesidad de ningún conocimiento *a priori* de la ubicación. A partir de ello el robot captura la información necesaria de su trayectoria con la ayuda de uno o más sensores según el campo de investigación que se realice.

6.1 Hardware

Esencialmente, se optó por dos principales componentes para poder analizar y comprender el SLAM, el robot y sus sensores que cumplirán la necesidad de ser usados para la medición de distancias y captura de datos. Estos componentes se describen en las secciones 6.2.1, 6.2.2 y 6.2.3.

6.1.1 Robot móvil diferencial TurtleBot3 Burger

Basados en el artículo [36] el robot móvil considerado en este documento posee tracción de dos ruedas, con una tercera rueda para proporcionar su estabilidad. El bosquejo del robot se visualiza en la (Figura 1), cada una de las ruedas se conduce independiente, por lo tanto, este mecanismo es capaz de maniobrar el robot en cualquier sentido de la trayectoria en un determinado entorno.

El robot móvil diferencial aplicado es capaz de desplazarse hacia adelante y hacia atrás, no obstante, cumple la función de dirigir su ángulo de rumbo diferencial, la velocidad lineal de la rueda izquierda (VI) y la velocidad de la rueda derecha (VD).

Al cumplir con esta serie de funciones, se hace el uso de este dispositivo por su capacidad de hacer un radio de giro lo más pequeño posible, incluso ser capaz de girar en su mismo lugar como se muestra en la (Figura 1), cumpliendo con las tareas de realizar su navegación en ambientes anteriores sin posibilidad de chocar con algún objeto.

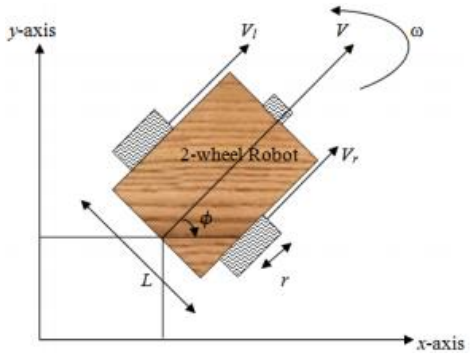

Figura 1. Robot de 2 ruedas [36].

En la gama de los robots móviles, el *TurtleBot3 Burger* siendo un robot móvil pequeño, asequible y programable, juega un papel muy importante debido a su robustez y personalización. Sus ventajas de exploración en *Robot Operating System* (ROS) generan incentivos de trabajo en sus diferentes ámbitos como educación, investigación y desarrollo.

El principal objetivo que maneja TurtleBot3 es apuntarle a reducir su precio y tamaño para su fácil asequibilidad sin dejar atrás su robustez, funcionalidad y capacidad de desarrollo. Cuenta con un buen y cómodo chasis para diferentes tipos de adaptaciones de sensores, cubriendo desarrollar aplicaciones de gran complejidad y largo alcance a pequeña escala y menor costo.

6.1.1.1 Componentes de fábrica y dimensiones del TurtleBot3 Burger

El Turtlebot3 Burger está conformado por una serie de dispositivos los cuales son capaces de adaptarse a las distintas investigaciones y desarrollos que se puedan aplicar. Los principales componentes con los que cuenta son los siguientes:

- **SBC (Single Board Computer)**

  Raspberry Pi 3 Model B

- **Sensor**

  Laser Distance Sensor (360° LiDAR for SLAM & Navigation)

- **Control board**

  OpenCR (ARM Cortex-M7)

- **Actuador**

  Dynamixel X series (X2 for Wheels)

- **Battery**

  Li-Po Battery 11.1V – 1,800mAh

Una vez se encuentren implementados los componentes anteriormente mencionados, el Turtlebor3 Burger queda sometido bajo las siguientes dimensiones:

En la (Figura 2) se denota las dimensiones brindadas desde una vista superior y el peso equivalente.

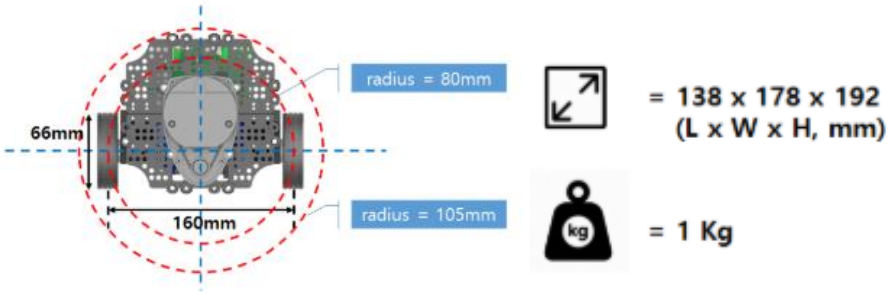

Figura 2. Vista superior & especificaciones TurtleBot3 Burger [37].

En la (Figura 3) se denota las dimensiones brindadas desde una vista frontal y lateral.

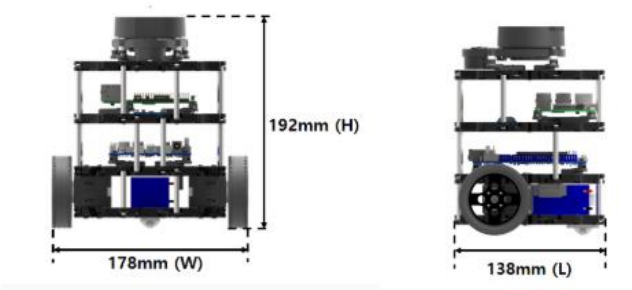

Figura 3. Medidas TurtleBot3 Burger [37].

6.1.2 Sensor laser LIDAR 360°

Iniciando en el área de los dispositivos de medición de distancia, y siendo un sensor incorporado por TurtleBot3 Burger, el sensor láser *LIDAR 360º*, es un tipo de escáner láser que se encuentra en continuas aplicaciones de la robótica. Las técnicas que dominan estos dispositivos es la obtención de distancia basadas en láser, y se les denomina técnicas de tiempo de vuelo (TOF por sus siglas en ingles *Time-of-Flight*).

Según el artículo [14] Las ventajas más notables del sensor láser es su rapidez ya que maneja notablemente su desempeño en tareas siendo instantáneo su medida, La precisión es bastante notable ya que las nuevas generaciones actúan con precisión de margen de error inferior a 10 milímetros y la resolución angular es manejada entre 0.5° o de 0.25° según su utilidad. Por lo tanto, los datos obtenidos de un escáner láser pueden ser interpretados directamente como la distancia de los objetos en una posición determinada. Por otra parte, presta un excelente manejo por las propiedades de extracción de superficies planas, como paredes y puertas, gracias a la densidad de distancias que ofrece. A partir del artículo anterior es un sensor adecuado para poner a prueba inicialmente el SLAM en el TurtleBot3 Burger, en la (Figura 4) se muestra su implementación dentro del robot.

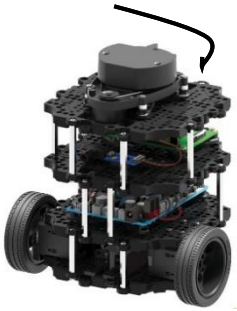

Figura 4. Sensor 360° LIDAR en el TurtleBot3 Burger [37].

6.1.3 Sensores de visión (Sensor Microsoft Kinect v1)

Los sensores de visión se emplean en aplicaciones en los que se requieren parámetros detallados del entorno de trabajo. En [14] se realiza énfasis en parámetros de las imágenes a color o escala de grises, donde permiten utilizar un amplio repertorio de datos para localizar e identificar objetos que se encuentren en su entorno de navegación. Destacándose en la recepción de gran cantidad de información del entorno y la extracción de datos en 3D.

Las principales ventajas de los sensores de visión son:

- Gran cantidad de información del entorno.

- Extracción de información en 3D.

Principales inconvenientes:

- Captura de datos altamente influenciada por la cantidad de luz del entorno donde se realice la navegación.
- Dispositivos costosos.
- Alto complejidad computacional para recepción y tratamiento de datos.

Al realizar una introducción de las principales ventajas e inconvenientes de los sensores visuales, los algoritmos seleccionados (Filtro de Kalman y Filtro de Partículas) están ampliamente ligados con la elección del hardware, ya que la precisión en el entorno de navegación y la tolerancia de ruido en entornos sociales son punto de referencia a tomar en cuenta. El Kinect visualizado en la (Figura 5) siendo un sensor visual juega un papel crucial en este tipo de aplicaciones, además de cumplir la función del desarrollo del modelo con variación de iluminación que usaremos para buscar nuestra implementación en las técnicas SLAM.

Figura 5. Sensor Kinect v1 de Microsoft [14].

6.1.3.1 Características del sensor Kinect

El sensor Microsoft Kinect siendo el dispositivo utilizado para esta implementación se define como un controlador de juego libre y entrenamiento desarrollado por Microsoft para poner a prueba video juegos en la consola XBOX 360, y desde junio del 2011 para pc desde la versión Windows 7. Permite conectar a los usuarios de una forma más interactiva sin necesidad de tener contacto físico con la consola de video juegos, todo este proceso se lleva a cabo por reconocimiento de gestos, comandos de voz, objetos e imágenes.

Componentes del sensor Kinect:

- Cámara RGB

El sensor Kinect cuenta con una pequeña cámara RGB situada en la parte frontal, con la cual realiza la captura de imágenes a color. Estas cámaras funcionan utilizando unos sensores capaces de transformar la señal recibida en fotones a

una señal electrónica digital que descompone la luz capturada en tres componentes (RGB Por sus siglas en ingles *Red, Green, Blue*)

- Sensor de profundidad

El sensor de profundidad está principalmente compuesto por dos componentes: un generador de infrarrojos y un sensor CMOS monocromo de luz infrarroja. Al tener esta composición se logra capturar video con información en tres dimensiones bajo cualquier condición de iluminación.

A diferencia de la cámara RGB, el sensor implementado con tecnología (CMOS) permite un tiempo de exposición menor, y por lo tanto, una mayor sensibilidad a la luz. El sensor de distancia produce imágenes con una resolución de 320x240 (QVGA) con una información de profundidad de 11 bits.

6.2 ROS

Por sus siglas en inglés *Robot Operating System* (ROS) comprende una biblioteca de software y herramientas que apoyan el desarrollo de sistemas de robots. ROS se origina en un proyecto desarrollado en la Universidad de Stanford con fines de investigación. Siendo una plataforma de código abierto se ha convertido en un plus para marcos de desarrollo de software de robots [38].

6.2.1 Paquete de ROS

Los paquetes en ROS son los elementos principales de su arquitectura, debido que proporcionan una capa de comunicación para facilitar la transmisión entre los nodos, siendo estos los paquetes que pueden contener los procesos de ROS en tiempo de ejecución, conjunto de datos y archivos de configuración [39].

El principal objeto de los paquetes de ROS es ofrecer una funcionalidad de manera que el software se pueda construir y reutilizar fácilmente. Sin embargo, es posible crear propios paquetes para implementar un desarrollo o contar con las dependencias que pueden tener los paquetes entre sí.

Pasos para la creación de un paquete en ros:

- Se debe tener configurada correctamente la variable de entorno ROS_PACKAGE donde se debe contener el directorio de trabajo.
- Se debe tener configurado un directorio de trabajo donde se crean los paquetes, tanto para crear el directorio de trabajo como para configurar las variables de entorno de ROS.

Una vez configurado el entorno de trabajo se procede a la creación de los paquetes necesarios para la ejecución, existen dos métodos:

- Comando ROSCREATE-PKG: Herramienta clásica en todas las versiones de ROS.
- Comando CATKIN_CREATE_PKG: Nueva herramienta para la creación de paquetes disponibles a partir de la versión *Groovy* de ROS.

6.2.2 Utilidades de línea de comando

En esta etapa del proceso se denotarán algunas de las utilidades de línea de comando, para ejecutar o procesar los paquetes de ROS utilizados en este desarrollo:

- roscd: Herramienta que logra cambiar el directorio comenzando con el nombre del paquete, la pila o la ubicación del mismo.
- rosls: Herramienta para visualizar la lista de archivos.
- rosed: Herramienta para editar un archivo en un paquete.
- roscp: Herramienta utilizada para la realizar la copia de un archivo en un determinado paquete.
- rosrun: herramienta utilizada para ejecutar un solo nodo ROS.
- roslaunch: herramienta utilizada para ejecutar múltiples nodos ROS de forma local y remota. Cuyo código está escrito en XML y brinda de forma asequible automatizar complejos procesos y configuraciones de arranque para un paquete a partir de un solo comando.
- Catkin: Herramienta de compilación de ROS actual, basada en Cmake la cual consiste en una herramienta multiplataforma de generación o automatización de código (Software libre y de código abierto).

6.3 SLAM (Mapeo y Localización Simultáneos)

En este capítulo se presentarán las bases matemáticas del problema de Localización y modelado simultáneo de robots móviles, así mismo se desarrollarán las expresiones que modelan el problema del SLAM y algunos algoritmos que se encuentran diseñados para abarcar este problema.

SLAM por sus siglas en inglés (*Simultaneous localization and mapping)* consiste en una técnica que le permite al robot móvil navegar a través de un entorno

desconocido, crear un mapa compacto del mismo y determinar su posición dentro de este mapa [13].

Las posibles soluciones del SLAM planteadas en cada una de las aplicaciones como lo son; robótica móvil, navegación de robots en entornos interiores, exteriores, robots subacuáticos y robots voladores hacen que el problema sea uno de los sucesos más notables para la comunidad en los últimos años [14]. A un nivel teórico y conceptual, SLAM puede considerarse un problema resuelto. Sin embargo, aún quedan problemas en algunas posibles soluciones con distintas pruebas en las aplicaciones más generales del SLAM [40].

A partir de una serie de pasos el SLAM empieza a desarrollarse en aplicaciones de navegación en entornos desconocidos. Inicialmente se realiza la extracción de características del entorno donde se encuentre realizando su navegación, se procede a la asociación de datos, estimación del estado y actualización constante de las características, lo que conlleva a actualizar la posición del robot dentro de su entorno.

6.3.1 Slam_Gmapping

El paquete Gmapping perteneciente a los autores Giorgio Grisetti, Cyrill Stachniss, y Wólfram Burgard, y además siendo una obra de Brian Gerkey puede ser encontrado en los repositorios de ROS en *Github* con sus correspondientes archivos, presentándose como una técnica al problema del SLAM.

Basados en [41] este método utiliza un Filtro de Partículas, en el cual cada una de ellas se lleva un mapa individual del entorno y se estima un gran número de partículas que habrá que reducir. Para resolver este problema, Slam_Gmapping presenta una solución basada en una técnica de adaptación consistiendo en la disminución de cantidad de muestras mediante un cálculo de la distribución exacta, con parámetros iniciales tales como movimiento del robot y la observación más reciente.

Mediante esta disminución de incertidumbre sobre la pose del robot se lleva a cabo el paso de predicción del filtro, donde se aplica un enfoque selectivo para llevar a cabo la operación de re-muestreo que reduce notablemente el problema de la reducción de partículas. Para implementar el algoritmo Gmapping es necesario contar con un robot móvil que proporcione datos de odometría y que cuente con un sensor para la captura de los datos.

6.3.2 Filtro de Partículas

Definido en [42] el Filtro de Partículas es una implementación alternativa no paramétrica del Filtro de Bayes, su cálculo parte por un número finito de parámetros difiriendo en la forma en que se generan y en la forma de poblar el espacio de estado. La principal idea del filtro de partículas es generar aleatoriamente un conjunto de partículas, donde cada partícula es una posición hipotética del robot.

En [43] el Filtro de Partículas también conocido como *Sequential Monte Carlo Method* (MCL) involucra dos principales fases para concluir su localización. En primera instancia se imparte la predicción del estado de las partículas y en la segunda fase la actualización de la población de las partículas mediante la sustitución de partículas de bajo peso.

6.3.2.1 Filtro de Partículas aplicado al SLAM

El filtro de Partículas es un algoritmo muy recursivo y se basa principalmente en dos fases: la predicción y la actualización. La primera fase se basa en el movimiento del robot, generando las N partículas y la segunda fase procede a realizar la actualización de ponderación de las partículas generadas utilizando el modelo de observación.

Para resolver el problema del SLAM, cada partícula representa una trayectoria posible del robot y un mapa, y no únicamente posiciones, a diferencia de los *EKF's* [41]. La principal idea es estimar una distribución posterior $p(s_{1:t} \mid z_{1:t}, u_{0:t})$ sobre las trayectorias potenciales $s_{1:t}$, sus mediciones de odometría $u_0 : t$ y utilizar esta distribución posterior para recalcular su nuevo mapa y trayectorias:

$$p(s_{1:m} \mid z_{1:t},\ u_{0:t}) = p(m \mid s_{1:t},\ z_{1:t})\ \ p(s_{1:t} \mid z_{1:t}, u_{0:t}) \qquad (1)$$

Este resultado puede hacerse de manera adecuada, debido a que la distribución superior sobre mapas $p(m \mid s_{1:t},\ z_{1:t})$ puede calcularse analíticamente, dados por conocidos $s_{1:t}\ y\ z_{1:t}$ y puede hacerse ya que los mapas son condicionalmente independientes [41].

Basado en el modelo matemático del filtro de partículas aplicado al SLAM se estima la distribución posterior $p(s_{1:t} \mid z_{1:t}, u_{0:t})$ sobre trayectorias potenciales, en donde un mapa individual se asocia a cada muestra. Cada mapa es construido mediante las observaciones $z_{1:t}$ y las trayectorias $s_{1:t}$ representadas por la partícula que corresponde.

6.3.2.2 Modelo Matemático

Basados en el artículo [43] se describe el modelo matemático del algoritmo filtro de partículas principalmente en 4 pasos:

$$X_k \quad (2)$$

$$\frac{1}{N} \quad (3)$$

$$x_{k+1} \approx p(x_{k+1}|x_k) \quad (4)$$

$$w_{k+1} = p(y_{k+1}|x_{k+1}, U) * w_k; \quad (5)$$

$$w_{k+1} = \frac{w_{k+1}}{\sum_{n=0}^{N} w_{k+1}^{[n]}} \quad (6)$$

En el primer paso se hace referencia a la ecuación donde se crea una distribución aleatoria de partículas X_k.

En el segundo paso se realiza la inicialización de los pesos de las partículas donde N es el número total de partículas $\frac{1}{N}$.

En el tercer paso se procede al paso de predicción donde el modelo cinemático del robot es aplicado a cada partícula y el peso estimado es el producto de medidas de probabilidad presentado como el estado después de la aplicación de un comando de control U para la partícula n. La normalización es aplicada al poner el peso de la partícula en el intervalo [0,1], de la forma $x_{k+1} \approx p(x_{k+1}|x_k)$

El cuarto paso parte de la estimación de la evolución de los pesos de las partículas $w_{k+1} = p(y_{k+1}|x_{k+1}, U) * w_k;$

El quinto y último paso realiza la normalización de los pesos $w_{k+1} = \frac{w_{k+1}}{\sum_{n=0}^{N} w_{k+1}^{[n]}}$

A partir de la normalización de los pesos, se permite volver a tomar muestras de un conjunto de partículas elegidas de acuerdo con sus valores de pesos y el proceso se repite recursivamente, tomado como el paso de actualización X_k.

6.3.3 Filtro de Kalman

El filtro de Kalman en definición central es uno de los casos típicos de redes Bayesianas y es un algoritmo capaz de identificar el estado oculto (no medido) de un sistema dinámico lineal. Al contar con los datos de mediciones de entrada, estado calculado previamente y matriz de incertidumbre es capaz de procesar su información en tiempo real y no requiere de alguna otra información adicional.

En el área de la navegación de robots móviles describe una solución recursiva para el problema del filtrado lineal de datos discretos. En [44] se denota que su derivación fue dentro de un gran contexto de modelos espacio-espacio, donde el núcleo se torna como la estimación por medio de mínimos cuadrados recursivos. A partir de ello, el filtro de Kalman ha sido objeto de una extensiva investigación y aplicación particularmente en el área de la navegación autónoma. La solución que brinda se hace factible por cuanto el filtro combina toda la información observada y el conocimiento previo del comportamiento del sistema para producir una estimación de tal manera que estadísticamente el error es minimizado.

Basados en [41], analizando el algoritmo a “alto nivel”, los filtros de Kalman estiman un proceso cuya forma de control consiste en la retroalimentación: el filtro estima el estado del proceso en un instante dado y posterior obtiene retroalimentación en forma de mediciones ruidosas. Las ecuaciones para el filtro de Kalman se clasifican en dos grupos: a) ecuaciones de actualización de estado y b) ecuaciones de actualización de medición.

En el caso del grupo (a) el proceso consiste en proyectar al futuro la estimación del estado actual y la estimación de la covarianza del error para obtener el estimado a priori del siguiente instante.

Por otra parte, en el caso del grupo (b) el proceso consiste en la retroalimentación, al incorporar a la estimación *a priori* una nueva medición para adquirir un estado posteriormente mejorado.

Partiendo de una definición conjunta las ecuaciones de actualización de estado pueden ser visualizadas como predictorias, mientras que las actualizaciones de medición pueden ser vistas como correctoras.

6.3.3.1 Filtro de Kalman aplicado al SLAM

El enfoque más utilizado para resolver el problema de mapeo y localización simultánea es utilizar los filtros de Kalman [44]. La principal ventaja del enfoque del filtro de Kalman es el hecho de que estima la distribución posterior completa de los mapas en línea y además de ello conserva toda la incertidumbre en el mapa, lo cual resulta beneficioso para la navegación, adicionalmente el enfoque puede

demostrar que converge con probabilidad de uno al verdadero mapa y posición del robot sobre una distribución de incertidumbre residual que se origina en gran medida de una fluctuación original aleatoria [41].

Una desventaja del filtro de Kalman es que solo puede estimar el estado de un proceso controlado en tiempos discretos, dirigidos por una ecuación lineal estocástica. Sin embargo, surge el EKF (*Extended Kalman Filter*) por sus siglas en inglés, el cual linealiza sobre la media actual y la covarianza.

Ahora bien, la solución al mapeado mediante la utilización de EKF sufre dos principales limitaciones [41]:

- Los EKF no logran incorporar información negativa, es decir, las mediciones negativas producen distribuciones posteriores no Gaussianas que no pueden representarse en los EKF.
- Los EKF no brindan soluciones al problema de asociación de datos, por tanto, se conduce a notables errores en el mapeado.

6.3.3.2 Modelo Matemático

Guiados por el artículo [44] el inicio del proceso parte por generar un pronóstico en el tiempo del estado hacia adelante, tomando en cuenta todos los datos disponibles en ese instante, y el segundo proceso genera un pronóstico mejorado del estado como se ilustra en la (Figura 6), minimizando el error estadísticamente.

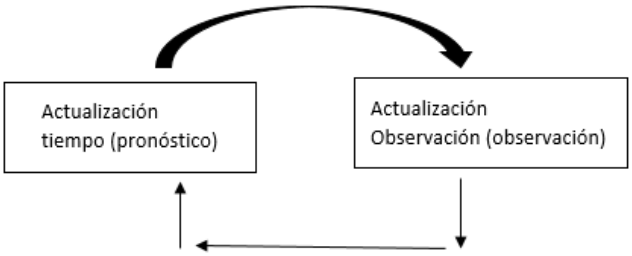

Figura 6. Ciclo inicial del filtro de Kalman [44].

$$\hat{X}^*_t = A\hat{X}_{t-1} \quad (7)$$

$$P^*_t = AP_{t-1}A^T + Q \quad (8)$$

A partir de la (ecuación 7) y (ecuación 8), se pronostica las estimaciones del estado y la covarianza hacia delante $t-1$ a t. La matriz A relaciona el estado en el momento previo $t-1$ con el estado al momento actual t, esta matriz podría ser alterada para los diferentes momentos en el tiempo t. Q representa la covarianza de la perturbación aleatoria del proceso que trata de estimar el estado.

$$K_t = P^*{}_t H^T \left(HP^*{}_t H^T + R\right)^{-1} \quad (9)$$

$$\hat{X}_t = \hat{X}^*{}_t + K_t \left(Z_t - H\hat{X}^*{}_t\right) \quad (10)$$

$$P_t = \left((I - K_t H)\, P^*{}_t\right) \quad (11)$$

La corrección de proyección del estado es la primera tarea que surge, ya que éste es el cálculo de la ganancia de Kalman, K_t (Ecuación 9). Este factor de ganancia es seleccionado con el fin de minimizar la covarianza del error de la nueva estimación del estado. El segundo paso consiste en medir el proceso para obtener Z_t y de esta manera generar una nueva estimación del estado que incorpora la nueva observación como en la (Ecuación 10), por último la tarea final se encargará de obtener una nueva estimación de la covarianza del error mediante la (Ecuación 11).

Posteriormente de cada par de actualizaciones, tanto del tiempo como de la medida el proceso será repetido tomando como punto de partida las nuevas estimaciones del estado y de la covarianza del error, siendo esta la naturaleza recursiva del filtro de Kalman, ilustrado en la (Figura 7).

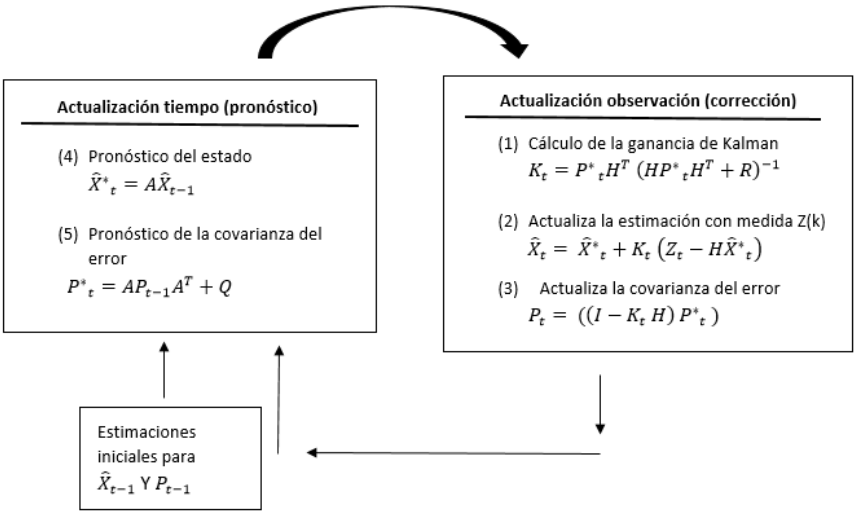

Figura 7. Ciclo completo del filtro de Kalman [44].

6.3.4 RGB-D SLAM

El paquete RGB-D SLAM es una solución SLAM (Localización y mapeo simultáneo) para cámaras RGB-D. Proporciona la pose actual de la cámara y permite crear una nube de puntos registrada o un *octomap*. Así mismo RGB-D SLAM permite adquirir rápidamente modelos 2D y 3D en color de objetos y escenas interiores con una cámara de estilo Kinect [45].

A alto nivel en [46], las tareas que ha de llevar a cabo un software RGB-D SLAM son principalmente cuatro: reconocer puntos de referencia, usarlos para calcular el desplazamiento, crear el mapa y reconocer el entorno el cual ha transcurrido. A continuación, en la (Figura 8) se ilustra la arquitectura típica de RGB-D SLAM.

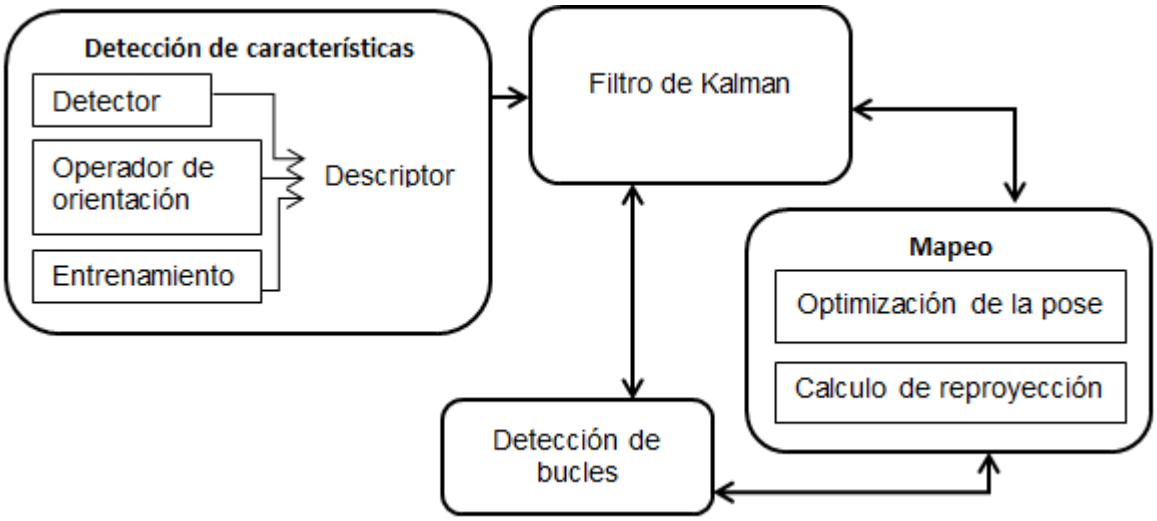

Figura 8. Arquitectura RGB-D SLAM [46].

Para su implementación el sistema SLAM RGB-D de código abierto se basa en un sensor RGB-D, el cual suministra información de profundidad y cuenta con la capacidad de extraer descriptores de características en imágenes a color que determina concretamente la asociación de datos. En [47] las imágenes de profundidad RGB y registradas del entorno generan un modelo de nube de puntos 3D, sin embargo, el análisis de los temas ROS suscritos por el nodo RGB-D SLAM y el nodo del servidor *octomap* permiten identificar su interconexión y obtener un mapa de cuadricula 2D del entorno.

7. DESARROLLO METODOLÓGICO

En el desarrollo metodológico se presenta paso a paso los procesos necesarios para llevar a cabo las conexiones entre el computador remoto, el turtlebot3 Burger y los sensores LIDAR laser 360° y Microsoft Kinect, los cuales permitirán llevar a cabo inicialmente experimentos básicos de prueba. Posteriormente la implementación de los algoritmos filtro de Kalman y filtro de Partículas en SLAM (Mapeo y Localización Simultanea), así como las respectivas pruebas en variaciones de iluminación. Los resultados obtenidos a partir de las pruebas y experimentos planteados en este ítem se observan en la sección 8, implementación de algoritmos SLAM y resultados.

7.1 Pruebas Iniciales

7.1.1 Preparación de hardware y software

Figura 9. Conexión remota, modelo de configuración [40].

Para iniciar, fue necesaria la configuración de los requisitos del sistema operativo y del robot TurtleBot3 Burger, en el PC remoto, fue necesaria la instalación del sistema operativo UBUNTU 16.04, así mismo en la *Raspberry Pi* correspondiente al robot móvil. A partir de esto el paso a ejecutar es la instalación de ROS (Sistema Operativo Robótico), en ambos dispositivos, ya que los algoritmos SLAM se ejecutan en el mismo, al igual que las pruebas iniciales de sensores y de movimientos con el robot.

A continuación, se muestra el procedimiento para cargar las demostraciones incluidas en [48], en donde se realizan las pruebas iniciales de movimiento del robot móvil diferencial TurtleBot3 Burger, del sensor incluido en el robot, el sensor 360° *LIDAR* y de algunos algoritmos SLAM, vinculados al sensor *LIDAR*. Al encontrar el indicativo *[Remote PC],* se hará referencia a un código compilado en el computador remoto y el indicativo *[TurtleBot],* referencia una línea de código compilada en la terminal de la *Raspberry Pi* incluida en el robot móvil diferencial.

7.1.2 Proceso inicial de ejecución

En esta sección se visualiza el proceso inicial para lanzar los nodos principales que soportan la conexión y el enrutamiento con el Turtlebot3 Burger.

Se inicia el sistema operativo robótico (ROS), permitiendo comunicación entre los nodos.

[Remote PC]

- Roscore

Proceso de ejecución de los paquetes básicos para correr las aplicaciones del TurtleBot3 Burger.

[TurtleBot]

- roslaunch turtlebot3_bringup turtlebot3_robot.launch

Definición del modelo del robot TurtleBot3 implementado.

[Remote PC]

- export TURTLEBOT3_MODEL=burger

7.1.3 Procedimiento de visualización del trabajo del sensor LIDAR 360°

De acuerdo a lo mencionado en el ítem 7.1.2, se procede a describir los pasos necesarios para visualizar el proceso de captura de información que realiza el sensor LIDAR 360°, implementado con el fin de iniciar el entendimiento de las herramientas tanto físicas como digitales. El procedimiento no emplea una técnica SLAM y se lleva a cabo con librerías y repositorios disponibles en la página del fabricante.

Proceso de ejecución para la conexión remota.

[Remote PC]

- roslaunch turtlebot3_bringup turtlebot3_remote.launch

Continuamente, se ejecuta el nodo de la herramienta *Rviz,* el cual permite ilustrar el entorno en 2D en tiempo real obtenido por el sensor *LIDAR 360º* en el ordenador.

[Remote PC]

- rosrun rviz rviz -d
- rospack find turtlebot3_description`/rviz/model.rviz

7.1.4 Tele operación Turtlebot3 Burger

En esta sección se realizan los procedimientos para llevar a cabo el movimiento del robot operando desde el ordenador, el cual permite controlar el desplazamiento del mismo por medio de las teclas ‘w’ para avanzar, ‘x’ para retroceder, ‘d’ para girar hacia la derecha, ‘a’ para girar hacia la izquierda y ‘s’ para detener su desplazamiento. Estos procesos se llevan a cabo mediante librerías y repositorios existentes disponibles en la página del fabricante.

Definición del modelo implementado y nodo que permite ejecutar el proceso de operación de un sistema o máquina a distancia.

[Remote PC]

- export TURTLEBOT3_MODEL=burger
- roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

A continuación en la (Figura 10) se ilustra la interfaz de línea de comandos mediante un ejemplo, con el fin de evidenciar las teclas correspondientes de movimientos, velocidad angular y velocidad lineal en tiempo real obtenida por los encoders del Turtlebot3 Burger.

```
Control Your TurtleBot3!
---------------------------
Moving around:
        w
   a    s    d
        x

w/x : increase/decrease linear velocity (Burger : ~ 0.22, Waffle and Waffle Pi : ~ 0.26)
a/d : increase/decrease angular velocity (Burger : ~ 2.84, Waffle and Waffle Pi : ~ 1.82)

space key, s : force stop

CTRL-C to quit

currently:      linear vel 0.0   angular vel -2.2
currently:      linear vel 0.0   angular vel -2.3
currently:      linear vel 0.0   angular vel -2.4
```

Figura 10. Interfaz de línea de comandos que visualiza el proceso de tele operación.

7.2 Primeras pruebas experimentales Sensor Kinect

El modelo de Sensor Kinect implementado es el 1414, para su funcionamiento y lectura de datos es necesaria una conexión adecuada y se basa en dos funciones básicamente, conexión física y conexión lógica. La conexión física se basa en la conexión del adaptador con su cable de poder, el cuál suministra la suficiente corriente y tensión al dispositivo. Por otro lado, consta del puerto USB 2.0 que es el medio por el cuál ocurre la transmisión de datos. La conexión permite evidenciar su funcionamiento con cada una de las pruebas posteriormente mencionadas.

7.2.1 Pasos de prueba en el sistema operativo LINUX (Ubuntu 16.04)

El sistema operativo LINUX es el seleccionado para realizar el proceso de trabajo de este documento con el fin de llevar a cabo los procesos por medio de ROS (*Robot Operating System*). El sistema operativo LINUX debe estar presente tanto en el *PC Remote* de trabajo como en la tarjeta *Raspberry Pi* incluida en el *turtlebot3 Burger* y la conexión entre ambos sistemas se realiza por medio del protocolo SSH que permite el acceso remoto al servidor del ordenador *Raspberry Pi* gracias a la dirección IP, los dispositivos deben estar enlazados a una misma red para que la conexión sea efectiva.

Para poder utilizar la interfaz del sensor Microsoft Kinect desde el *PC Remote* conectado a través de SSH con la tarjeta *Raspberry Pi* es necesario crear los archivos .rules respectivos, ya que el sistema debe reconocer las dependencias del dispositivo. A partir de ello se agregan los archivos mencionados en el ítem 7.2.1.1.

7.2.1.1 Archivos *.rules*

Fue necesario crear dos archivos con reglas para el administrador de dispositivos Linux, estos son *51-kinect.rules* y *66-kinect.rules*. Estos dos archivos permiten el funcionamiento del Kinect, brindando al sistema las reglas para el correcto funcionamiento del dispositivo [49].

7.2.1.2 Pasos de instalación libfreenect:

En la parte experimental se procede a realizar la configuración del Kinect e instalar las pertinentes librerías (*OpenKinect/libfreenect*) para su correcto funcionamiento en el ordenador bajo los pasos enseñados a continuación. Cabe resaltar que, aunque las librerías permiten realizar capturas de profundidad con el dispositivo Kinect aún no se vincula con ROS (*Robot Operating System*) ni emplea una técnica SLAM.

Se recomienda descargar el OpenKinect desde el repositorio GitHub.

- git clone git://github.com/OpenKinect/libfreenect.git

Una vez descargado, acceder al directorio *libfreenect* desde una ventana de terminal de comandos y construir el paquete de instalación desde las fuentes.

7.3 Pruebas experimentales SLAM mediante sensor Microsoft Kinect

A partir de este ítem, se plantean las pruebas a realizar mediante el dispositivo Microsoft Kinect aplicada al SLAM (Localización y modelado simultáneos en español) en ROS (*Robot Operating System*). Para ello es necesario consultar acerca de los paquetes de ROS que permiten operar con el dispositivo Kinect en

el sistema operativo *Linux – Ubuntu 16.04*. A partir de la investigación fueron encontrados dos paquetes que hacen esto posible, *freenect_launch* y *openni_launch* [50]**.** El paquete *freenect_launch* contiene algunas modificaciones de *openni_launch* además de ejecutar el controlador basado en *libfreenect* trabajado en el desarrollo metodológico en el ítem 7.2.1.2. Por estos motivos fue seleccionado para ser utilizado en el proyecto.

Se puede acceder a las herramientas *frenect_launch* y *openni_launch* en la plataforma wiki.ros dentro de los repositorios de *GitHub*.

7.3.1 Paquete freenect_launch

Este paquete contiene todas las dependencias necesarias para lanzar el nodo de activación del sensor, así como los parámetros de calibración, que en caso de que ciertos de estos parámetros no sean encontrados define unos parámetros por defecto e inicializa el mapeo. Luego de lanzar este nodo, se puede ejecutar el programa *Rviz*, el cual muestra gráficamente los *topics* que están publicando en el momento.

El comando para lanzar el archivo principal del paquete *freenect_launch* es el siguiente:

- roslaunch freenect_launch freenect.launch

7.3.2 Definición de trayectorias de mapeo

Para llevar a cabo las pruebas experimentales se requiere definir la trayectoria que seguirá el robot móvil en los procesos de mapeo, dichas pruebas se llevan a cabo en el laboratorio de robótica de la Universidad Santo Tomás. Esto permite realizar mediciones de localización y trayectoria además de la comparación de los datos obtenidos experimentalmente con las medidas reales, concluyendo en el cálculo de los errores de medición que se efectúan en las pruebas experimentales. Por medio de estos cálculos se podrá evaluar el funcionamiento de los algoritmos implementados.

Inicialmente se plantea una trayectoria, definida como “trayectoria 1” visualizada en la (Figura 11) físicamente y en el plano de medidas reales en la (Figura 12), sin embargo, luego de un proceso de prueba en el “experimento 1 (Creación nube de puntos 3D)” en la sección 8.3 se observa que el robot no logra captar secciones del laboratorio, por ello se opta por rediseñar una nueva trayectoria, definida como “trayectoria 2” la cual abarca un recorrido más extenso del laboratorio, visualizada en la (Figura 13) físicamente y en el plano de medidas reales en la (Figura 14). Esta es seleccionada para las distintas pruebas experimentales y así mismo, la

implementación de los algoritmos Filtro de Partículas y Filtro de Kalman en el turtlebot3 Burger.

- **Trayectoria 1**

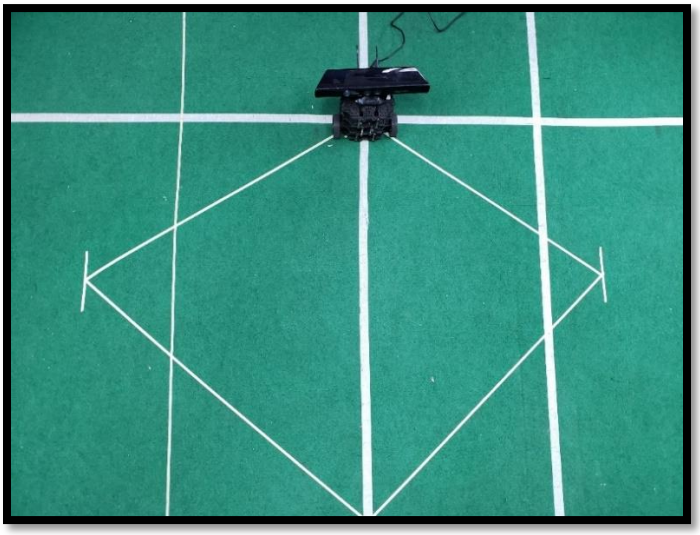

Figura 11. Trayectoria 1 Mapa Inicial

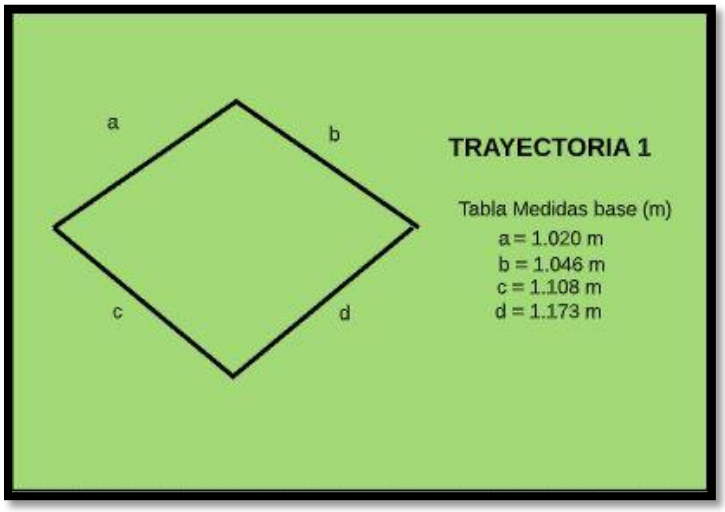

Figura 12. Plano de medidas Trayectoria 1

- **Trayectoria 2**

Figura 13. Trayectoria 2 mapa final.

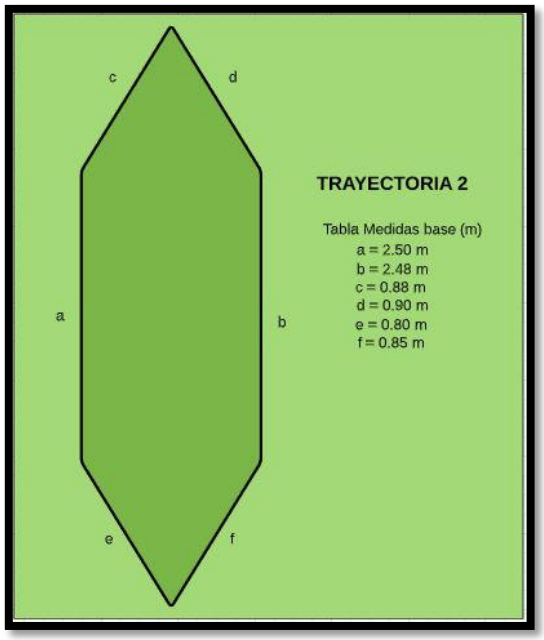

Figura 14. Plano de medidas trayectoria.

En la (Figura 15) se puede observar el plano del laboratorio de robótica en 2D. En él, se pueden detallar los objetos que se encuentran en su interior, esto con el fin de entender las figuras que se obtienen en los mapeos 3D y tener una idea de lo que el robot está censando en los procesos SLAM.

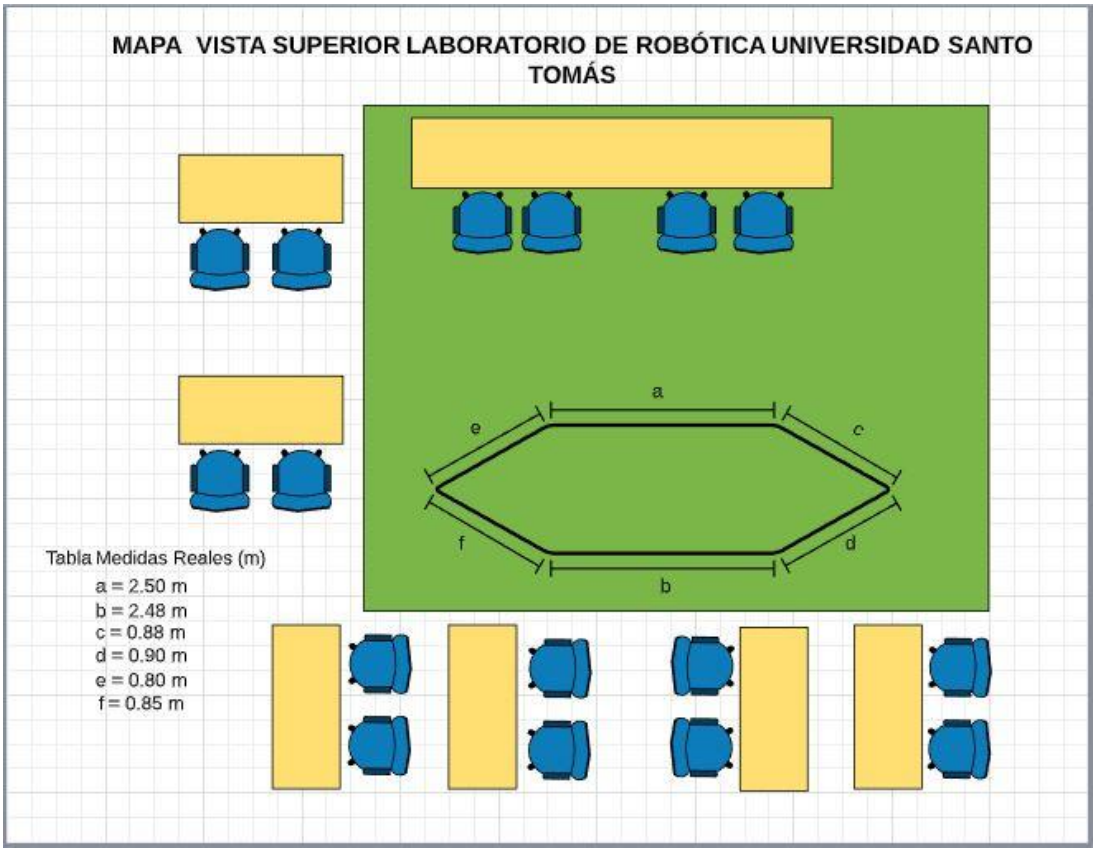

Figura 15. Plano de medidas laboratorio USTA (Trayectoria 2).

8. IMPLEMENTACIONES DE ALGORITMOS SLAM Y RESULTADOS

En esta sección, se encuentran detalladamente los análisis de los experimentos realizados, con el fin de determinar los parámetros que influyen en la técnica SLAM (Localización y mapeo simultáneos) ante variaciones de iluminación mediante un sensor Microsoft Kinect en mapeos 2D y 3D.

Mediante los datos obtenidos en las implementaciones SLAM del Filtro de Partículas y el Filtro de Kalman, se determina el valor RECM (Raíz del Error Cuadrático Medio), con el fin de hallar el rango de error e ilustrar el desfase que presentan los datos de estimación de trayectoria del robot contra los datos reales de trayectoria recorrida por el mismo.

8.1 Visualización trabajo del sensor 360° LIDAR

A través de esta sección se realiza el proceso de visualizacion del trabajo del sensor LIDAR 360° cuya finalidad consiste en lograr contextualizar los conceptos teorico-practicos aplicados al TurtleBot3 Burger contando con su sensor original. A partir de los pasos mencionados en el ítem 7.1.3 del desarrollo metodológico, se logra obtener los paquetes necesarios para realizar esta ejecución, para esta prueba se utiliza una posición fija y no se emplea ninguna de las trayectorias definidas. En la (Figura 16) se observa la lectura 2D del sensor LIDAR 360° en el programa *Rviz* en el *PC Remote,* en ella se encuentran los obstáculos en color rojo detectados en el laboratorio de robótica por el turtlebot3 Burger del entorno en tiempo real.

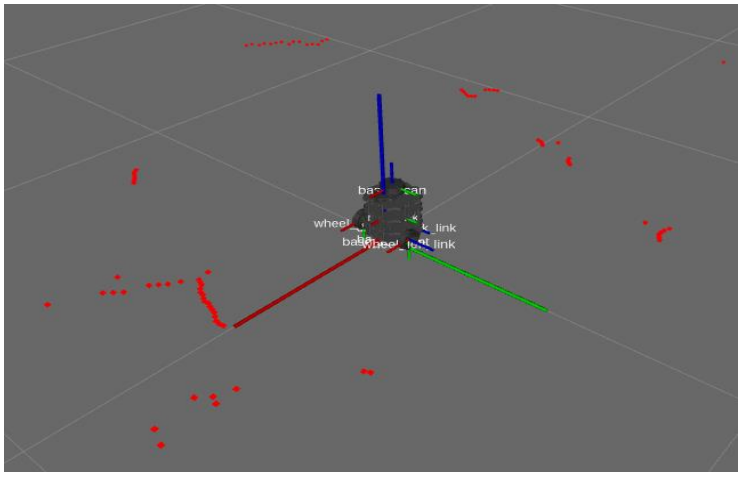

Figura 16. Visualización en RViz del mapeo del sensor LIDAR 360°.

8.2 Pruebas sensor Microsoft Kinect en S.O. LINUX (Ubuntu 16.04)

Por medio de esta sección de experimentación se obtiene la aplicación de pruebas iniciales del sensor Microsoft Kinect con la implementación de OpenKinect; como resultado se logran 3 principales pruebas: *freenect-cppview*, *freenect-glpclview* y *freenectglview*.

En la primera prueba, se visualiza la salida en modo de video RGB en base a los intervalos de los valores que ofrece la cámara, esto se muestra en la (Figura 17), al oprimir la tecla 'f' se muestra la salida en modo de video de la cámara de profundidad (infrarrojos), este cambio se muestra en la (Figura 18). Las imágenes son captadas en horas del día con iluminación artificial por medio de la librería *freenectglview*.

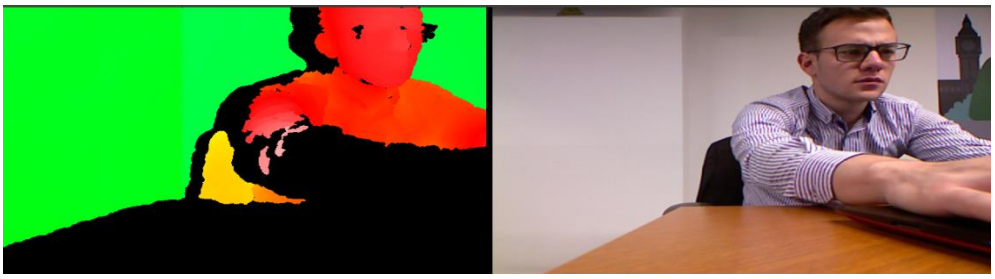

Figura 17. Prueba Kinect #4.

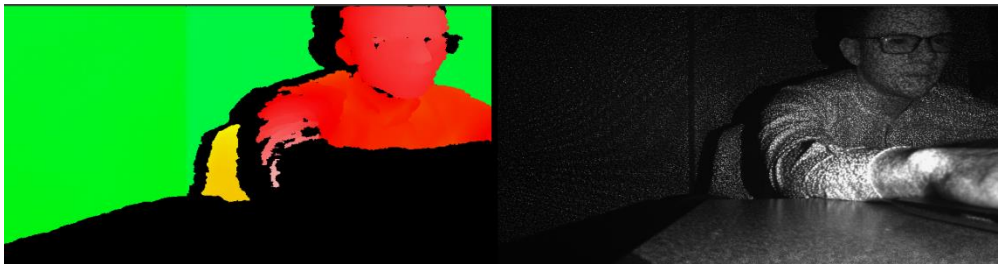

Figura 18. Prueba Kinect #5.

En la segunda prueba, por medio de *freenect-cppview* se obtiene el acceso al led de actividad del frontal del dispositivo y el motor, que ofrece la capacidad de cambiar el color del led y activar el motor para cambiar la posición vertical del dispositivo dentro de un ángulo de ±30°, esta aplicación no cuenta con una imagen de salida.

Como tercera y última prueba con la librería *libfreenect*, *freenect-glpclview*, ofrece una mezcla de las dos imágenes para hacer un alzamiento 3D de la escena con la textura de la imagen RGB sobrepuesta, como se denota en la (Figura 19). Aquí se pueden utilizar las teclas 'w' y 's' para aumentar o decrementar el zoom de la imagen. También la tecla 'c' para activar la textura o no.

Figura 19. Prueba Kinect #6.

Por último, se puede deducir que mediante las pruebas anteriores se logra visualizar todas las configuraciones de hardware para el Kinect V1 y la prueba de cada una de ellas en el sistema operativo Linux Ubuntu 16.04 siendo el S.O utilizado para realizar todas las implementaciones en este desarrollo.

8.3 Experimento 1 (SLAM 3D)

A través de la ejecución de este experimento adicional se logra implementar la técnica SLAM 3D mediante la nube de puntos, los mapas se visualizan en el *PC remote* a través de la herramienta RTabMap, la cual permite visualizar el entorno desde cualquier ángulo de visión. En esta sección se realizan capturas de vista superior y lateral ya que estas permiten visualizar una mayor parte del entorno con el fin de verificar el alcance de mapeo del sensor Microsoft Kinect, donde se observan los puntos desconocidos del entorno en las secciones resaltadas en color negro.

Para realizar el mapeo de nube de puntos fue necesario implementar un archivo lanzador que permitiera acelerar la frecuencia de envío de mensajes del Kinect en el robot a 5 Hz y sincronizar previamente los *topics* de la cámara. Este lanzador pone en marcha el nodo principal *freenect_launch* (anteriormente explicado en el ítem 7.3.1) y se brindan los parámetros necesarios para su funcionamiento, así como la asignación de *topics* de entrada y de salida.

8.3.1 Mapas (Nube de puntos 3D) en luz natural trayectoria 1

En la (Figura 20) y la (Figura 21) se observan las vistas superior y lateral respectivamente del lugar donde se realiza el experimento, realizando el recorrido de la “trayectoria 1” en condiciones de luz natural. La trayectoria obtenida por medio de odometría visual se denota en color rosa.

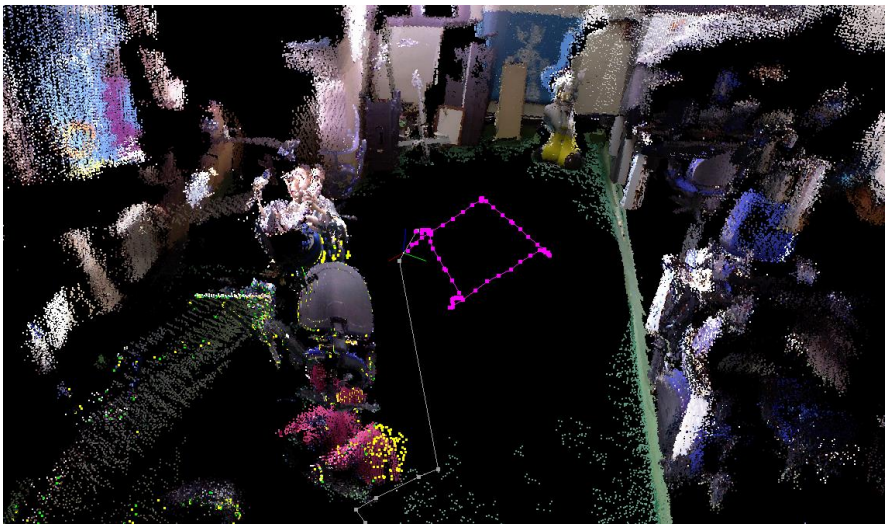

Figura 20. SLAM vista lateral (Trayectoria 1) luz natural sala de robótica.

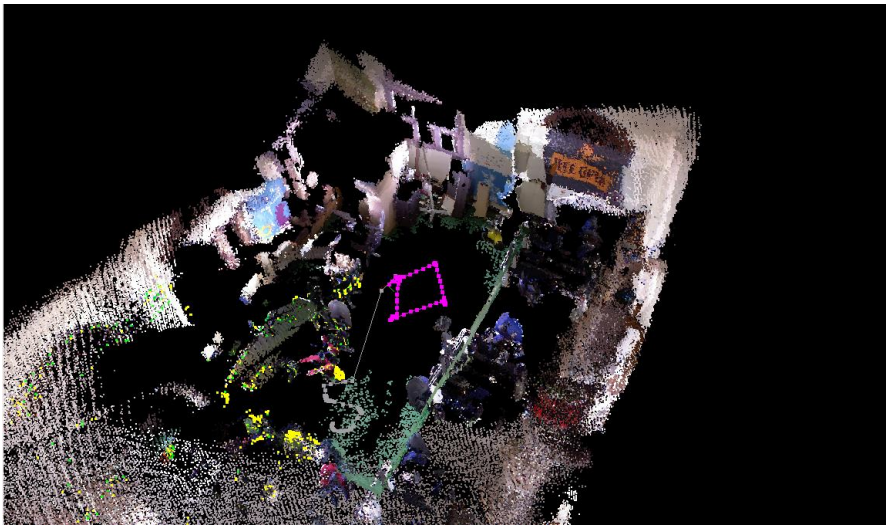

Figura 21. SLAM vista superior (Trayectoria 1) luz natural sala de robótica.

8.3.2 Mapas (Nube de puntos 3D) en luz artificial trayectoria 1

En la (Figura 22) y la (Figura 23) se observan las vistas lateral y superior respectivamente del lugar donde se realiza el experimento, realizando el recorrido de la "trayectoria 1" en condiciones de luz artificial. La trayectoria obtenida en este experimento se denota en color rojo y la trayectoria obtenida en luz natural en color blanco, siendo esta última guardada a partir del experimento expuesto en el ítem 8.1.1 para su respectiva comparación.

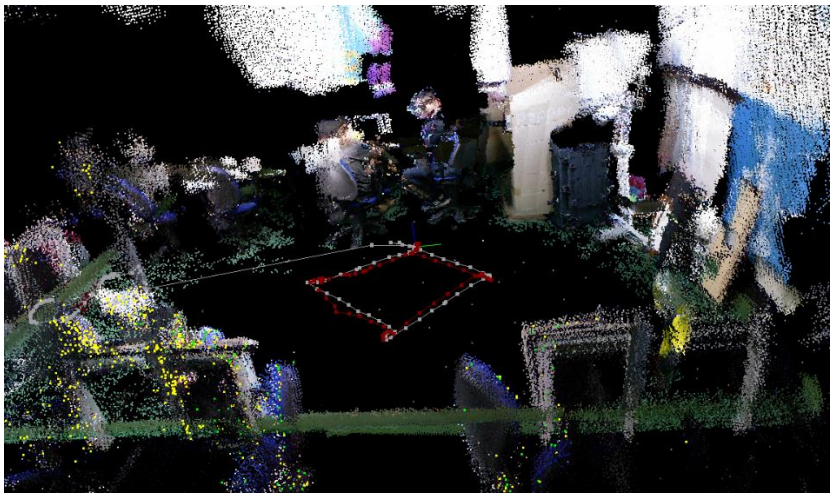

Figura 22. SLAM vista lateral (Trayectoria 1) luz artificial sala de robótica.

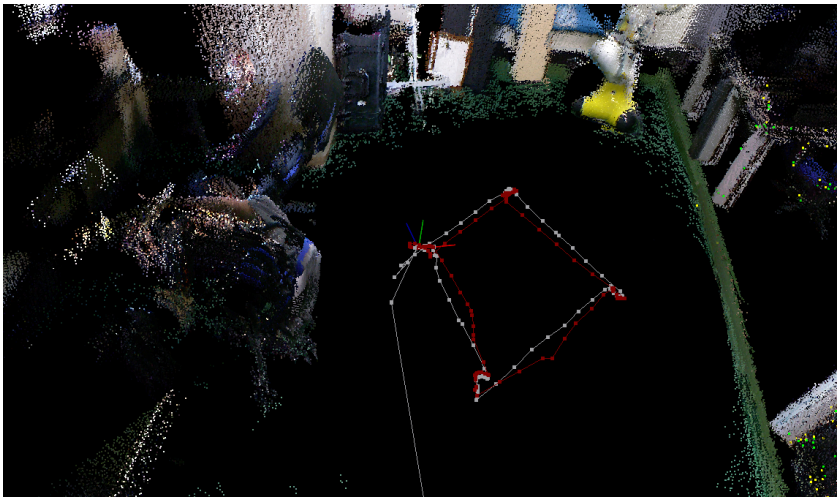

Figura 23. SLAM vista superior (Trayectoria 1) luz artificial sala de robótica.

Los resultados obtenidos y visualizados anteriormente en los ítems 8.3.1 y 8.3.2 fueron suficientes para redefinir una mejor trayectoria de mapeo visualizadas en los ítems 8.3.3 en luz natural y 8.3.4 en luz artificial. Estos reflejan la importancia de una trayectoria más extensa, en la que se logra mayor cantidad de datos en la trayectoria y aumento de captura de profundidad con los objetos debido a la reducción de distancia entre los mismos y el sensor.

8.3.3 Mapas (Nube de puntos 3D) en luz natural trayectoria 2

En la (Figura 24) y la (Figura 25) se observan las vistas lateral y superior respectivamente del lugar donde se realiza el experimento, realizando el recorrido de la “trayectoria 2” en condiciones de luz natural. La trayectoria obtenida en este expe-

rimento se denota en color amarillo y se resalta por medio de unos indicadores resaltados dentro del mapa en color blanco.

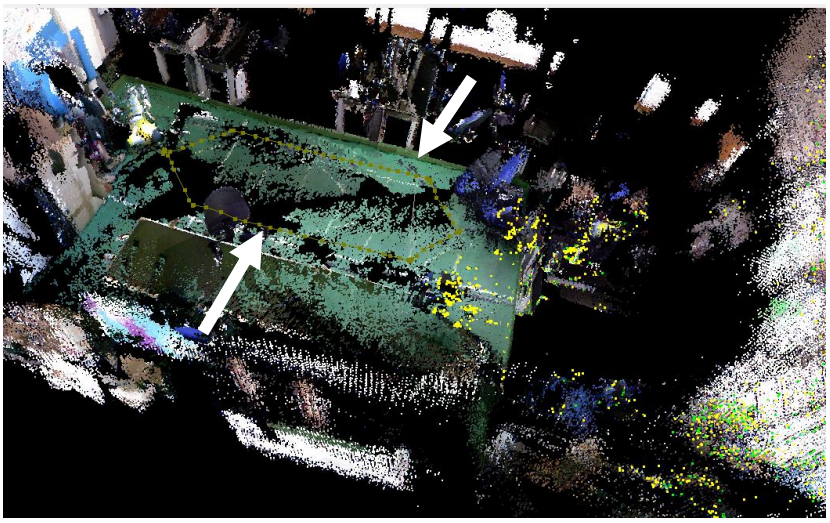

Figura 24. SLAM vista lateral (Trayectoria 2) luz natural sala de robótica.

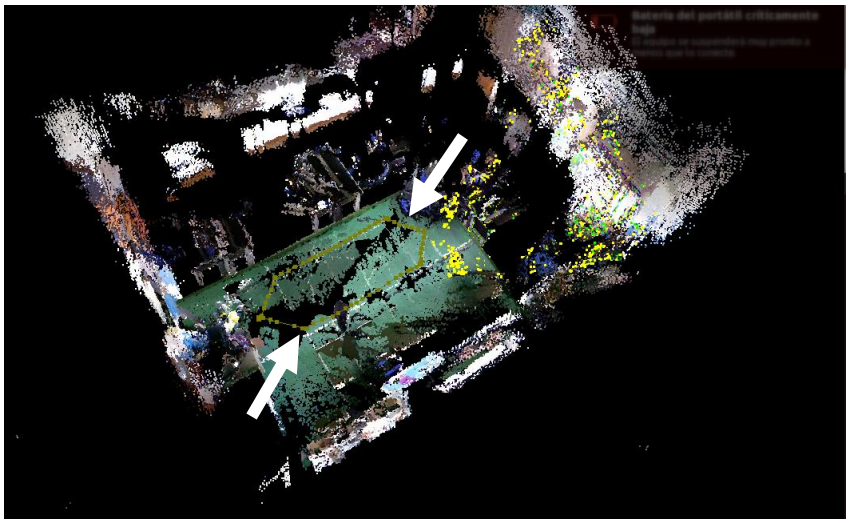

Figura 25. SLAM vista superior (Trayectoria 2) luz natural sala de robótica.

8.3.4 Mapa (Nube de puntos 3D) en luz natural trayectoria 2

En la (Figura 26) y la (Figura 27) se observan las vistas lateral y superior respectivamente del lugar donde se realiza el experimento, realizando el recorrido de la “trayectoria 2” en condiciones de luz artificial. La trayectoria obtenida en este experimento se denota en color morado y es resaltada por indicadores en color

blanco. En este punto se observa la comparación con la trayectoria obtenida en el ítem 8.3.2 que corresponde a la prueba de “trayectoria 1” en luz artificial encontrada en color gris y resaltada con indicadores en color rojo.

.

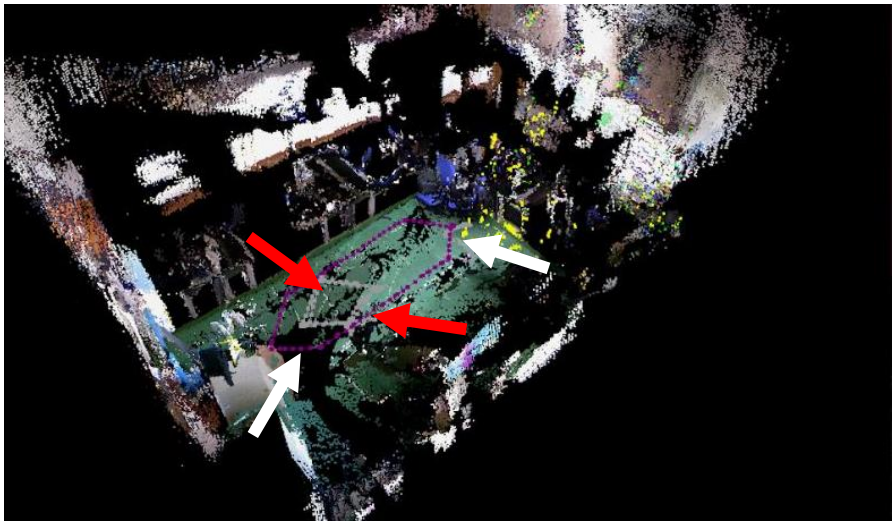

Figura 26. SLAM vista lateral (Trayectoria 2) luz artificial sala de robótica.

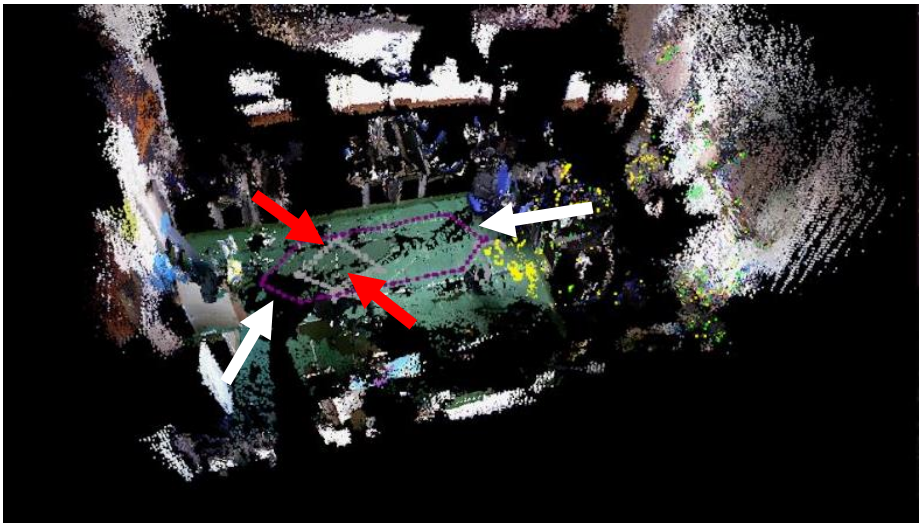

Figura 27. SLAM vista superior (Trayectoria 2) luz artificial sala de robótica.

A partir de los experimentos realizados en el ítem 8.3 se observa mayor captura de profundidad del entorno en la “trayectoria 2”, lo cual disminuye considerablemente los puntos de referencia desconocidos por el sensor Microsoft Kinect en comparación con los mapas obtenidos en la “trayectoria 1”, siendo la “trayectoria 2” definida para continuar con los experimentos planteados en los ítems 8.4 y 8.5.

8.4 Experimento 2 - PointCloud_To_LaserScan

El objetivo de este experimento es implementar una técnica que realice la transformación del mapeo 3D del sensor Kinect en un mapeo láser 2D. Esto debido a que el algoritmo Gmapping es una librería que utiliza como entradas información periódica sobre una estimación de la posición del robot (odometría – calculada por los encoders del motor) y la medida de un sensor láser (vector de distancia) en un entorno con superficie plana [51].

En la (Figura 28) se puede observar la señal de entrada del paquete P*ointcloud_to_laserscan*, siendo esta el *topic PointCloud2* (superficie blanca) mapeo de nube de puntos 3D y su respectiva salida (superficie multicolor) la cual corresponde a un escaneo 2D. Esta imagen es captada directamente con el ángulo frontal del sensor Microsoft Kinect a una altura y rango determinado y es visualizada por medio de la herramienta *Rviz*.

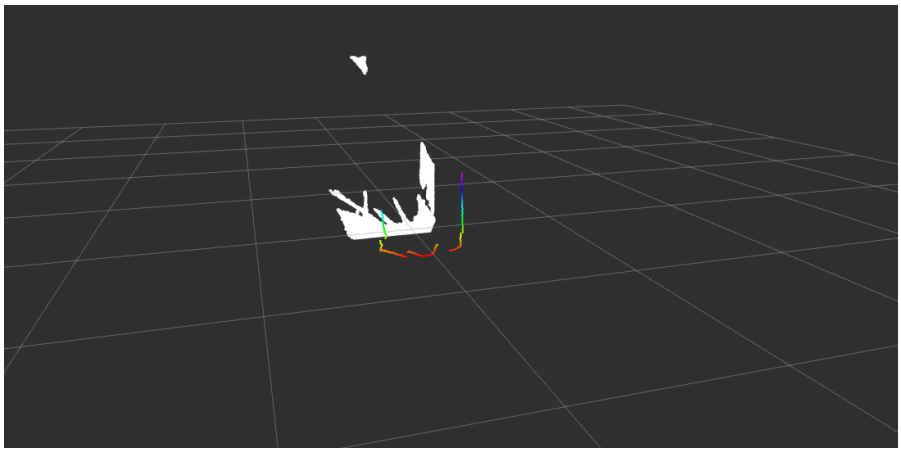

Figura 28. Conversión escaneo nube de puntos 3D a escaneo laser 2D.

Al finalizar la prueba de conversión y una vez obtenida la salida de tipo LaserScan a través del *topic* /camera/scan, se procede a realizar la implementación al SLAM 2D, que se presenta en el algoritmo Gmapping ilustrada en el ítem 8.6.

8.5 Experimento 3 (Mapeo 2D)

A través de la ejecución de este experimento adicional se implementa un sistema RGB-D SLAM en dos locaciones distintas a partir del sensor Microsoft Kinect v1 con el fin de extraer descriptores de características en imágenes a color del entorno para generar un modelo de nube de puntos 3D, y un mapa de cuadricula 2D.

8.5.1 Prueba A (Casa [Autor])

En esta prueba se logra generar un mapa de nube de puntos 3D visualizada en la (Figura 29), un mapa de cuadricula 2D visualizada en la (Figura 30) y la estimación de posición del *TurtleBot Burger3* dentro de los mismos en un entorno de trabajo distinto al de la Universidad Santo Tomas. Las imágenes presentadas en esta sección se ilustran desde una vista superior y lateral en la herramienta *Rviz* por medio del paquete RGBD-SLAM descrito en el ítem 6.4.4.

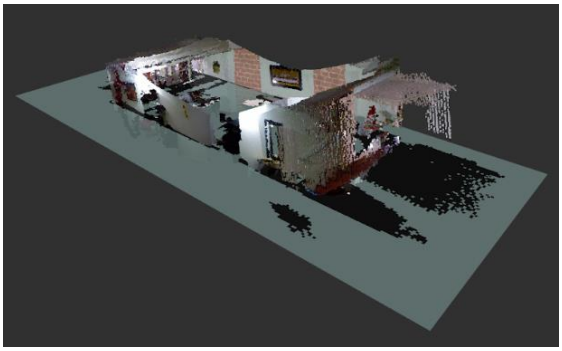

Figura 29. Mapa nube de puntos 3D RGB-D.

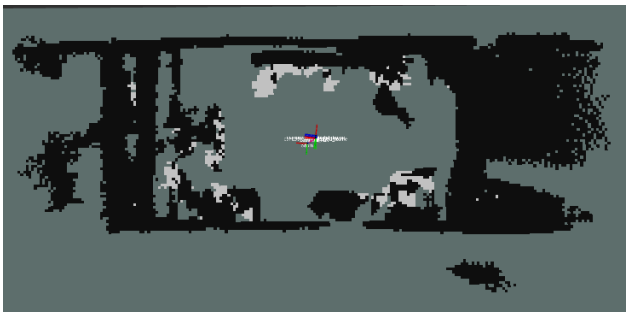

Figura 30. Mapa de cuadricula 2D RGB-D.

8.5.2 Prueba B (Sala de robótica Universidad Santo Tomás)

En esta prueba se logra generar un mapa de cuadricula 2D y la estimación de posición del *TurtleBot Burger3* dentro del mismo visualizado en la (Figura 31). Las secciones de flechas rojas indican la estimación del movimiento que se encontraba realizando el robot dentro del laboratorio de la universidad Santo Tomás manipulado por tele operación desde la *PC remote*. La imagen presentada en esta sección se ilustra desde una vista superior en la herramienta *Rviz* por medio del paquete RGBD-SLAM descrito en el ítem 6.4.4.

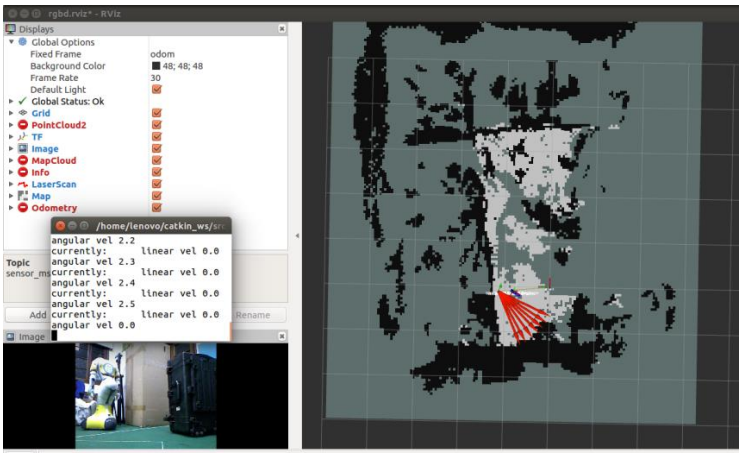

Figura 31. Mapa de cuadricula 2D RGB-D del laboratorio.

En la (Figura 31) se logra evidenciar la similitud con el plano del laboratorio de la Universidad Santo Tomás (Figura 15) en el cual se observa en pixeles de color blanco las áreas sin obstáculos y en color negro las áreas ocupadas por obstáculos, así mismo se logra visualizar la ausencia de píxeles como los lugares inexplorados, debido a que no entran en el rango de captura del sensor Microsoft Kinect.

8.6 Experimento 4 - SLAM Gmapping (Filtro de partículas)

En este experimento se realiza la implementación del Filtro de Partículas utilizando los parámetros internos determinados provenientes del repositorio de Github, con el fin de determinar los parámetros influyentes en la técnica SLAM obtenidos en dos etapas de prueba, ambiente interior con luz natural y ambiente interior con luz artificial.

Inicialmente se plantean las pruebas experimentales correspondientes al algoritmo SLAM Gmapping, las cuales se dividen en dos etapas principales, luz natural ubicada en el ítem 8.6.1 y luz artificial ubicada en el ítem 8.6.2. En el presente experimento se realiza la implementación del algoritmo Filtro de Partículas, el cual se encuentra localizado dentro del paquete SLAM Gmapping, para ello se realiza el proceso de SLAM 2D en la sala de robótica de la Universidad Santo Tomás.

8.6.1 SLAM algoritmo Gmapping luz natural

Inicialmente al ejecutar el algoritmo SLAM Gmapping mediante el sensor Microsoft Kinect, se obtiene el mapa 2D resultante del recorrido de la “trayectoria 2” y la posición del robot a lo largo del mismo en laboratorio de robótica de la Universidad Santo Tomás ilustrado en la (Figura 32). La imagen presentada en esta sección se ilustra desde una vista superior en la herramienta Rviz.

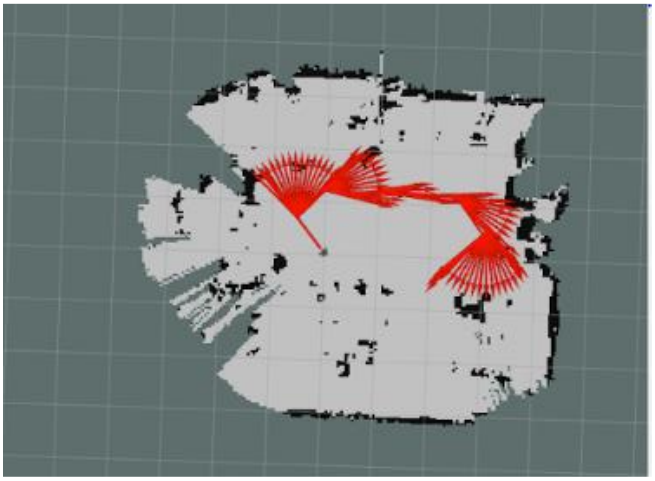

Figura 32. Mapa algoritmo Gmapping (luz natural).

8.6.1.1 Análisis teórico pruebas experimentales algoritmo Gmapping

Inicialmente, ilustrado en la (Figura 33) se encuentran las medidas reales de la "trayectoria 2" y las medidas de estimación de posición del robot encontradas a partir de las entradas del proceso, siendo estas la información periodica de estimación de posición del robot (odometría – calculada por los encoders del *Turtlebot3 Burger*) y el escaneo laser (vector de distancia – obtenido por el proceso del paquete *Pointcloud_to_LaserScan*).

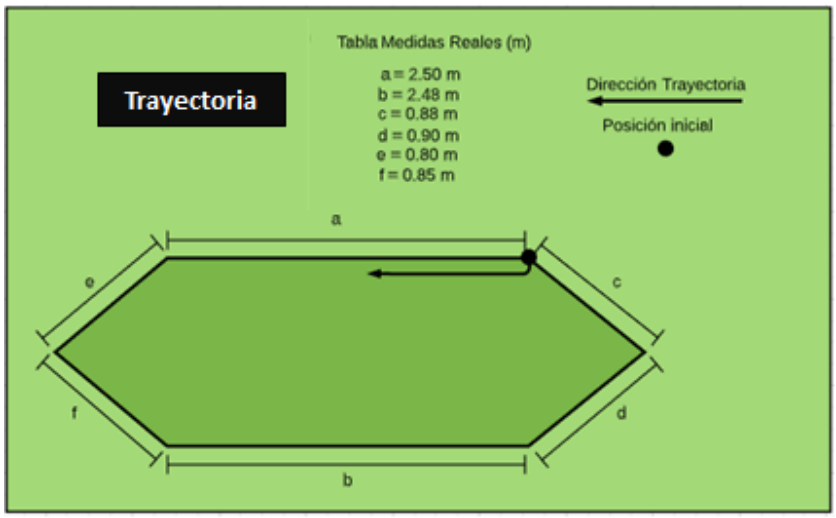

Figura 33. Plano de medidas (Algoritmo Gmapping - Luz natural).

A continuación, ilustrado en la "Tabla 1", se presenta la comparación de los valores de medición reales de la trayectoria recorrida y los valores estimados de posición del robot móvil, visualizando gráficamente el desfase presentado en la (Figura 34).

Tabla 1. Medidas reales vs estimación de posición de odometría en luz natural (Algoritmo Gmapping).

Tabla Medidas Estimación Robot (m)	***Tabla Medidas reales (m)***
a = 2.3792	*a = 2.50*
b = 2.4585	*b = 2.48*
c = 0.8473	*c = 0.88*
d = 0.9301	*d = 0.90*
e = 0.7642	*e = 0.80*
f = 0.923	*f = 0.85*

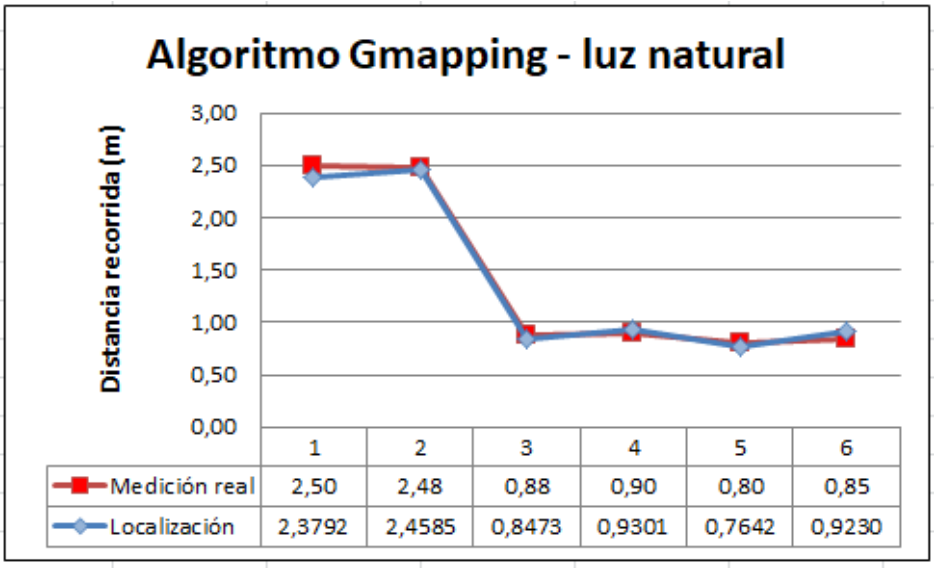

Figura 34. Desfase de medición real vs estimación de posición en luz natural (Algoritmo Gmapping).

Al obtener el desfase de medición real vs estimación de posición en el algoritmo Gmapping en luz natural, se procede a realizar el cálculo de la raíz del error cuadrático medio presentado en la “tabla 2”.

Tabla 2. Datos de error cuadrático medio en luz natural (Algoritmo Gmapping).

Valor Observado	Valor Predicho	Diferencia	***RECM* *(Raíz del Error Cuadrático Medio)***
2,5	*2,3792*	0,1208	**0,062770521**
2,48	*2,4585*	0,0215	
0,88	*0,8473*	0,0327	
0,9	*0,9301*	-0,0301	
0,8	*0,7642*	0,0358	
0,85	*0,923*	-0,073	

8.6.2 SLAM algoritmo Gmapping luz artificial

Luego de la implementación en luz natural, se procede a ejecutar el algoritmo SLAM Gmapping mediante el sensor Microsoft Kinect en el ambiente con luz artificial. A partir de ello se obtiene el mapa 2D resultante del recorrido de la "trayectoria 2" y la posición del robot a lo largo del mismo en la sala de robótica de la Universidad Santo Tomás ilustrado en la (Figura 35). La imagen presentada en esta sección se ilustra desde una vista superior en la herramienta Rviz.

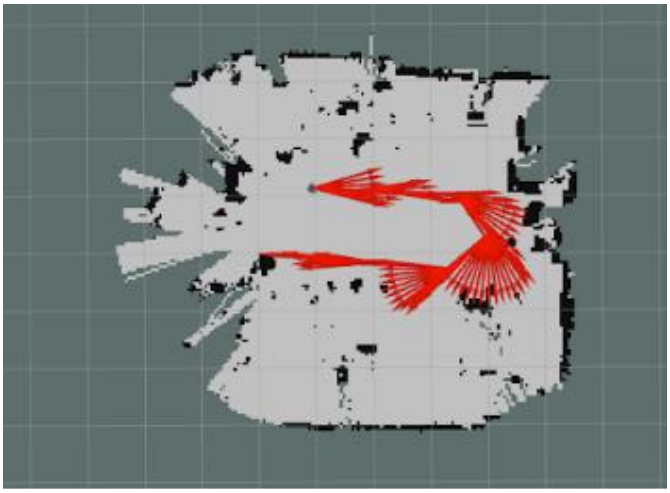

Figura 35. Mapa algoritmo Gmapping (luz artificial).

8.6.2.1 Análisis teórico pruebas experimentales algoritmo Gmapping

Continuando, ilustrado en la (Figura 36) se encuentran las medidas reales de la "trayectoria 2" y las medidas de estimación de posición del robot encontradas a partir de las entradas del proceso, siendo estas la información periodica de estimación de posición del robot (odometría – calculada por los encoders del

Turtlebot3 Burger) y el escaneo laser (vector de distancia – obtenido por el proceso del paquete *Pointcloud_to_LaserScan*).

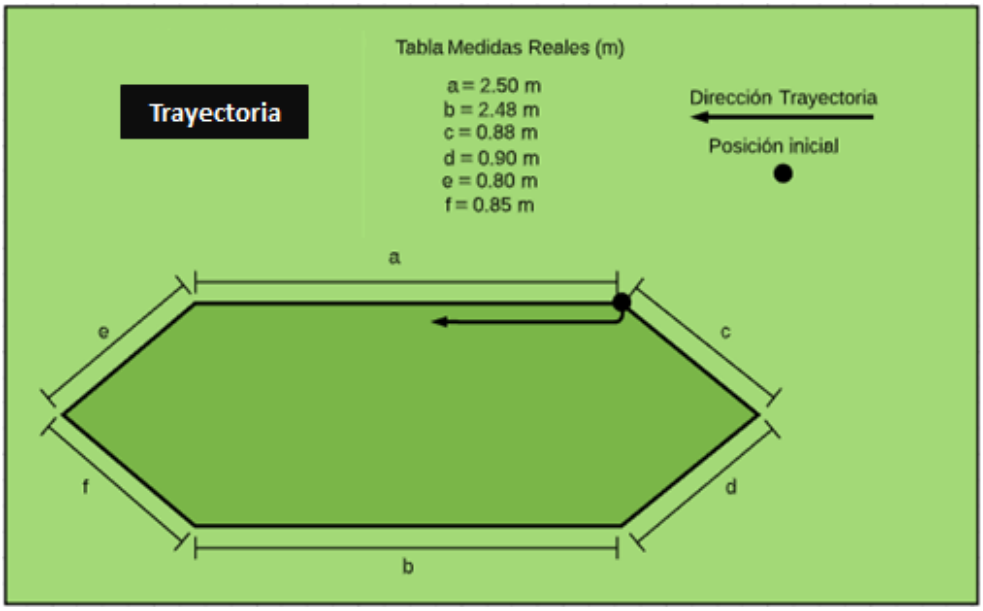

Figura 36. Plano de medidas (Algoritmo Gmapping - Luz artificial).

A continuación, ilustrado en la "Tabla 3", se presenta la comparación de los valores de medición reales de la trayectoria recorrida y los valores estimados de posición del robot móvil, visualizando gráficamente el desfase presentado en la (Figura 37).

Tabla 3. Medidas reales vs estimación de posición de odometría en luz artificial (Algoritmo Gmapping).

Tabla Medidas Estimación Robot (m)	***Tabla Medidas reales (m)***
a = 2.4552	*a = 2.50*
b = 2.502	*b = 2.48*
c = 0.895	*c = 0.88*
d = 0.954	*d = 0.90*
e = 0.812	*e = 0.80*
f = 0.8216	*f = 0.85*

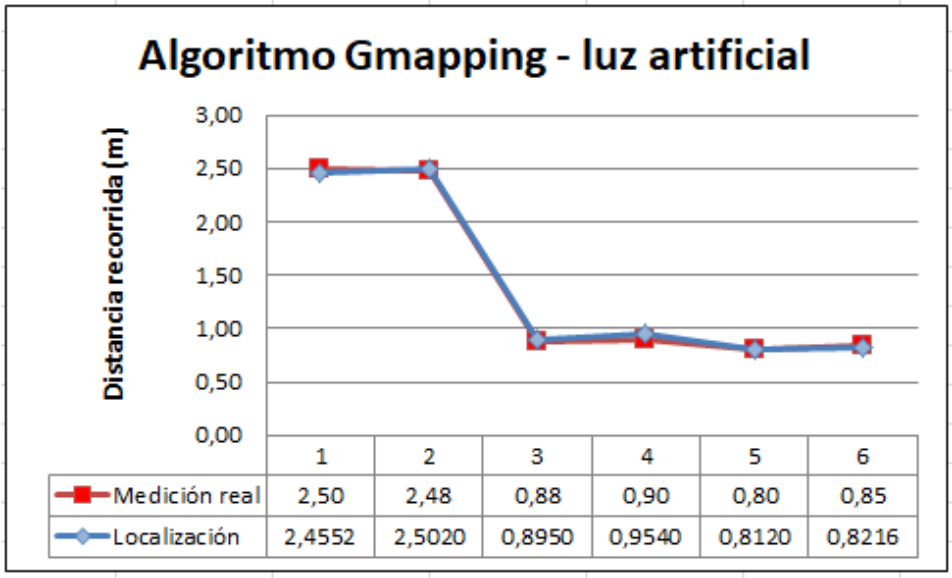

Figura 37. Grafica comparativa de medición real vs estimación de posición en luz artificial (Algoritmo Gmapping).

Al obtener el desfase de medición real vs estimación de posición en el algoritmo Gmapping en luz artificial, se procede a realizar el cálculo de la raíz del error cuadrático medio presentado en la "tabla 4".

Tabla 4. Datos de error cuadrático medio en luz artificial (Algoritmo Gmapping).

Valor Observado	Valor Predicho	Diferencia	***RECM* (Raíz del Error Cuadrático Medio)**
2,5	*2,4552*	0,0448	**0,0331225**
2,48	*2,502*	-0,022	
0,88	*0,895*	-0,015	
0,9	*0,954*	-0,054	
0,8	*0,812*	-0,012	
0,85	*0,8216*	0,0284	

8.7 Experimento 5 – SLAM (Filtro de Kalman)

Inicialmente se plantean las pruebas experimentales correspondientes al algoritmo EKF (Filtro de Kalman Extendido), las cuales se dividen en dos etapas principales, luz natural ubicada en el ítem 8.7.1 y luz artificial ubicada en el ítem 8.7.2. Por medio de las implementaciones se realiza la captura de datos de estimación de posición del *Turtlebot3 Burger* obtenidas por medio del algoritmo EKF, basados en la medición de distancia desde el punto de partida *(*identificado con un punto negro*)* hasta cada uno de los vértices de la "trayectoria 2" *(*puntos de colores*)* encontrados en la (Figura 38) y la (Figura 40).

Así mismo, se llevan a cabo los procesos de análisis y comparación de los resultados obtenidos con las medidas reales del entorno de trabajo y el cálculo del valor *RECM (Raíz del Error Cuadrático Medio)*.

8.7.1 Filtro de Kalman luz natural

A continuación, se visualizará en la "tabla 5" los datos de estimación de posición del *Turtlebot3 Burger* obtenidos por medio de la terminal de comandos a partir de las pruebas experimentales de medición implementadas por medio del algoritmo EKF en condiciones de luz natural, para cada una de las trayectorias desde el punto inicial hasta el punto final.

Tabla 5. Datos de traslación Algoritmo EKF (luz natural)

Trayectoria recorrida	**Translación 'X'**	**Translación 'Y'**	**Translación 'Z'**
(Punto inicial vs punto rojo)	2,51631498337	3,01382017136	0,0
(Punto inicial vs punto azul)	3,13361954689	3,01842832565	0,0
(Punto inicial vs punto blanco)	2,66223096848	2,96516728401	0,0
(Punto inicial vs punto gris)	1,27768814564	2,44661235809	0,0
(Punto inicial vs punto Amarillo)	0,92998564243	1,93637502193	0,0

En la (Figura 38) se encuentra el plano correspondiente a la "trayectoria 2", donde se visualizan los datos de estimación de posición del robot desde su punto inicial (punto negro, entre los segmentos 'a' y 'c') hasta cada una de las posiciones finales (Puntos de colores), en la fase de luz natural. Estos datos fueron obtenidos por medio del *topic* publicado por el paquete EKF, el cual publica los valores de medición de distancia estimados por el robot.

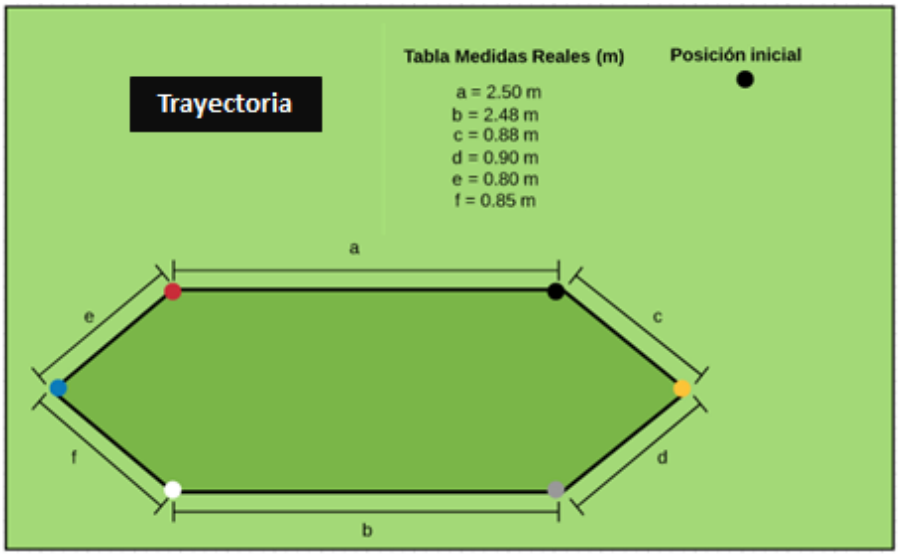

Figura 38. Plano de medidas Algoritmo EKF (Luz natural).

8.7.1.1 Análisis teórico pruebas experimentales algoritmo EKF

A continuación, en la "Tabla 6", se presenta la comparación de los valores reales de medición de la trayectoria recorrida y los datos estimados por el robot móvil, así mismo, se visualiza gráficamente el desfase presentado en esta medición (Figura 39).

Tabla 6. Medidas reales vs estimación de posición en luz natural (Algoritmo EKF).

Tabla Medidas Posición final (m)	***Tabla Medidas reales (m)***
Posición Inicial	*Posición Inicial*
2.5163	*2.50*
3.1336	*2.95*
2.6622	*2.79*
1.2776	*1.25*
0.9299	*0.88*

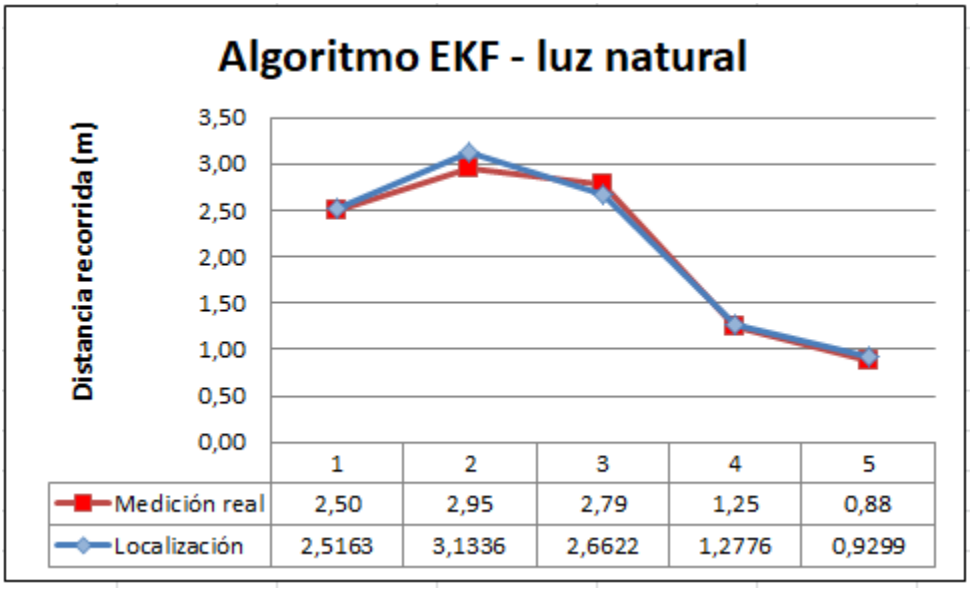

Figura 39. Grafica comparativa de medición real vs estimación de posición en luz natural (Algoritmo EKF).

Al obtener el desfase de medición real vs estimación de posición en el algoritmo EKF en luz natural, se procede a realizar el cálculo de la raíz del error cuadrático medio presentado en la “tabla 7”.

Tabla 7. Datos de error cuadrático medio en luz natural (Algoritmo EKF).

Valor Observado	Valor Predicho	Diferencia	(RECM)
2,5163	*2,5*	0,0163	**0,103498077**
3,1336	*2,95*	0,1836	
2,6622	*2,79*	-0,1278	
1,2776	*1,25*	0,0276	
0,9299	*0,88*	0,0499	

8.7.2 Filtro de Kalman luz artificial

A continuación, se visualizará en la “tabla 8” los datos de estimación de posición del *Turtlebot3 Burger* obtenidos por medio de la terminal de comandos a partir de las pruebas experimentales de medición implementadas por medio del algoritmo EKF en condiciones de luz artificial, para cada una de las trayectorias desde el punto inicial hasta el punto final.

Tabla 8. Datos de traslación Algoritmo EKF (luz natural).

Trayectoria recorrida	**Translación ‘X’**	**Translación ‘Y’**	**Translación ‘Z’**
(Punto inicial vs punto rojo)	2.51352000237	1.08716225624	0,0
(Punto inicial vs punto azul)	3.05566954613	1.69720184803	0,0
(Punto inicial vs punto blanco)	2.87699174881	1.74882626534	0,0
(Punto inicial vs punto gris)	1.12570667267	1.96073234081	0,0
(Punto inicial vs punto Amarillo)	0.872734129429	1.93386828899	0,0

En la (Figura 40) se encuentra el plano correspondiente a la “trayectoria 2”, donde se visualizan los datos de estimación de posición del robot desde su punto inicial (punto negro, entre los segmentos ‘a’ y ‘c’) hasta cada una de las posiciones finales (Puntos de colores), en la fase de luz artificial. Estos datos fueron obtenidos por medio del *topic* publicado por el paquete EKF, el cual publica los valores de medición de distancia estimados por el robot.

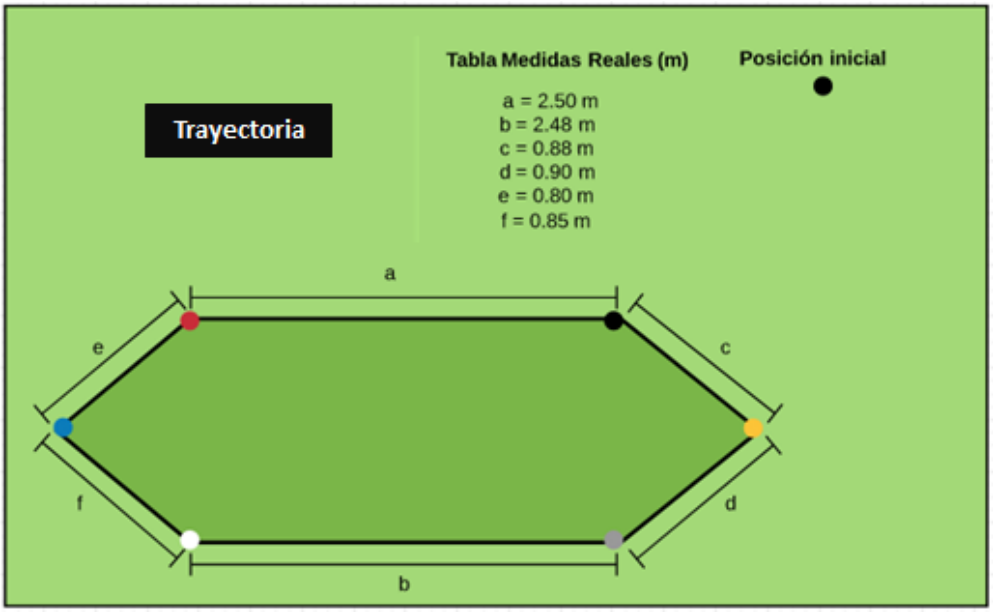

Figura 40. Plano de medidas (Algoritmo EKF - Luz artificial).

8.7.2.1 Análisis teórico pruebas experimentales algoritmo EKF

A continuación, en la "Tabla 9", se presenta la comparación de los valores de medición reales de la trayectoria recorrida y los valores estimados por el robot móvil, así mismo, se visualiza gráficamente el desfase presentado en esta medición (Figura 41).

Tabla 9. Medidas reales vs estimación de posición en luz artificial (Algoritmo EKF).

Tabla Medidas Posición final (m)	***Tabla Medidas reales (m)***
Posición Inicial	*Posición Inicial*
2.5135	*2.50*
3.0556	*2.95*
2.8769	*2.79*
1.1257	*1.2587*
0.8727	*0.88*

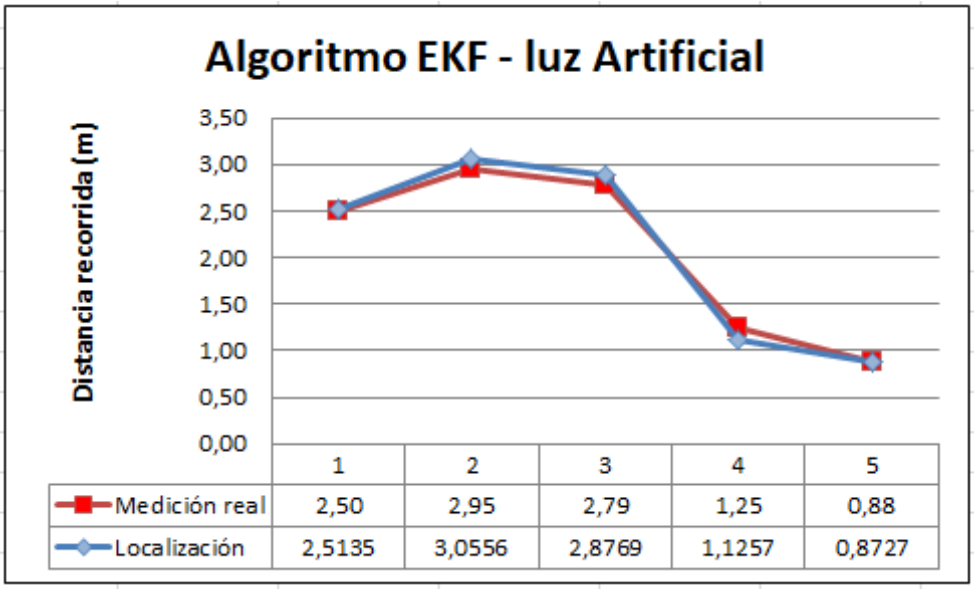

Figura 41. Grafica comparativa de medición real vs estimación de posición en luz artificial (Algoritmo EKF).

Al obtener el desfase de medición real vs estimación de posición en el algoritmo EKF en luz artificial, se procede a realizar el cálculo de la raíz del error cuadrático medio presentado en la “tabla 10”.

Tabla 10. Datos de error cuadrático medio en luz artificial (Algoritmo EKF).

Valor Observado	Valor Predicho	Diferencia	***RECM (Raíz del Error Cuadrático Medio)***
2,5135	*2,5*	0,0135	**0,082932503**
3,0556	*2,95*	0,1056	
2,8769	*2,79*	0,0869	
1,1257	*1,25*	-0,1243	
0,8727	*0,88*	-0,0073	

Al finalizar con la sección de experimentación y al obtener los datos obtenidos por el cálculo de la raíz del error cuadrático medio en las 4 secciones de prueba en los ítems 8.6.1.1, 8.6.2.1, 8.7.1.1, 8.7.2.1, se procede a realizar la elección del algoritmo más asertivo a partir de los resultados visualizados en la “tabla 11” y (Figuras 42, 43, 44) respectivamente.

Tabla 11. Valores de Error Cuadrático medio.

Algoritmo	Condiciones de iluminación	*Raíz del Error Cuadrático Medio*
Filtro de Kalman	Luz natural	0,103498077
Filtro de Partículas	Luz natural	0,062770521
Filtro de Kalman	Luz artificial	0,082932503
Filtro de Partículas	Luz artificial	0,0331225

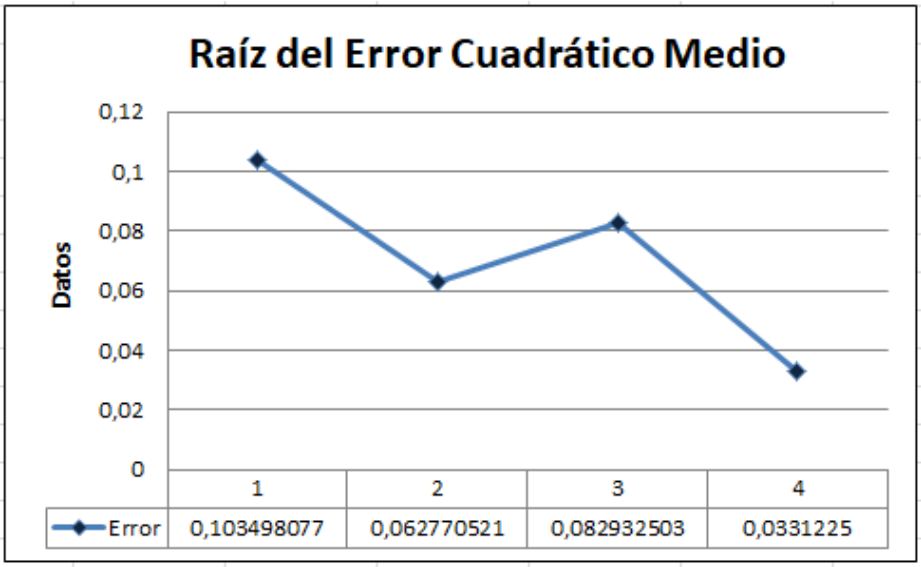

Figura 42. Comparación datos de la Raíz del error cuadrático medio.

A continuación, en las (Figuras 43, 44) se ilustran los comportamientos de los algoritmos Gmapping (Filtro de Partículas) y del algoritmo EKF respectivamente bajo los dos escenarios de prueba con fin de evaluar cuantitativamente la consistencia de la estimación de posición.

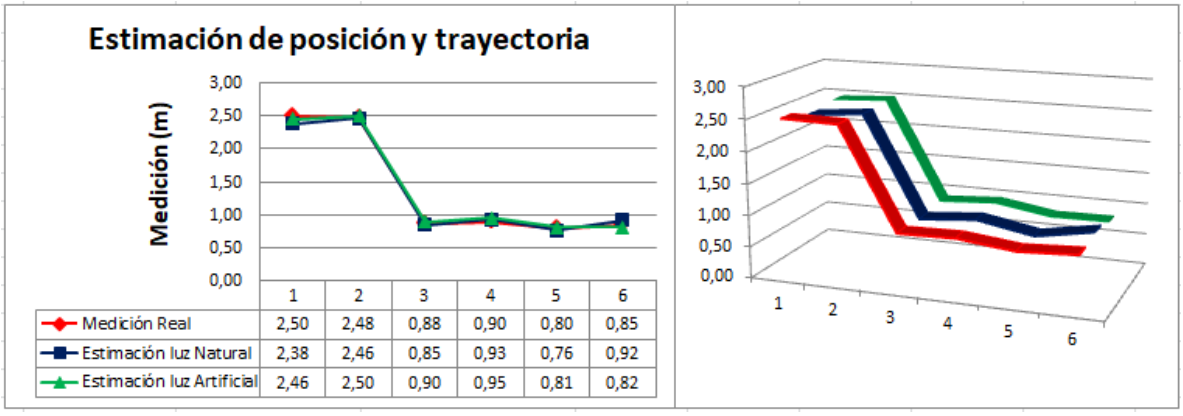

Figura 43. Ilustración Medición real vs estimación de posición (Algoritmo Gmapping).

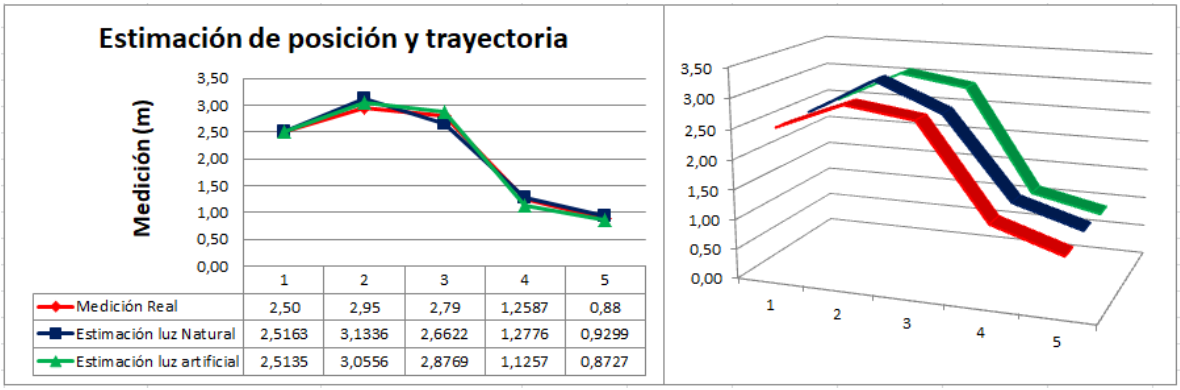

Figura 44. Ilustración Medición real vs estimación de posición (Algoritmo EKF).

A partir de las pruebas experimentales denotadas en las tablas 1, 3, 6, 9, se logra identificar que los parámetros de estimación de posición influyentes en la técnica SLAM son aquellos que se ven alterados en distintas condiciones de iluminación. En la (Figura 42) se obtiene una gráfica de los resultados obtenidos a partir del cálculo de la raíz del error cuadrático medio frente al comportamiento en los algoritmos Filtro de Kalman y Filtro de partículas, los datos expuestos en las (Figuras 42, 43) brindan información que sustenta mayor rendimiento en la implementación del algoritmo Gmapping (Filtro de Partículas) y menor rendimiento en la implementación del algoritmo EKF (Filtro de Kalman Extendido) a partir de los parámetros de estimación de posición del TurtleBot3 Burger ante variaciones de iluminación.

9. CONCLUSIONES

- Teniendo en cuenta los objetivos que se plantearon en este documento, en la implementación de los algoritmos Filtro de Kalman y Filtro de partículas, en un robot móvil diferencial en ambientes interiores a partir de un sensor Kinect se logra evidenciar en las tablas 1, 3, 6, 9 que los parámetros de estimación de posición son aquellos que se ven alterados ante variaciones de iluminación. Estos resultados concluyen mejor rendimiento en la implementación del algoritmo SLAM Gmapping (Filtro de Partículas) en ambientes interiores con iluminación artificial.

- En relación al análisis de los resultados obtenidos en las pruebas experimentales a partir del cálculo de la raíz del error cuadrático medio contemplado en las tablas 2, 4, 7, 10, se concluye que se obtuvo mayor rendimiento en la implementación del algoritmo Gmapping (Filtro de Partículas) y menor rendimiento en la implementación del algoritmo EKF (Filtro de Kalman Extendido) a partir de los parámetros de estimación de posición del TurtleBot3 Burger ante variaciones de iluminación.

- A partir de los resultados obtenidos en la implementación del algoritmo Gmapping (Filtro de Partículas) se logra evidenciar que no se presentan alteraciones ante variaciones de iluminación en los mapas obtenidos. El área de pixeles blancos siendo el área conocida y negros área desconocida presentan equivalentes resultados en la implementación tanto de luz natural visualizada en la Figura 32 como de luz artificial visualizada en la Figura 35.

- En este documento se logra implementar técnicas SLAM a partir de un sensor Microsoft Kinect, convirtiéndose en una buena elección como alternativa de bajo costo, proporción de datos 3D y obtención de imágenes de profundidad al alcanzar resultados favorables en aplicaciones de robótica avanzada de alto reconocimiento como lo son todas aquellas técnicas SLAM puestas en marcha dentro de un ambiente interior.

- A través del resultado obtenido gráficamente y visualizado en la Figura 42 de los datos calculados del error frente al comportamiento de los algoritmos Filtro de Kalman y Filtro de Partículas, se deduce que el Filtro de Kalman obtiene resultados de menor exactitud en condiciones de luz artificial frente a los resultados obtenidos en la implementación del algoritmo Filtro de Partículas en condiciones de luz natural, lo que representa al algoritmo Gmapping un algoritmo más competente ante variaciones de iluminación en aplicaciones de robótica móvil en ambientes interiores mediante un sensor de visión.

- A partir de un proceso complementario se logra convertir la nube de puntos (*PointCloud2*) obtenida por el sensor Microsoft Kinect en una señal de

escaneo láser, con el fin de obtener la imagen de profundidad en 2D para implementar el algoritmo Gmapping (Filtro de Partículas).

10. TRABAJO FUTURO

Como continuación de este proyecto de grado y cualquier otro tipo de investigación realizada, se tienen en cuenta las diferentes líneas como es el caso de aplicaciones con el *TurtleBot3 Burger* con diversas técnicas SLAM que quedan abiertas y en las que es posible continuar a partir de este proyecto, en las cuales se espera finalmente poder complementar en un futuro ya que están directamente relacionadas con el presente proyecto de grado, siendo esta el resultado de la realización de la misma.

A continuación, se presentan algunos trabajos futuros que se podrán implementar como resultado de esta investigación:

- Realizar la implementación de los algoritmos filtro de Kalman y filtro de Partículas en un robot móvil diferencial con un sensor 360º LIDAR, obteniendo como resultado mapeo y localización en 2D para aplicaciones de robótica móvil.

- Encontrar que variaciones se presentan en los resultados obtenidos a través de las aplicaciones SLAM (Localización y mapeo simultáneos) 2D y 3D de los algoritmos filtro de Kalman y filtro de Partículas ante variaciones de iluminación en un robot móvil diferencial mediante un sensor visual diferente al sensor Kinect implementado en este proyecto.

- Implementar los algoritmos filtro de Kalman y filtro de partículas en un robot aéreo, para SLAM (Localización y mapeo simultáneos), utilizando una versión de ROS distinta de ROS kinetic aplicada en este proyecto, mediante un sensor 3D.

- Aplicar la conversión de mapeo 2D a 3D para SLAM (Localización y mapeo simultáneos), con el fin de implementar el proceso inverso al utilizado en este proyecto, en el cual se realizó conversión de mapeo 3D a 2D.

11. IMPACTO SOCIAL

Se considera que la investigación presentada en este documento será una herramienta a la hora de implementar aplicaciones SLAM (Mapeo y Localización Simultanea) en entornos sociales. Esta técnica representa un avance considerable en el campo de la robótica y su aplicación en entornos netamente sociales, desde aplicaciones en hospitales y empresas hasta aplicaciones en hogares o misiones de búsqueda y rescate, situaciones en las que se presentan incontables variaciones en la iluminación, tal como las descritas en este documento.

Estas variaciones de iluminación intervienen en el correcto funcionamiento de los robots móviles en entornos sociales e interacción con personas [31], aun lo suficiente para impedir la comercialización de los mismos, por ello este estudio provee de una serie de experimentos y procesos enfocados en la exposición y el análisis de los factores influenciados por las variaciones de iluminación en las aplicaciones SLAM del filtro de Kalman y filtro de Partículas en un robot móvil diferencial a partir de un sensor visual.

12. REFERENCIAS BIBLIOGRÁFICAS

[1] D. L. Martínez Herrera, «Reconocimiento de Espacios con Optimización de Trayectorias Utilizando el Robot,» 2019. [En línea]. Available: https://repository.usta.edu.co/bitstream/handle/11634/18667/2019davidmartinez.pdf. [Último acceso: 2 Octubre 2019].

[2] J. Navarro García, « Definición ABC,» 05 2016. [En línea]. Available: https://www.definicionabc.com/ciencia/parametro.php. [Último acceso: 22 07 2020].

[3] H. Jo, S. Jo, E. Kim, C. Yoon y S. Jun, «3D FastSLAM algorithm with Kinect sensor,» de *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS)*, Kitakyushu, 2014.

[4] A. Burguera, Y. González y G. Oliver, «RANSAC Based Data Association for Underwater Visual SLAM,» de *ROBOT2013: First Iberian Robotics Conference*, 2014.

[5] H. S. Oh, M. Hahn y J. Kim, «Simultaneous Localization and Mapping for Mobile Robots in Dynamic Environments,» de *Information Science and Applications (ICISA) 2013 International Conference on*, 2013.

[6] X. Chen, «An Adaptive UKF-Based Particle Filter for Mobile Robot SLAM,» de *2009 International Joint Conference on Artificial Intelligence*, Hainan Island, 2009.

[7] J. Viñals Pons, «Localización y generación de mapas,» Valencia, 2012.

[8] K. Kamarudin, «Method to convert Kinect's 3D depth data to a 2D map for indoor SLAM,» de *2013 IEEE 9th International Colloquium on Signal Processing and its Applications*, Kuala Lumpur, 2013.

[9] T. Hachaj, M. R. Ogiela y K. Koptyra, «Effectiveness Comparison of Kinect and Kinect 2 for Recognition of Oyama Karate Techniques,» de *18th International Conference on Network-Based Information Systems*, Taipei, 2015.

[10] G. Bermudez, «Robots moviles. Teoría, aplicaciones y experiencias,» *Tecnura 10,* vol. 5, nº 10, pp. 6-17, 2002.

[11] A. Reina Gutiérrez, «El diseño de experiencias en robótica social asistencial para la tercera edad: Analisis de la pelicula ROBOT & FRANK (Proyecto de grado-Maestria en Filosofía-Universidad del Valle),» Cali, 2015.

[12] J. Kim y A. Kim, «Light condition invariant visual SLAM via entropy based image fusion,» de *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Jeju, South Korea, 2017.

[13] J. Viñals Pons, «Localización y generación de mapas del entorno (SLAM) de un robot por medio de una kinect,» 2012 Septiembre 2012. [En línea]. Available: https://riunet.upv.es/bitstream/handle/10251/17544/Memoria.pdf?sequence=1&isAllowed=y. [Último acceso: 03 Octubre 2019].

[14] J. VIÑALS PONS, «Localización y generación de mapas del entorno (SLAM) de un robot por medio de una Kinect (Proyecto de grado-Ingenieria Informatica Superior-Universitat Politècnica de València),» Valencia, 2012.

[15] L. Tae-jae, K. Chul-hong y C. Dong-il Dan, «A Monocular Vision Sensor-Based Efficient,» *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS,* vol. 66, nº 1, pp. 318-328, 2019.

[16] A. P. Sarmiento Andrade y D. A. . Ramírez Patiño, «Navegación autónoma de un quadrotor en entornos interiores mediante SLAM (Proyecto de grado-Ingeniería Electrónica-Universidad Javeriana),» Bogota D.C., 2014.

[17] P. . Monroy Vilcahuaman, «Desarrollo de algoritmo basado en inteligencia artificial y SLAM (Simultaneous localization and mapping) aplicado a un robot autónomo para modelar entornos (Proyecto de grado - Ingeniería Electrónica - ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA),» AREQUIPA, 2018.

[18] R. Munguía y A. Grau, «SLAM con mediciones angulares: método por triangulación estocástica,» *Ingeniería, Investigación y Tecnología,* vol. XIV, nº 2, pp. 257-274, 2013.

[19] T. Genevois y T. Zielińska, «A SIMPLE AND EFFICIENT IMPLEMENTATION OF EKF-BASED SLAM RELYING ON LASER SCANNER IN COMPLEX INDOOR ENVIRONMENT,» *Journal of Automation, Mobile Robotics & Intelligent Systems,* vol. 8, pp. 58-67, 2014.

[20] S.-Y. An, J.-G. Kang, L.-K. Lee y S.-Y. Oh, «Line Segment-Based Indoor Mapping with Salient Line Feature Extraction,» *ADVANCED ROBOTICS,* vol. 26, nº 5-6, pp. 437-460, 2011.

[21] K. Ho-Duck, S. Sang-Wook, J. In-hun y S. Kwee-Bo, «SLAM of mobile robot in the indoor environment with Digital Magnetic Compass and Ultrasonic Sensors,» de *International Conference on Control, Automation and Systems*, Seoul, 2007.

[22] G. Hendeby y G. Bleser, «Using optical flow as lightweight SLAM alternative,» de *8th IEEE International Symposium on Mixed and Augmented Reality*, Orlando, FL, USA, 2009.

[23] J. Motsch, S. Benammar y Y. Bergeon, «Interior mapping of a building: A real-life experiment with Microsoft Kinect for Windows v1 and RGBD-SLAM,» de *International Conference on Military Technologies (ICMT)*, Brno, Czech Republic, 2017.

[24] Y.-T. Wang, Y.-C. Feng y D.-Y. Hung, «Detection and tracking of moving objects in SLAM using vision sensors,» de *IEEE International Instrumentation and Measurement Technology Conference*, Binjiang, China, 2011.

[25] K. Watanabe, C. Pathiranage y K. Izumi, «T-S fuzzy model adopted SLAM algorithm with linear programming based data association for mobile robots,» de *IEEE International Symposium on Industrial Electronics*, Seoul, South Korea, 2009.

[26] B. Keun Kim, «Indoor localization and point cloud generation for building interior modeling,» de *IEEE RO-MAN*, Gyeongju, South Korea, 2013.

[27] R.-J. Yan, J. Wu, S.-J. Lim, J.-Y. Lee y C.-S. Han, «Natural Corners-based SLAM in Unknown Indoor Environment,» de *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Daejeon, Korea, 2012.

[28] R. Ozaki y Y. Kuroda, «6-DoF EKF SLAM con características planas globales en entornos artificiales,» de *Simposio internacional 2020 IEEE / SICE sobre integración de sistemas (SII)*, Honolulu, HI, EE. UU., 2020.

[29] M. S. Kashi, T. B. Sriram y R. Mohan, «An Approach to Labelled Indoor Mapping using SLAM and Object Detection,» de *2019 IEEE Region 10 Symposium (TENSYMP)*, Kolkata, India, 2019.

[30] G. F. Perez Paina, C. J. Paz, M. Baudino y L. Canali, «SLAM monocular basado en UKF para la localización de un robot móvil.,» de *VIII Jornadas Argentinas de Robótica (JAR)*, 2014.

[31] F. d. P. Gonzalez, J. G. Guarnizo y G. Benavides, «Emulation System for a Distribution Center Using Mobile Robot, Controlled by Artificial Vision and Fuzzy Logic,» *IEEE Latin America Transactions,* vol. 12, nº 4, pp. 557-563, 2014.

[32] T. Chong, X. Tang, C. Leng, M. Yogeswaran, O. Ng y Y. Chong, «Sensor Technologies and Simultaneous Localization and Mapping (SLAM),» *Robotics and Autonomous,* vol. 98, pp. 67-88, 2018.

[33] K. Granstrom, C. Lundquist, F. Gustafsson y U. Orguner, «Random Set Methods:Estimation of Multiple Extended Objects,» *IEEE Robotics & Automation Magazine,* vol. 21, nº 2, pp. 73-82, 2014.

[34] C. Stachniss, L. J. J y S. Thrun, «Simultaneous Localization and Mapping,» 2016. [En línea]. Available: https://link.springer.com/chapter/10.1007/978-3-319-32552-1_46#citeas.

[35] H. Durrant-Whyte y T. Bailey, «Simultaneous localization and mapping: part I,» *IEEE Robotics & Automation Magazine,* vol. 13, nº 9018822, pp. 99-110, 2006.

[36] L. Abolore Yekinni y A. Dan-Isa, «2019 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS),» de *Fuzzy Logic Control of Goal-Seeking 2-Wheel Differential Mobile Robot Using Unicycle Approach*, Selangor, Malaysia, Malaysia, 2019.

[37] J. &. M. Mistakes, «ROBOTS e-manual (TurtleBot3),» 2019 . [En línea]. Available: http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/. [Último acceso: 02 Octubre 2019].

[38] N. Yasuhiro, T. Sou, Y. Hidetoshi y T. Hideki, «ZytleBot: FPGA Integrated Development Platform for ROS Based Autonomous Mobile Robot,» de *2019 International Conference on Field-Programmable Technology (ICFPT)*, Tianjin, China, China, 9-13 Dec. 2019.

[39] R. Blog, «WordPress.com,» Romerobots Blog, [En línea]. Available: https://jjromeromarras.wordpress.com/2014/08/27/como-crear-un-paquete-en-ros/. [Último acceso: 07 03 2020].

[40] J. Stückler y S. Behnke , «Robust Real-Time Registration of RGB-D Images using Multi-Resolution Surfel Representations,» de *German Conference on Robotics (ROBOTIK)*, Munich, Germany, 2012.

[41] A. Fuentes Rios, «Localización y Modelado Simultáneos en ROS para la plataforma robótica MANFRED,» Octubre 2011. [En línea]. [Último acceso: 27 Febrero 2020].

[42] S. Thrun, D. Fox y W. Bugard, «Probabilistic Robotics,» 2000. [En línea]. Available: https://docs.ufpr.br/~danielsantos/ProbabilisticRobotics.pdf. [Último acceso: 1 Octubre 2019].

[43] E. Baklouti, N. Ben Amor y M. Jallouli, «Particle filter localization and real time obstacle avoidance control based on 3D perception for wheelchair mobile robot,» de *2016 IEEE 8th International Conference on Intelligent Systems (IS)*, Sofia, Bulgaria, 2016.

[44] Á. Solera Ramírez , «Documento de trabajo del Banco Central de Costa Rica,» Julio 2003. [En línea]. Available: https://activos.bccr.fi.cr/sitios/bccr/investigacioneseconomicas/DocMetodosCuantitativos/Filtro_de_Kalman.pdf. [Último acceso: 04 Octubre 2019].

[45] F. Endres, J. Hess, N. Engelhard, J. Sturm y W. Burgard, «An Evaluation of the RGB-D SLAM System,» de *Conf. sobre Robótica y Automatización (ICRA)*, 2012.

[46] Á. P. López, «Estudio y evaluación de los sistemas RGB-D SLAM,» Universidad Carlos III de Madrid, Leganés, 2017.

[47] E. R. Magsino, E. N. Young y J. R. Cornejo, «Application of a Fast Map Mergind Algorithm and RGB-D SLAM,» Yogyakarta, 2014.

[48] ROBOTIS, «ROBOTIS,» 2019. [En línea]. Available: http://emanual.robotis.com/docs/en/platform/turtlebot3/pc_setup/#install-ubuntu-on-remote-pc. [Último acceso: 15 10 2019].

[49] Openkinect, «Openkinect,» 04 08 2016. [En línea]. Available: https://openkinect.org/wiki/Getting_Started. [Último acceso: 10 12 2019].

[50] ROS.org, «ROS.org,» Open Source Robotic Foundation, 06 10 2014. [En línea]. Available: http://wiki.ros.org/freenect_launch. [Último acceso: 27 02 2020].

[51] G. Blog, «GeuS' Blog: Robotics, Computer Science and More,» 02 12 2010. [En línea]. Available: https://geus.wordpress.com/2010/12/02/robotics-simultaneous-localization-and-mapping-slam-con-ros/. [Último acceso: 15 01 2020].

[52] I. Bambino, «Una introduccion a los Robots Moviles,» 2008. [En línea]. Available: http://www.aadeca.org/pdf/CP_monografias/monografia_robot_movil.pdf. [Último acceso: 19 Octubre 2019].

[53] ROS.org, «ROS.org,» Open Source Robotic Foundation, 11 04 2019. [En línea]. Available: http://wiki.ros.org/rtabmap_ros/Tutorials/RemoteMapping. [Último acceso: 02 01 2020].

[54] C. Moreno Pulido, «Diseño de aplicación basada en Kinect,» de *Trabajo Fin de Grado (Diseño de aplicación basada en Kinect)*, Madrid, Universidad Carlos III de Madrid, 2014, pp. 45-46.