Mapeo de espacio universitario con Robot Humanoide Pepper

Trabajo de proyecto de grado presentado al departamento de Ingeniería de Sistemas y Computación

por

Vilma Marcela Tirado Gómez

Profesores asesores:

Mario Fernando De la Rosa Rosero

Y

Carolina Higuera Arias

Ingeniería de Sistemas y Computación

Universidad de los Andes

Bogotá, Colombia

julio de 2021

Tabla de Contenido

TABLA DE FIGURAS

RESUMEN

En la actualidad uno de los campos más estudiados en la robótica es la navegación, en particular la capacidad de un robot de generar un mapa de un ambiente dado y localizarse en él. Este proyecto busca entender el funcionamiento de la librería RTAB-Map (Mapeo basado en apariencia en tiempo real) de ROS, la cual usa un enfoque de SLAM basado en grafos, con el propósito de realizar el mapeo de una escena estática usando el robot humanoide Pepper. Con este propósito en mente se estudiaron y documentaron algunos de los parámetros de esta librería y se hicieron diferentes pruebas, tanto en un ambiente controlado como en un espacio libre, para determinar cómo infiere el valor de dichos parámetros en los mapas generados por esta librería.

ABSTRACT

Nowadays one of the most studied fields in robotics is navigation, in particular, the capacity of a robot to generate the map of a given environment and locate itself in it. This project aims to understand the behavior of ROS's RTAB-Map (Real-Time Appearance-Based Mapping) library, which uses a Stereo and Lidar Graph-Based SLAM approach, to map a static scene with the humanoid robot Pepper. With this purpose in mind, some of the parameters in this library were studied be documented and several tests were made in two different environments, one controlled and one in free space, to determine the inference of the parameters values in the maps generated with this library.

CAPITULO I. INTRODUCCIÓN

En la robótica existen varios retos en el tema de la navegación, uno de los más importantes es la detección de obstáculos. Existen varios métodos para enfrentar este problema como por ejemplo realizar un mapa del espacio por el que va a transitar el robot y realizar la planeación de ruta teniendo en cuenta los obstáculos mostrados en este espacio. Esto si bien resuelve una parte del problema sigue sin solucionar una parte importante, la detección de objetos dinámicos. Para resolver esto es necesario que el robot tenga la posibilidad de mapear su entorno en tiempo detectando los obstáculos en movimiento.

En la actualidad una de las plataformas más utilizadas para la interacción con robots es ROS. Una de las ventajas de esta plataforma es que cuenta con muchas librerías para afrontar diversos problemas, entre ellos el mapeo en tiempo real. La librería RTAB-Map de ROS permite hacer un mapeo real del entorno del robot mediante cámaras y sensores laser. Para que esta librería funcione correctamente es necesario configurar de forma adecuada los parámetros que utiliza de tal forma que se tengan en cuenta los factores del entorno en el que se está moviendo el robot.

Con base en lo anterior este proyecto de investigación pretende estudiar y evaluar como la modificación de los valores que toman estos parámetros infiere en el mapa generado por el robot. Para lograr esto se proponen 4 etapas diferentes: la primera consiste en explorar los diferentes parámetros de la librería agruparlos y definir cuáles son los más importantes para la tarea de mapeo. Una vez identificados dichos parámetros es necesario crear una documentación para explicar el funcionamiento de estos. La segunda etapa del proyecto consiste en ver el funcionamiento de los parámetros en un ambiente de simulación, donde las condiciones son ideales y los mapas resultantes no se ven afectados por factores externos, como la calidad de las cámaras y los laser del robot o las condiciones variantes del ambiente. La tercera etapa del proyecto consiste en crear diversos mapas modificando los parámetros de cada uno de ellos en un ambiente controlado donde las condiciones del ambiente no varían a lo largo del recorrido. La última etapa del proyecto consiste en constatar el funcionamiento de dichos parámetros en un segundo ambiente el cual en este caso es un pasillo del sótano 1 del edificio de ingeniería de la Universidad de los Andes.

La solución propuesta resuelve una parte importante del problema de navegación en robótica ya que teniendo un mapa bien definido del entorno el robot puede realizar diferentes tareas de navegación como, por ejemplo, tareas de planeación de ruta. Debido a que las condiciones de cada ambiente al que se enfrenta el robot pueden variar, el tener una documentación de los parámetros sirve como guía para saber que parámetros se deben variar en cada entorno. Para evaluar los mapas resultantes se tendrá en cuenta la exactitud de su forma en comparación al ambiente mapeado y la definición de los bordes de este.

En términos de diseño e implementación este proyecto parte de una solución previamente diseñada que es RTAB-Map creada por M. Labbé y F. Michaud [2] la cual tiene una librería adaptada para ROS. Para estudiar esta librería se va a hacer uso del robot humanoide Pepper que cuenta con 2 cámaras 2D y una cámara 3D además de 3 sensores laser que son el hardware necesario para utilizar esta librería.

A partir del desarrollo de las etapas descritas previamente este proyecto obtuvo los siguientes resultados, se generó una descripción de 13 parámetros los cuales se dividen en 4 categorías diferentes: Odometria, la cual se divide en odometria visual y odometria lidar, creación del mapa en memoria y loop closure. Estos parámetros fueron evaluados en simulaciones donde se obtuvieron hallazgos significativos de 5 de ellos. Adicionalmente haciendo uso del robot humanoide Pepper se evaluaron los parámetros en los dos ambientes mencionados anteriormente con diferentes pruebas que permitieron ver el funcionamiento estos en la vida real.

En el escenario controlado se obtuvieron diversos mapas para cuatro de los parámetros donde cada uno de ellos es el resultado de la modificación del valor de algún parámetro. En el escenario no controlado fue posible darse cuenta de que, debido a las características del ambiente, no solo es importante el valor de los parámetros en el mapa, sino que también es importante la técnica de mapeo utilizada para recorrer el entorno. Es por ello por lo que para este ambiente no solo se generaron mapas para 4 de los parámetros, sino que también se exploró el efecto de diferentes técnicas de mapeo en el resultado final.

Siguiendo la tabla de contenido, la lectura de este documento puede iniciar en este capítulo de introducción (capítulo I) donde se encuentra la visión global de todo el trabajo desarrollado en este proyecto. En el capítulo II se pueden encontrar los objetivos del proyecto y el marco conceptual en el que se formaliza el desarrollo de las pruebas implementadas para llegar a la solución. En el capítulo III el lector podrá encontrar la delimitación del problema en donde se definen y agrupan los parámetros de la librería que se van a evaluar a lo largo del proyecto y se genera una documentación apropiada para entender el funcionamiento estos. En el capítulo IV el lector podrá encontrar las pruebas hechas en el ambiente de simulación. En los capítulos V y VI se encuentran las diferentes pruebas realizadas haciendo uso del robot humanoide Pepper con sus respectivos resultados. Finalmente, el capítulo VII presenta las conclusiones de este trabajo.

CAPITULO II. DESCRIPCIÓN GENERAL

2.1 OBJETIVOS DEL PROYECTO

2.1.1 Objetivo general

Mapear el interior de un espacio universitario, en particular los laboratorios del primer sótano de la universidad de los Andes, con el robot humanoide Pepper. La(s) estrategia(s) propuesta(s) debe(n) probarse con robot humanoide Pepper en una ambiente simulación y un ambiente experimental usando la plataforma ROS, enfocándose en el uso de la librería RTAB-Map.

2.1.2 Objetivos específicos

1. Estudiar a profundidad el funcionamiento de la librería RTAB-Map apoyándose en simulaciones con la herramienta Gazebo, probar y documentar las funciones de RTAB-Map relacionadas con el mapeo de una escena.

2. Identificar el comportamiento de los parámetros de RTAB-Map para el mapeo de una escena compuesta usando un robot humanoide Pepper.

3. Validar el funcionamiento de RTAB-Map mediante diferentes pruebas en dos escenarios experimentales: un ambiente controlado y un pasillo del sótano 1 del edificio de ingeniería de la Universidad de los Andes.

2.2 MARCO CONCEPTUAL

Para poder entender el diseño y la implementación de la solución propuesta es necesario definir un marco conceptual y tener en cuenta la investigación de trabajos previos a este, esto con el fin de definir los conocimientos base que orientan el desarrollo de este trabajo y permiten interpretar los resultados del mismo.

Lidar

Detección de luz y rango. Es un método para determinar rangos (distancia variable) apuntando a un objeto con un láser y midiendo el tiempo que se tarda la luz reflejada en regresar. Se usa para crear imágenes digitales en 3D de un terreno o área dada. Este método puede usar luz ultravioleta o visible para generar imágenes 3D una de sus grandes ventajas es que puede apuntar a objetos de un amplio rango de materiales. [9]

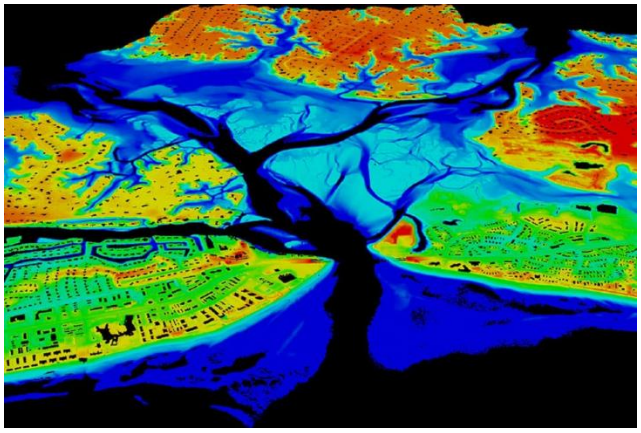

Figura 1 Detección de distancia (LiDAR) para una imagen [9]

SLAM

Localización simultanea y mapeo. Este es un término colectivo que reúne auto localización y mapeo de un entorno desconocido haciendo uso de varios sensores. Existen varios tipos de SLAM algunos de ellos son

- SLAM visual (Vslam): Calcula la posición y la orientación del robot con respecto a lo que lo rodea mientras mapea el ambiente usando las entradas de una cámara RGBD. Este tipo de SLAM encuentra puntos de interés para triangular la posición 3D de la cámara. Debido a la gran cantidad de información que proveen las cámaras el Vslam permite detectar posiciones previamente medidas.

- SLAM lidar: El SLAM lidar usa un láser como sensor para generar un mapa (2D o 3D) del ambiente. Una de las ventajas de usar SLAM lidar es que provee mediciones de alta precisión respecto a la distancia entre los objetos en un entorno [10]

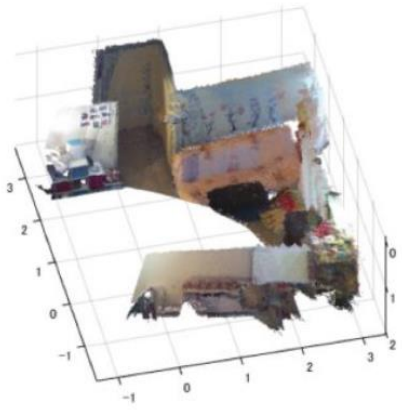

Figura 2 SLAM visual de un entorno [10]

Frame

Un frame es una de las muchas imágenes que hacen parte de una imagen en movimiento (video). En el contexto de este proyecto un frame es una imagen dentro de la secuencia de imágenes capturadas que usa el robot para construir un mapa. [11]

Figura 3 Frames de un video [12]

KeyFrame

Es un frame en el que ocurre un cambio a lo largo del tiempo. Un key frame indica que un nuevo objeto apareció en el frame. Es una toma que define el principio o el final de una transición. [13]

Feature

Es un punto con información del contenido de la imagen. Un feature puede ser un punto, un borde, o un objeto en una imagen que permite identificar una imagen de forma repetitiva obtenida desde diferentes ángulos y en distintas condiciones de iluminación. Son aquellas características que permiten reconocer un objeto en diferentes imágenes. [14]

Figura 4 Detección de features en una imagen [15]

Frame to Map (F2M)

Es una aproximación de odometria visual que registra cada nuevo frame capturado contra un mapa local de features creado a partir de los key frames anteriores. [2]

Frame to Frame (F2F)

Es una aproximación de odometria visual que registra cada nuevo frame capturado contra el key frame inmediatamente anterior. [2]

Loop closure

Es una aproximación usada para determinar si el lugar en el que se encuentra el robot (la imagen que registra en la cámara) corresponde a un lugar nuevo o a un lugar que ya ha sido visitado antes. [2]

2.3 TRABAJOS PREVIOS

A continuación, se muestra la investigación del estado del arte realizada con un breve de resumen de porque cada una de estas referencias aporta a esta investigación.

Titulo del proyecto: 3D Map Generation for Decommissioning Work (2020) [1]

Descripción: El propósito de esta investigación es crear un mapa 3D utilizando las cámaras del robot humanoide Pepper mediante el sistema operativo implementando y ejecutando SLAM (Localización y mapeo simultaneo). Para implementar SLAM se hace uso de la librería de ROS RTAB-Map, de forma que se pueda obtener la postura y el estatus del robot mediante la información que brindan tanto la cámara de profundidad como la cámara RGB.

Lo primero que se hace es explicar el funcionamiento de RTAB-Map. Usando tanto los sensores de profundidad como la cámara de profundidad de Pepper se detectan los puntos característicos del objeto que se tiene en frente, la rotación y la traslación en la posición de la cámara son calculadas a medida que se necesita teniendo en cuenta las coordenadas tridimensionales de los puntos característicos captados anteriormente. A partir de la fijación de puntos característicos, obtenidos a medida que el robot va rotando, se genera la información del mapa. Esta no cambia, sin importar el ángulo de la cámara.

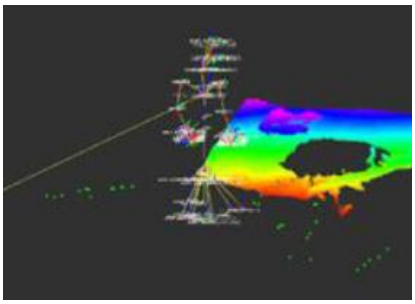

Figura 5 Mapa 3D generado con RTAB-Map

Una vez explicado el funcionamiento del algoritmo se hace una breve explicación del procedimiento utilizado para probar la librería donde uno de los mayores retos fue lograr el correcto movimiento del robot. El resultado de la investigación no es satisfactorio ya que no es posible generar un mapa 3D puesto que se presentan conflictos en el procesamiento de las imágenes de RTAB-Map y la operación manual de Pepper por lo tanto solo fue posible obtener una parte del mapa. Los resultados de esta rotación se pueden ver en la Figura 5 Mapa 3D generado con RTAB-Map. Los errores durante esta investigación se atribuyen a que no fue posible calcular puntos característicos en las imágenes debido a la distorsión presente en la imagen generada por el robot y la información sobre su profundidad. Como alternativa para resolver este problema se propone remover la distorsión en las imágenes ya sea recalibrando las cámaras de Pepper o quitando la tapa que se puede encontrar en la parte superior de la cámara de Pepper.

Titulo: RTAB-MAP as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation (2019) [2]

Descripción: En este capítulo de este Journal se da una explicación detallada del funcionamiento de la librería RTAB-Map y se explica cómo se puede usar esta para el mapeo y la localización simultánea. Para explicar el funcionamiento de esta librería se hace uso de un diagrama de bloques (Figura 6) en donde se explica la interacción entre los diferentes tópicos de los que hace uso la librería. Se explica que tópicos se llaman entre ellos, cuáles de ellos tienen el rol de publicador y cuales el de suscriptor. Adicionalmente se habla de cuáles son los parámetros de entrada y cuáles son los parámetros de salida del algoritmo. Se resalta que una de las

ventajas de la librería RTAB-Map es que la odometria que se usa para hacer el mapa puede venir de distintas fuentes ya sea una imagen (visual odometry), un láser (lidar odometry) o puede ser la odometria que viene de las llantas del robot (Wheel odometry)

De igual forma se habla de la importancia de la calibración de los sensores utilizados por el robot (ya sean cámaras o láser) para que el número de datos tomados por segundo concuerde con la velocidad a la que se mueve el robot de forma tal, que

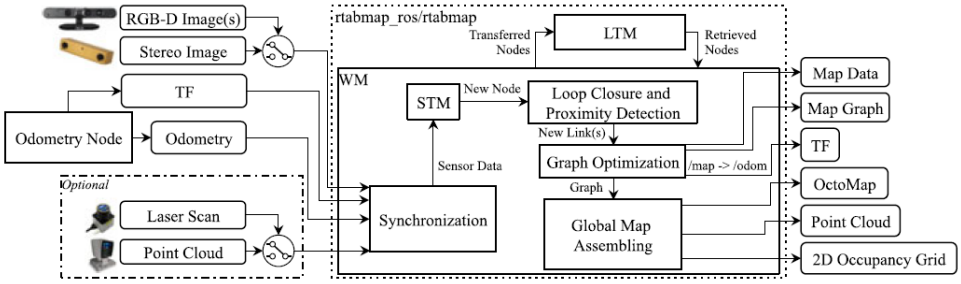

Figura 6 Diagrama de bloques del nodo RTAB-Map de ROS

se pueda generar un buen mapa. A su vez se explican algunos de los métodos utilizados por el algoritmo para optimizar la cantidad de memoria que utiliza para guardar la información del mapeo.

Titulo: RGB-D Odometry and SLAM (2019)[3]

Descripción: Este capítulo explica el funcionamiento de la odometría RGB y el SLAM para el mapeo 3D, el cual es el enfoque que utiliza la librería RTAB-Map. Se da una visión general del algoritmo y como funciona, la cual se ilustra en la Figura 7. Para explicar el funcionamiento del algoritmo este se divide en 3 grandes fases, el “rastreo” de la cámara (camera tracking), el mapeo de la escena y el “cierre del ciclo” (loop closing). En la primera fase se hace una explicación de la matemática utilizada para determinar los 6 grados de movimiento de una cámara RGB-D los cuales son los métodos basados en la alineación fotométrica y los métodos basados en la alineación geométrica. Ambos métodos son buenos, sin embargo, se resalta que los métodos fotométricos (utilizados para el SLAM) tienen una mayor precisión tanto para la odometria como para el mapeo.

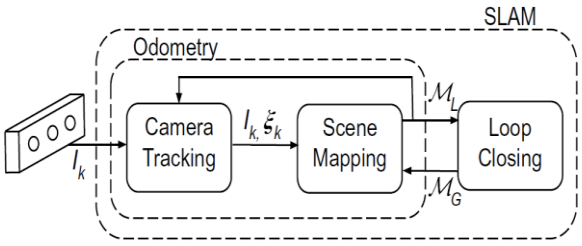

Figura 7 Explicación general del funcionamiento del algoritmo

En cuanto al mapeo de escena (scene maping) se explican los dos métodos básicos mapas basados en puntos y mapas volumétricos. Se explica nuevamente que dependiendo de las necesidades se puede escoger uno de los dos pero que, por lo general, se hace uso del mapeo de puntos cuando se tiene una cámara RGB.

Finalmente se explica el loop closing, que corrige los desvíos creados durante toda la exploración, tratando de mantener una representación global consistente de todo el entorno. Para el loop detection, el cual hace parte del loop closing, se hace uso del método conocido como bolsa de palabras (bag of Words) cuya implementación es bastante compatible con SLAM.

CAPITULO III. DISEÑO E IMPLEMENTACIÓN

La librería RTAB-Map (Mapeo basado en apariencia de tiempo real) es un enfoque de SLAM basado en RGB-D (camaras RGB y de profundidad) y un sensor lidar que usa un detector de loop closure incremental basado en apariencia. El detector loop closure usa un enfoque del algoritmo “bag-of-words” para determinar si una nueva imagen viene de un lugar que ya ha sido visitado antes o es de un lugar nuevo. Cuando se acepta una hipótesis de loop closure se añade una nueva restricción al mapa actual, después se usa un optimizador para minimizar errores en el mapa. Adicionalmente RTAB-Map cuenta con métodos de manejo de memoria para limitar el número de lugares que se utilizan para el loop closure y la optimización del mapa, de forma tal que para grandes ambientes siempre se respeten las restricciones que se ven en tiempo real. [4]

3.1 DEFINICIÓN DEL FUNCIONAMIENTO DE LOS PARÁMETROS

Debido a que la librería de RTAB-Map tiene una extensa cantidad de parámetros para efectos de este proyecto se tendrán en cuenta aquellos que se consideran fundamentales para obtener un mapa de buena calidad. Con esto en mente los parámetros que se van a estudiar se dividieron en 4 categorías: odometria visual, odometria lidar, memoria a corto plazo y loop closure.

3.1.1 Odometria Visual

Detección de features

Vis/MaxFeatures Cada vez que se captura una imagen se hace uso del algoritmo GFTT (GoodFeaturesToTrack) para detectar el número de features en una imagen. Este parámetro determina el máximo número de features que se permiten por cada imagen-

Feature Matching

A medida que se van tomando las imágenes se extraen sus features y se comparan con las de la imagen anterior mediante el algoritmo NNDR (Neares Neighbor Distance Ratio) para ver que tanto se parecen entre sí. **Vis/corNNDR** define la distancia que se usa dentro del algoritmo para determinar si dos features son o no parecidas.

Predicción de movimiento (motion prediction)

Para predecir en que parte del mapa local van las features de la imagen actual se usa un modelo de movimiento que tiene en cuenta la transformación de la imagen anterior. Esto limita la ventana, el rango de imágenes, con las cuales se puede

comparar la imagen actual para obtener correspondencias. En particular en ambientes dinámicos. **Vis/CorGuessWinSize** determina el número de imágenes que se usan para buscar las correspondencias.

Estimación de movimiento

Una vez se tienen las correspondencias del marco actual se usa el algoritmo RANSAC para transformar la imagen actual, con sus key frames en una parte del mapa local. Para aceptar la transformación es necesario tener un mínimo de inliners **Vis/MinInliers** dicta este umbral.

Actualización del feature map

Odom/KeyFrameThr es el parámetro que dicta el umbral para aceptar una transformación en el mapa. Cuando el número de Inliers capturados en la estimación de movimiento está por debajo de este umbral se actualiza el mapa.

3.1.2 Odometria lidar

Predicción de movimiento

En muchos casos la odometria lidar no es lo suficientemente fiable y es necesario rescatar la posición del robot de una fuente de odometria externa. **Icp/PointToPlaneMinComplexity** es un umbral que dicta si se tiene o no en cuenta la odometria lidar si la complejidad de la nube de puntos es menor que este umbral se desprecia la odometria lidar.

Actualización del mapa

Para actualizar el mapa es necesario tener en cuenta que aproximación de odometria lidar se está utilizando.

S2S

Si el rango de correspondencias está por debajo del umbral que da el parámetro **Odom/ScanKeyFrameThr** el frame actual se vuelve un key frame

S2M

Si se usa esta aproximación se hace un paso extra antes de integrar el punto nuevo al mapa de la nube de puntos. Al mapa actual es restado del nuevo punto (usando el radio que dicta el parámetro **OdomF2M/ScanSubstractRadius).** Los puntos que quedan después de la resta son los que se adicionan a la nube de puntos.

Debido a que este algoritmo busca optimizar el uso de memoria cuando el mapa de la nube de puntos sobrepasa el umbral dado por **OdomF2M/ScanMaxSize** los puntos más viejos del mapa son removidos.

3.1.3 Memoria de corto plazo (STM)

Cuando se crea un nodo nuevo en la memoria de corto plazo se crea una matriz de ocupación local teniendo en cuenta la fuente de odometria que se esté utilizando (profundidad de la imagen, scan laser, nube de puntos). El robot tiene una matriz de ocupación local que contiene la información sobre las celdas que están vacías las que están ocupadas y las que tienen obstáculos. El tamaño de estas celdas viene dado por **Grid/CellSize**.

Dependiendo de los parámetros **Grid/FromDepth** y **Grid/3D** (los cuales son de carácter booleano) **y** los tópicos a los que este suscrito el robot el mapa de

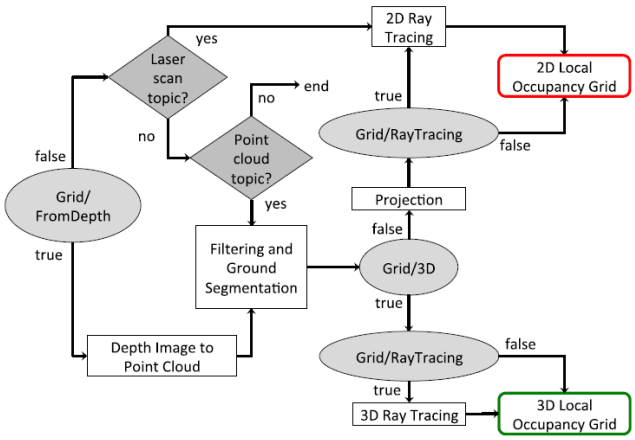

Figura 8 Relación entre los valores de los parámetros y los tópicos suscritos [2]

ocupación se genera en 2D o en 3D. La relación entre el valor de los booleanos y los tópicos suscritos se puede ver en la Figura 8.

La memoria de corto plazo STM puede ser vista como un buffer de tamaño **Mem/STMSize** que contiene los nodos más nuevos antes de pasarlos a la memoria en uso (WM)

3.1.4 Loop closure

Para ver la correspondencia entre los nodos que se encuentran en el STM y los que están en el WM se usa un filtro de bayes que estima las hipótesis de loop closure. El filtro determina si el nodo actual es de un lugar visitado previamente o si es de un nuevo lugar. Cuando la hipótesis de loop closure alcanza el umbral dado por **RTAB-Map/LoopThr** se acepta el valor calculado y se le hace la transformación correspondiente.

Kp/MaxFeatures: Para poder generar el loop closure no se necesitan todas las features dadas por la odometria visual, por lo tanto, para aumentar la eficiencia computacional, a la hora de generar el loop closure se usa un sub-set que tiene el número de features que dicta este parámetro. Es importante resaltar que para este subset se escogen las features que tienen una mejor respuesta al ser cuantificadas para usar el algoritmo de loop closure “bag-of-words”.

3.2 DISEÑO DE LA SOLUCIÓN

A partir del análisis realizado previamente y teniendo en cuenta lo visto en la sección 2.3 se propone el diagrama de la Figura 9 como diseño de la solución.

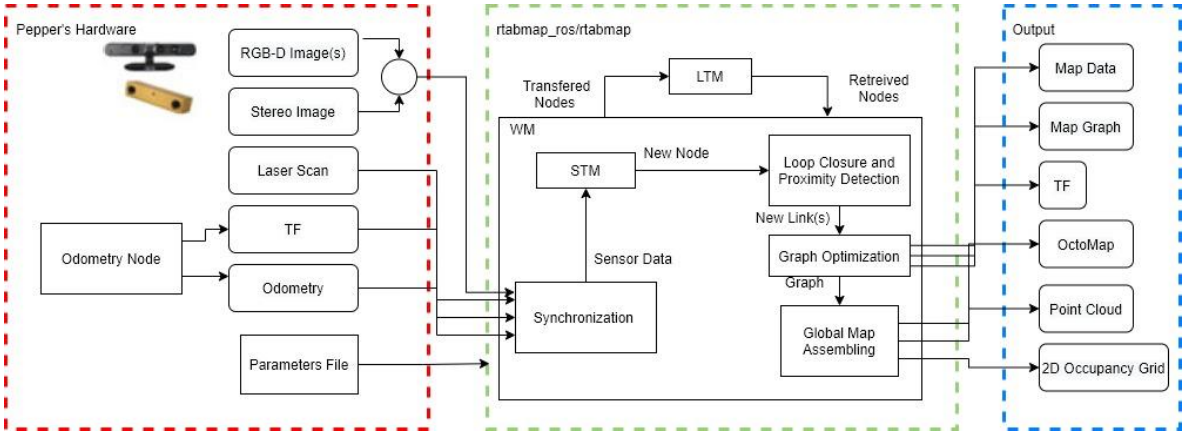

Figura 9 Diseño propuesto para la solución

El diagrama de la Figura 9 muestra la interacción entre el robot humanoide Pepper y la librería RTAB-MAP. En Rojo los componentes de hardware que hacen parte de Pepper. En Verde los módulos del paquete de mapeo de RTAB-Map de ROS. En Azul los diferentes resultados obtenidos del mapeo de una escena. Los resultados son los siguientes

Map data: Es la información del mapa que contiene el ultimo nodo añadido con la información obtenida de los sensores y el grafo.

Map Graph: El grafo del mapa sin información.

TF: La corrección de odometria publicada en el marco de referencia.

Octomap: La matriz de ocupación 3D del mapa.

Point cloud: La nube de puntos.

2D Ocupancy grid: La matriz de ocupación 2D del mapa.

CAPITULO IV SIMULACIONES EN GAZEBO

Partiendo de la documentación de los parámetros mostrada en el capitulo III es posible empezar a hacer simulaciones variando el valor que toma cada parámetro. El propósito de esto es generar pruebas mediante las cuales se pueda corroborar que el funcionamiento de los parámetros es acorde a lo documentado, de igual forma se busca entender que valores puntuales dan mejores resultados en el ambiente de simulación. Para estas simulaciones se va a hacer uso de la herramienta de simulación Gazebo la cual ya tiene las librerías necesarias para integrarse con ROS y permite crear distintos ambientes con una gran variedad de objetos. Una de las ventajas de esta herramienta es que cuenta con el modelo del robot turtlebot 2 que tiene una cámara RGB y una cámara de profundidad que pueden ser utilizadas como sensores para ejecutar la librería RTAB-Map.

4.1 PRIMER ENTORNO DE SIMULACIÓN

4.1.1 Ambiente de simulación

- **Sistema operativo:** Ubuntu 16.04.7 (Xenial Xerus)
- **ROS:** ROS kinetic
- **Versión de Gazebo:** 7
- **Robot:** Turtlebot 2
    - Dimensiones: 354 x 354 x 420 mm
    - Velocidad máxima: 0.65 m/s
    - Peso: 6.3 kg

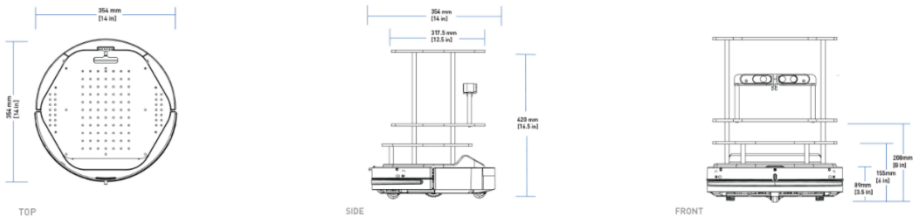

Figura 10 Dimensiones del turtlebot [6]

- **Camara:** Turtlebot simula el funcionamiento de una cámara Microsoft Kinect v1 la cual cuenta con las siguientes especificaciones

- Campo de visión: 43° vertical y 57° de visión en el campo horizontal
- Rango de movimiento vertical: $\pm 27°$
- Frames por segundo: 30 fps
- Distancia percibida por el sensor en simulación

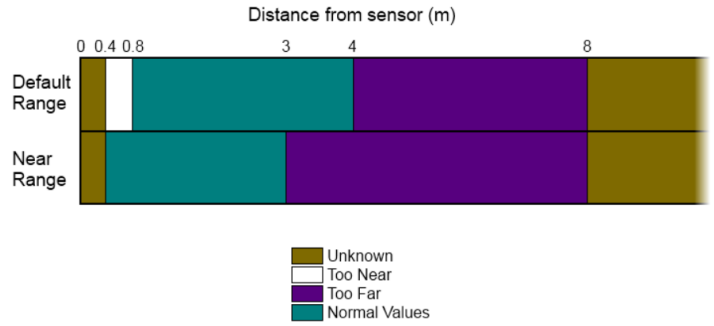

Figura 11 Distancia percibida por el Kinect [5]

- **Ambiente de simulación:** Para crear las simulaciones se usó el entorno que viene por defecto en gazebo. Como se puede ver en la Figura 12 este

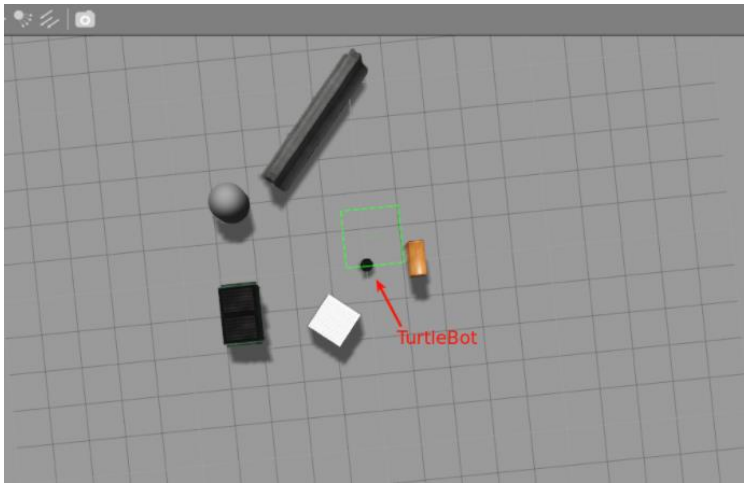

Figura 12 Ambiente de simulación usado en gazebo [17]

ambiente no tiene ningún tipo de pared y consta de 5 objetos diferentes cada uno de ellos tiene una forma y una dimensión diferente, de esta forma se pueden identificar las variaciones que dan los parámetros para objetos con diferentes características

4.1.2 Resultados obtenidos en primer entorno

Odometria visual

Lo primero que se hizo para entender el funcionamiento de la odometria visual fue comparar las aproximaciones de F2M y F2F

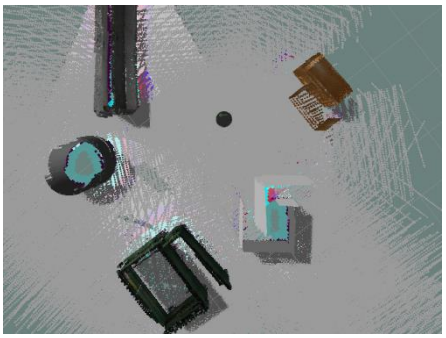

Figura 13 Mapa obtenido usando la aproximación de F2F

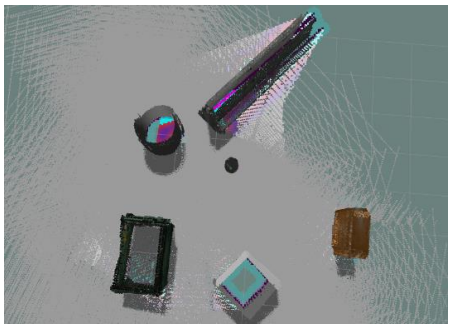

Figura 14 Mapa obtenido usando la aproximación de F2M

Comparando los resultados obtenidos se puede ver que la aproximación de F2M es mejor en términos de definición ya que los bordes de los objetos están mejor definidos. En el caso de F2F los objetos presentan errores de cálculo ya que como se puede ver en la Figura 13 paredes de los objetos no están claramente definidas, ya sea porque están corridas o porque se pintaron 2 veces en la figura.

corNNDR

Como se mencionó en el capítulo III este parámetro define la distancia que se usa dentro del algoritmo para determinar si dos features son o no parecidas. Como se puede ver en la Figura 15 y en la Figura 16 al cambiar los valores de estos parámetros no se notó ninguna diferencia al modificar el valor del parámetro.

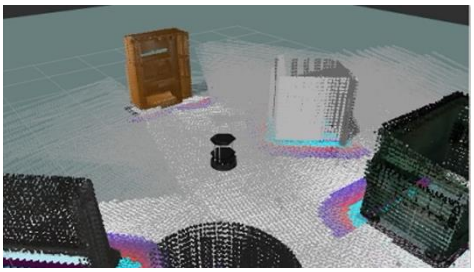

Figura 15 Resultado de usar una ratio de 5

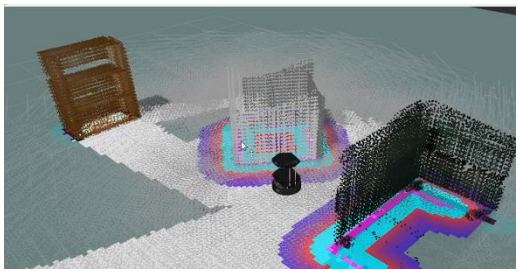

Figura 16 Resultado de usar una ratio de 20

MinInliers

El siguiente parámetro que se analizó fue el de los inliers los resultados obtenidos fueron los siguientes.

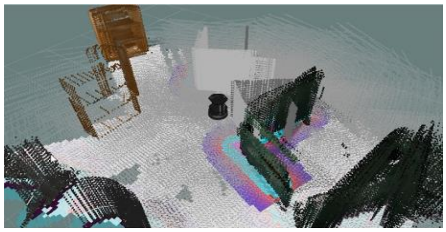

Figura 17 Mapeo de la escena completa usando un mínimo de 1 inlier

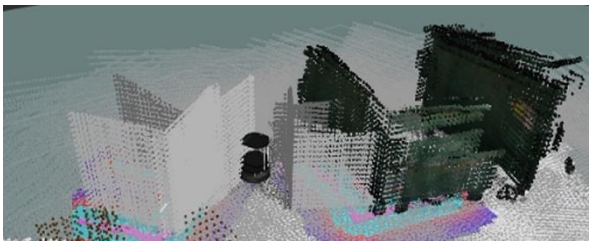

Figura 18: Mapeo obtenido usando un mínimo de 1 Inlier

Al disminuir el número de inliers al mínimo (1 inlier) posible los bordes de las imágenes dejan de estar delineados de forma correcta por lo que salen capas de cada objeto mucho más adelante de lo que estos realmente están (Figura 17 y Figura 18). A medida que se aumentan estos inliers la calidad de las imágenes mejora esto se puede notar comparando con el resultado obtenido al usar 15 inliers como se ve en la Figura 19 y en la Figura 20

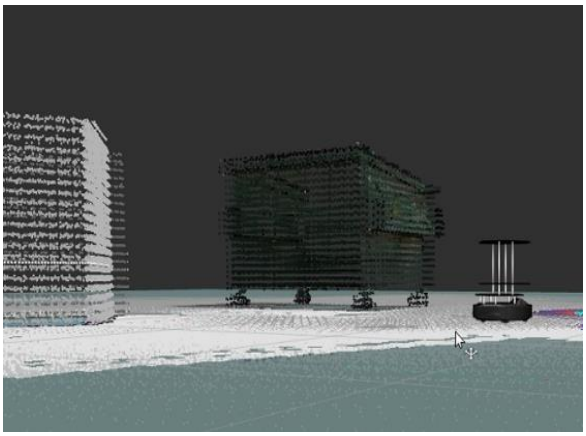

Figura 19 Mapeo obtenido teniendo un mínimo de 15 inliers

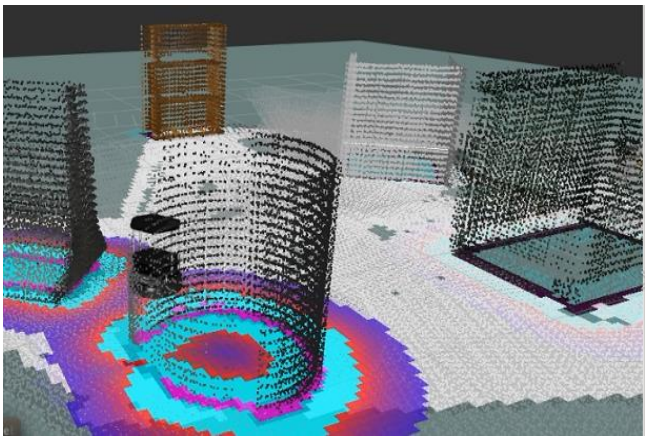

Figura 20 Mapeo obtenido teniendo un mínimo de 15 inliers

Como se puede ver la forma de los objetos, todas las líneas y el terminado de su borde está bien definido sin importar la forma de estos (cilíndricos o cuadrados).

Tras hacer estas pruebas y teniendo en cuenta los trabajos previos hechos con la librería se dedujo que una de las posibles razones para que las paredes de los objetos se corrieran era el hecho de que la simulación se estaba llevando a cabo en un ambiente abierto que no tenía paredes. Como consecuencia las imágenes obtenidas por la cámara se distorsionan generando este fenómeno en el mapa. Para ver si los resultados obtenidos se deben al ambiente o a la variación de parámetros se decidió repetir las pruebas en el mismo ambiente, pero colocando paredes alrededor de los objetos.

4.1.3 Conclusiones respecto ambiente de simulación

En términos generales este ambiente de simulación no tuvo mayores inconvenientes ya que permitió generar distintas simulaciones para comparar los efectos de la variación de parámetros en distintos entornos. Los mapas generados en esta herramienta son congruentes en términos de forma y por lo general tienen los bordes bien definidos. El único limitante de este ambiente de simulación está en que este no se asemeja lo suficiente al ambiente de prueba que se utilizara posteriormente, esto debido a que no tiene paredes alrededor de los objetos.

4.2 SEGUNDO ENTORNO DE SIMULACIÓN

4.2.1 Ambiente de simulación

El robot y la cámara usados para este ambiente se mantienen igual que los descritos en la sección 4.1.1 lo que cambia es la disposición de los objetos en el ambiente de simulación ya que estos tienen 4 paredes alrededor

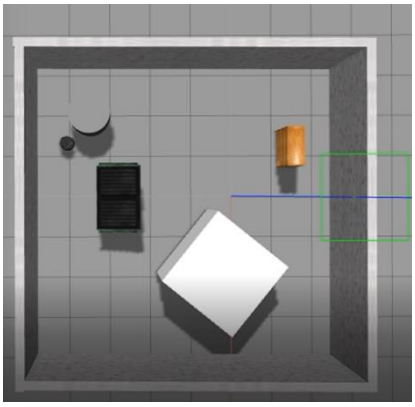

Figura 21 Distribución de los objetos del segundo entorno

4.2.2 Resultados obtenidos en el segundo entorno de simulación

Al realizar las simulaciones con este ambiente se generó un error en el mapeo, el cual se le atribuye al modelo del simulador, que hace que la imagen que está siendo mapeada se corra y no permita obtener un mapa adecuado como lo que se muestra en la Figura 23 y Figura 22

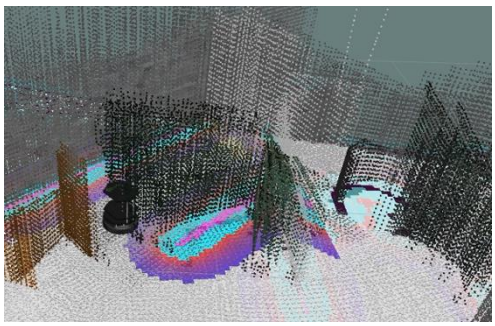

Figura 22 Visualización del error de mapeo

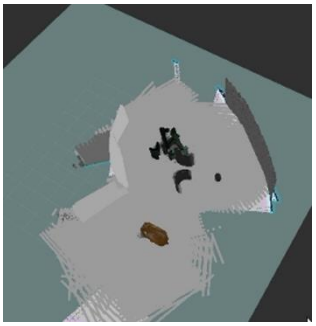

Figura 23 Vista superior del error de mapeo

La forma de los objetos se pierde por completo sin importar que parámetros se modifiquen. Como consecuencia el mapa que se obtiene carece de objetos definidos o paredes que lo rodeen Figura 22.

4.2.3 Conclusiones respecto al ambiente de simulación

A diferencia del entorno de simulación anterior este entorno tiene una gran limitante y es que no permitió generar ningún mapa sin importar que parámetros se utilizaran. Todos los mapas resultantes carecen de forma y no se asemejan para nada a el entorno que se quería mapear. Durante las pruebas de simulación se intentó usar diferentes configuraciones para el entorno, sin embargo, ninguna dio buenos resultados. Se usaron ambientes hechos a mano y también se intentó utilizar los ambientes que vienen por defecto en gazebo, en todos los casos se presentó el error mencionado anteriormente. Por esto se atribuye el error a un problema de la herramienta de simulación gazebo.

4.3 TERCER ENTORNO DE SIMULACIÓN

4.3.1 Ambiente de simulación

- **Sistema operativo:** Ubuntu 18.04.5 LTS (Bionic Beaver)
- **ROS:** ROS melodic
- **Versión Gazebo:** 9
- **Robot:** Turtlebot3 Waffle pi
    - Dimensiones: 281 x 306 x 141 mm

- Radio: 143.5 mm

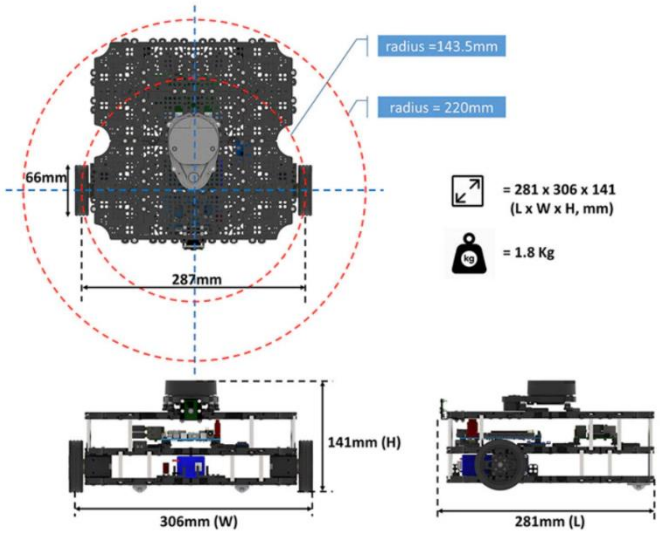

Figura 24 Medidas del Turtlebot3 Waffle pi [7]

- **Sensores:**
  - Camara Raspberry pi
  - 360° LiDAR
  - Giroscopio de 3 ejes
  - Acelerómetro de 3 ejes

- **Ambiente de simulación:** Para crear las simulaciones se usó el entorno que viene por defecto en gazebo para ROS melodic

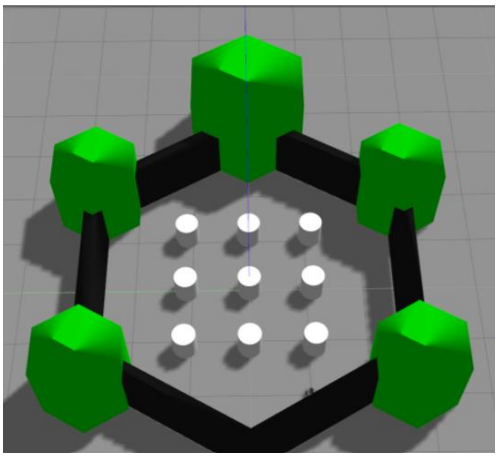

Figura 25 Imagen del tercer entorno de simulación [18]

4.3.2 Resultados obtenidos en el tercer entorno de simulación

Antes de explicar los resultados obtenidos en esta fase es importante resaltar que a diferencia de las etapas anteriores para este entorno de simulación se tiene un mayor conocimiento sobre las diferentes herramientas que se pueden usar para explorar los resultados obtenidos de una sesión de mapeo usando la librería RTAB-Map. Estas herramientas son database-viewer la cual permite ver todas las imágenes capturadas en el movimiento del robot junto con los features que se detectan en cada una de ellas. Adicionalmente se cuenta con la herramienta RTAB-Mapviz que permite ver la variación de los parámetros a medida que se va mapeando y los loop closure aceptados y rechazados.

Numero de Features

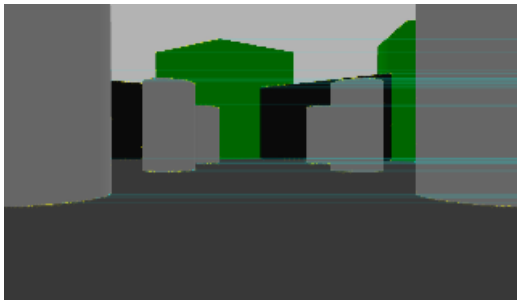

Figura 26 Resultado obtenido teniendo un máximo de 15 features por frame

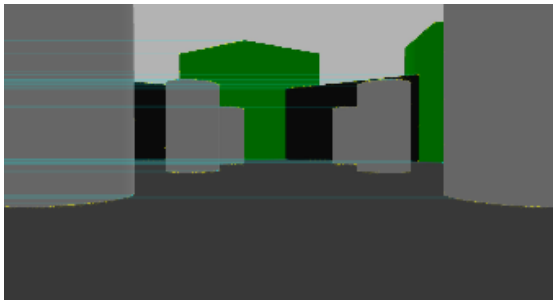

Figura 27 Resultado obtenido teniendo un máximo de 5 features por frame

Como se puede ver en la Figura 27 tiene una mayor cantidad de líneas azules que hacen referencia a los puntos amarillos (features) de la imagen a comparación de la Figura 26 que tiene una menor cantidad de líneas por lo que tiene una menor cantidad de features.

Para los demás parámetros se generó un mapa, como se hizo en las secciones anteriores, el mapa resultante se puede ver en la Figura 28. En todos los casos se obtuvo el mismo mapa y so fue posible evidenciar diferencias entre los mapas generados.

4.3.3 Conclusiones respecto al ambiente de simulación

Este entorno de simulación tuvo una gran limitación y es que ninguna de las variaciones de los parámetros se vio reflejada en los resultados vistos en los mapas, por ende, no se pueden sacar conclusiones de este.

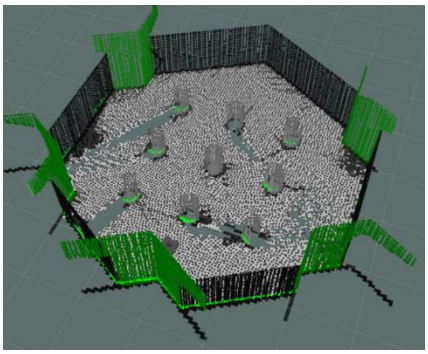

Figura 28 Mapa del tercer ambiente de simulación

CAPITULO V. TRABAJO CON ROBOT HUMANOIDE PEPPER EN AMBIENTE CONTROLADO

En esta parte del proyecto se decidió hacer las pruebas con los parámetros directamente con el robot humanoide Pepper en un ambiente para determinar los cambios en el mapa mediante la variación de parámetros.

5.1 AMBIENTE DE TRABAJO

- **Sistema operativo:** Ubuntu 16.04.7 (Xenial Xerus)
- **ROS:** ROS kinetic
- **Robot:** Pepper
    - **Peso:** 28 kg
    - **Altura:** 120 cm.
    - **Fondo:** 42,5 cm.
    - **Batería:** Litio, 30,0Ah/795Wh
    - **Motores:** 20
    - **Ruedas:** 3 (omnidireccionales)
    - **Movimiento:** 360º
    - **Velocidad máxima:** 3 km/h

  [19]
- **Camaras :**
    - **Camaras 2D**
        - **Modelo:** OV5640
        - **Resolucion:** 2560*1920
        - **Fps:** 15
    - **Camaras 3D**
        - **Modelo:** ASUS XTION
        - **Resolucion:** 320x240
        - **FPS** 20

  [8]

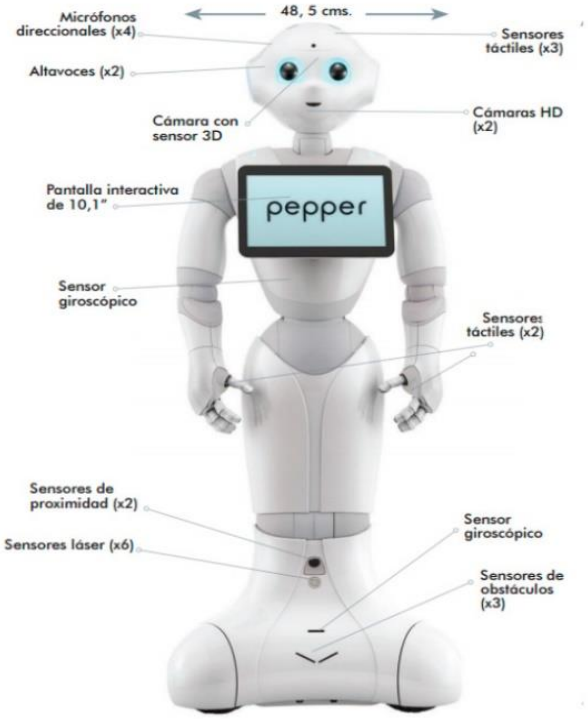

Figura 29 Imagen del Robot Pepper [19]

En la Figura 29 se ve una imagen que describe en que parte del robot humanoide Pepper se encuentran ubicadas las diferentes cámaras y los sensores laser.

El ambiente que se mapeo en esta etapa del proyecto consiste en un laberinto conformado por un set de paredes de cartón donde el robot debe navegar por diferentes espacios. La altura de las paredes es menor que la altura del robot y la iluminación de toda la escena es constante ya que todo el entorno está construido

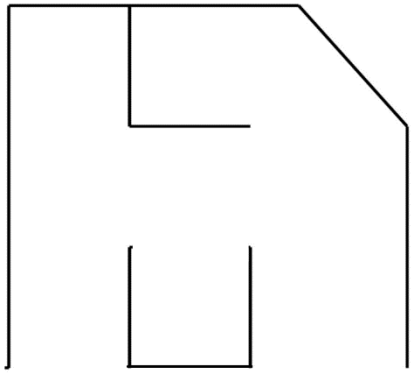

Figura 30 Vista superior del ambiente controlado usado para las pruebas

en dentro de uno de los laboratorios de la universidad. La Figura 30 presenta una imagen del entorno controlado construido

5.2 VERIFICACIÓN DEL FUNCIONAMIENTO DE LOS PARÁMETROS

Para garantizar que los resultados del mapeo sean correctos es necesario verificar que los tópicos que interactúan con la librería RTAB-Map estén publicando de forma correcta como se vio en el capítulo III se necesita que 5 tópicos le publiquen a RTAB-Map. No es suficiente que los tópicos esten activos, sino que también es necesario verificar que estos tópicos tienen frecuencia.

Los topicos de interes son

- Tópico del láser (/laser)
- Tópico de odometria (/odom)
- La transformación del robot respecto a las coordenadas globales (/tf)
- Camara de profundidad
- Camara rgb

Una vez se verificó que los tópicos estuviesen activos se prosiguió a verificar que todos ellos estuviesen publicando información constantemente.

Cámaras

Ya que es posible mapear se confirma que las cámaras están desplegadas y publicando en los tópicos necesarios de forma continua.

Tópico laser (/laser)

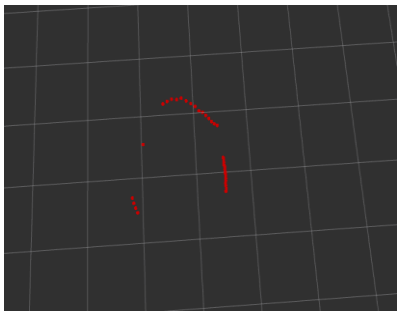

Figura 31 Visualización del tópico laser en la herramienta Rviz

En la Figura 31 se puede ver que lo que publica el tópico laser en la herramienta de visualización rviz. Cada uno de los puntos representa un punto del laser del robot.

Tópico odometria (/odom)

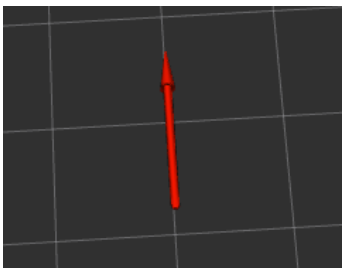

Figura 32 Visualización del tópico odom en la herramienta Rviz

En la Figura 32 se puede ver el tópico odom en la herramienta rviz. Este tópico tiene la información de la odometria del robot a medida que este se mueve y la actualiza a lo largo de todo el mapeo.

Tópico de transformacion (/tf)

```
vilma@vmtg:~$ rostopic hz /tf
subscribed to [/tf]
average rate: 50.726
        min: 0.000s max: 0.064s std dev: 0.01507s window: 45
average rate: 50.389
        min: 0.000s max: 0.064s std dev: 0.01505s window: 97
average rate: 50.250
        min: 0.000s max: 0.064s std dev: 0.01509s window: 146
average rate: 50.219
        min: 0.000s max: 0.064s std dev: 0.01494s window: 197
average rate: 50.177
        min: 0.000s max: 0.064s std dev: 0.01502s window: 247
average rate: 50.076
        min: 0.000s max: 0.064s std dev: 0.01498s window: 297
average rate: 50.126
        min: 0.000s max: 0.064s std dev: 0.01494s window: 347
average rate: 50.070
        min: 0.000s max: 0.064s std dev: 0.01505s window: 397
```

Figura 33 Publicaciones del tópico tf

En la Figura 33 se verifica que el tópico tf está publicando con cierta frecuencia. Este tópico tiene la posición del robot en el marco local. De esta forma se confirma que los 4 tópicos necesarios para ejecutar RTAB-Map están publicando información y tienen frecuencia al iniciar a manipular el robot.

El paso a seguir es verificar que al lanzar la RTAB-Map estos tópicos sigan publicando. Para verificar el funcionamiento correcto de los parámetros primero lanzamos únicamente el nodo de odometria visual de RTAB-Map. Esto se puede

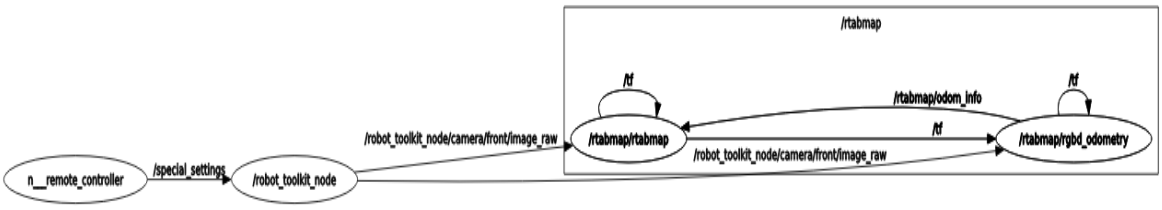

Figura 34 Grafo resultante de ejecutar el comando rqt_graph

ver en el siguiente grafo.

Como se puede ver en la Figura 34 el toolkit del robot le da las imágenes de las cámaras al nodo rtabmap **(/rtabmap/rtabmap)** y al nodo de odometria visual de rtabmap (**/rtabmap/rgbd_odometry**). Con esta información se obtiene una nueva transformada de la posición del robot con respecto al marco actual (**tf**) y se calcula la información de la odometria del robot nuevamente **(/rtabmap/odom_imfo)**, la cual se manda de nuevo al nodo de rtabmap para actualizar el mapa.

Al ejecutar el comando rostopic list se ve que los topicos de interes siguen publicando Al igual que en el paso anterior se verifica no solamente que estén *activos,* sino que todos los tópicos estén publicando de forma constante

Tópico laser (/laser)

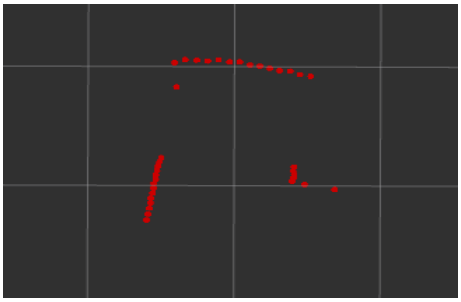

Figura 35 Visualización del tópico laser en la herramienta Rviz con RTAB-Map corriendo

En la Figura 35 puede ver que el tópico laser está publicando de forma constante aún con RTAB-Map corriendo, esto se puede corroborar ya que es posible visualizar la información publicada por el tópico con la herramienta rviz.

Tópico odometria (/odom)

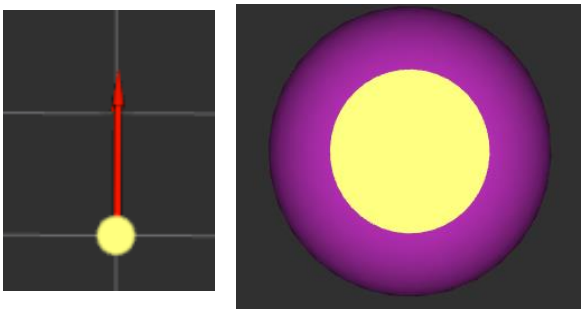

Figura 36 Visualización del tópico odom en la herramienta Rviz con RTAB-Map corriendo

En la Figura 36 se puede ver que el tópico odom está publicando de forma constante aún con RTAB-Map corriendo, esto se puede corroborar ya que es posible visualizar la información publicada por el tópico con la herramienta rviz

Tópico de transformación (/tf)

Figura 37 Actualización de RTAB-Map teniendo en cuenta la información recibida por las camaras

En la Figura 37 se ve que el tópico tf está publicando con cierta frecuencia se ve como se actualiza la información dada por RTAB-Map teniendo en cuenta la información recibida de las cámaras. De esta forma se confirma que los 4 tópicos necesarios para ejecutar RTAB-Map están publicando información y tienen frecuencia al iniciar a manipular el robot.

5.3 RESULTADOS

Los primeros parámetros analizados fueron aquellos que hacen parte *de* la odometria visual. *El* primer parámetro que se analizó fue el de **Vis/MaxFeatures.** Como se explicó anteriormente en este documento este parámetro determina el número máximo de features que se detectan por cada frame capturado. Al variar el valor del parámetro se espera que cuando mayor sea el valor de este parámetro mayor sea la calidad del mapa puesto que este va a tener un mayor nivel de detalle.

Figura 38 Resultado de recorrer el mapa con Max Features 10

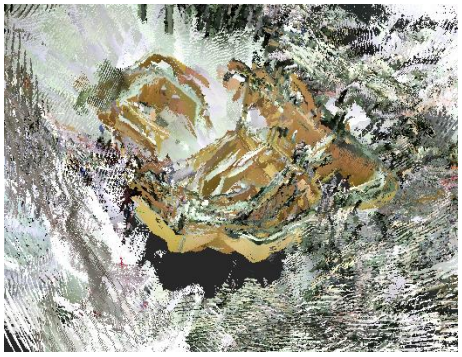

Figura 39 Resultado de recorrer el mapa con Max Features 20

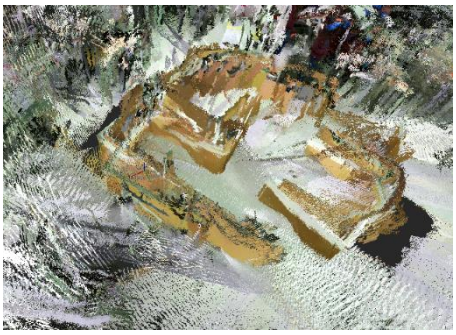

Figura 40 Resultado de recorrer el mapa con Max Features =100

Figura 41 Resultado de recorrer el mapa con Max Features =200

Como se puede ver de la Figura 38 a la Figura 41 todos los mapas tienen mucho ruido y los bordes de las imágenes se ven traslapadas ya que no están bien definidos. Después de definir los parámetros se ve que el que tiene un mejor funcionamiento es el que tiene Max Features con el valor de 10. Esto es contrario a lo que se esperaba debido a que a medida que la cantidad de features aumenta se espera que la calidad del mapa mejore pues hay una mayor cantidad de características formar las imágenes de los objetos presentes en el entorno.

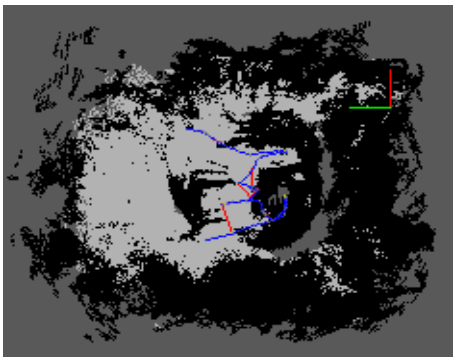

Figura 42 Mapa 2D de MaxFeatures 10

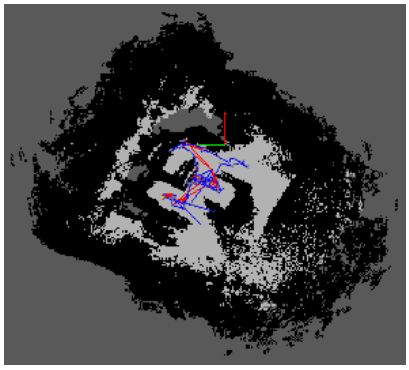

Figura 43 Mapa 2D de MaxFeatures 20

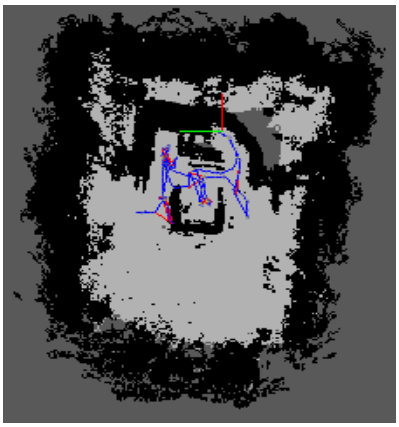

Figura 44 Mapa 2D de MaxFeatures 100

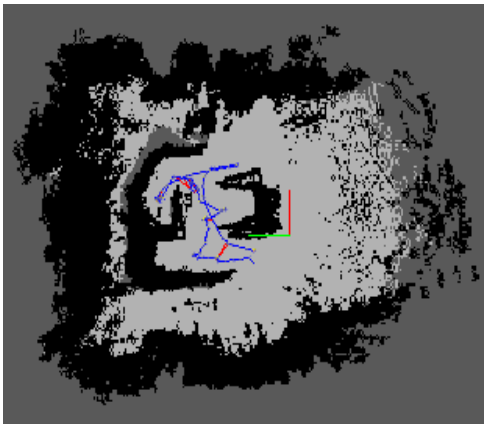

Figura 45 Mapa 2D MaxFeatures 500

En la Figura 42, la Figura 43, la Figura 44y la Figura 45 se pueden ver los occupancy gridmaps generados al variar los parámetros, es importante resaltar que la trayectoria en azul representa la trayectoria del robot y la trayectoria en rojo representa aquellos puntos en los que fue posible hacer loop closure.

En todas las imágenes es posible notar, por la trayectoria del robot, que el mapa generado no es solo del ambiente controlado, por el contrario, el robot está haciendo un mapa de todo el laboratorio. Esto se puede ver porque en todas las imágenes la trayectoria está situada en el centro, donde se encuentra el laberinto, y la imagen esta encerrada por un marco que representa los demás objetos en el laboratorio. Lo anterior se le puede atribuir al hecho de que durante el mapeo no fue posible mantener la cabeza del robot estática mirando hacia el piso. En los intervalos en los que el robot levantó la cabeza, debido a configuraciones de seguridad en su sistema operativo, se obtuvo información de las paredes del laboratorio.

El siguiente parámetro que se analizo fue el parámetro de **Vis/CorGuessWinSize** este parámetro limita la ventana, el rango de imágenes, con las cuales se puede comparar la imagen actual para obtener correspondencias.

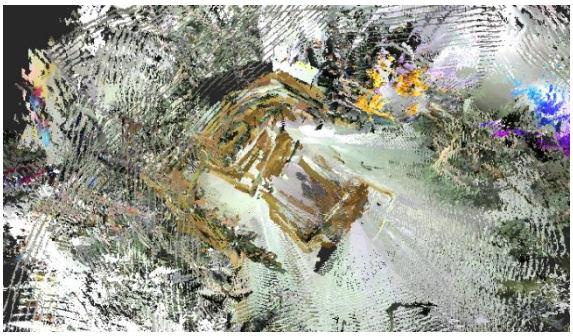

Figura 46 CorGuessWinSize con valor de 5

Figura 47 CorGuessWinSize con valor de 20

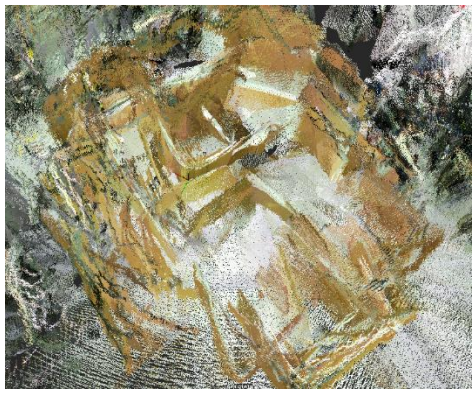

Figura 48 CorGuessWinSize con valor de 100

Como se puede ver en la Figura 46 la Figura 47 y la Figura 48 no existe ninguna mejora al variar el valor de este parámetro. Al ver que igual que con el parámetro anterior, La imagen está llena de ruido y los bordes están mal definidos. Teniendo los resultados obtenidos en simulación en cuenta se procede a modificar el parámetro **Vis/MinInliers** ya que este mostro mejoras en la definición de los bordes.

Este parámetro define el mínimo de Inliers que tienen que tener una imagen para aceptar una transformación en el mapa.

Figura 49 MinInliers con valor de 1

Figura 50 MinInliers con valor de 5

Figura 51 MinInliers con valor de 10

Figura 52 MinInliers con valor de 50

Al revisar los resultados obtenidos con la variación de parámetros, presentados de la Figura 49 a la Figura 52, no se ve ninguna mejora considerable al varias este parámetro opuesto a lo encontrado en las simulaciones. Los bordes siguen estando mal delimitados y no se ve ningún cambio al variar ninguno de los parámetros.

5.4 LIMITACIONES ENCONTRADAS EN EL AMBIENTE DE PRUEBAS

La primera limitación encontrada es que no se presenta ninguna mejora a partir de la variación de los parámetros, por lo tanto, todos los mapas generados no tienen la forma que se espera y no son útiles ya que no tienen una forma acorde al ambiente. Cómo consecuencia, en caso de querer hacer una prueba de localización es poco probable que el robot se ubique. La mala calidad de los mapas se puede atribuir a dos factores principales. El primero es la técnica de SLAM que utiliza la librería RTAB-Map y el segundo es el movimiento de las cámaras del robot durante el mapeo.

La técnica de SLAM estima el movimiento secuencial a lo largo de todo el mapeo generando un margen de error, este error se acumula a lo largo del tiempo generando una desviación entre los valores mapeados y los valores reales. Como consecuencia el mapa se distorsiona ya que los datos tienen un mayor error a medida que aumentan la cantidad de datos tomados.

La segunda limitación se presenta por el movimiento de la cabeza del robot a lo largo del recorrido. Cómo se mencionó al principio de este capítulo las cámaras del robot humanoide Pepper se encuentran ubicadas en la cabeza del robot, aun cuando el movimiento de la cabeza del robot fue controlado durante toda la sesión de mapeo en varias ocasiones el robot levanto la cabeza y miro hacia los lados debido a controles de seguridad del sistema operativo. Esto explica porque en los mapas se puede ver que no solo se encuentran los objetos del ambiente controlado dentro de los mapas, sino que también se pueden encontrar las paredes del laboratorio en ellos.

CAPITULO VI. TRABAJO CON ROBOT HUMANOIDE PEPPER EN PASILLO

En busca de obtener mejores resultados y ver el funcionamiento de los parámetros de **odometria visual** en un ambiente diferente se decidió experimentar con el mapeo del robot en uno de los pasillos del sótano 1 del ML, haciendo uso del robot humanoide Pepper cuyas características se describieron en capítulo anterior.

6.1 AMBIENTE DE TRABAJO

El segundo ambiente utilizado para pruebas es un pasillo del sótano 1 del edificio de ingeniería de la Universidad de los Andes. Algunas de las diferencias entre el ambiente anterior y este ambiente es que este ambiente tiene menos paredes y su forma es más sencilla, el robot no debe maniobrar por los diferentes segmentos si no que debe moverse en línea recta. Por otro lado, el pasillo es bastante ancho por lo tanto la distancia entre los bordes es mayor y consta de paredes altas. Finalmente, la iluminación de este ambienta varia debido a que existen segmentos del pasillo que están más iluminados que otros.

6.2 TÉCNICA DE MAPEO

Antes de empezar a variar los parámetros se realizaron varios recorridos de ida y vuelta con el robot para determinar cuál es la técnica correcta de mapeo. Esto es importante ya que el ambiente a mapear no solo es alto, sino que también es ancho, por lo cual para que los bordes del mapa queden bien definidos y la forma final del mapa sea acorde a la forma real del pasillo es necesario determinar que movimiento se le va a dar al robot durante el mapeo. Durante el proceso de mapeo del robot no se modificó ninguno de los parámetros, todos conservaron sus valores por defecto. En el caso de la odometria visual los valores de los parámetros durante las pruebas de mapeo son los siguientes:

- **Odom/Strategy:** 0
- **Vis/MaxFeatures:** 1000
- **Vis/MinInliers:** 20

La primera técnica de mapeo que se usa consiste en acercar al robot a una pared y girarlo para que escanee los bordes de esa pared. Una vez escaneada esa parte de la pared girarlo nuevamente hasta que quede en diagonal a la pared opuesta, avanzar en diagonal hacia la pared opuesta y cuando este en frente de esa pared girar el robot para obtener un escaneo adecuado de dicha pared. Esto se hace de forma repetitiva generando un patrón de zigzag.

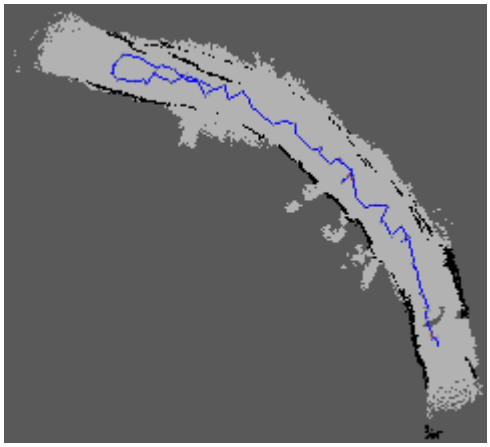

Figura 53 Mapa generado con la primera técnica de mapeo

Como se puede ver en la Figura 53 el patrón de zigzag (trayectoria en azul que representa el movimiento del robot) no es el correcto ya que el mapa generado no es consistente con la forma del pasillo, el mapa si presenta una noción de la longitud del pasillo y de las paredes a su alrededor sin embargo la forma de este no es la correcta. El pasillo mapeado es completamente recto y en este mapa pareciese que tiene una forma curva a medida que el robot avanza.

En busca de obtener mejores resultados se utiliza una segunda técnica de mapeo. Para esta segunda técnica de mapeo se mueve el robot paralelo a una pared, es decir, el robot avanza solamente cuando está mirando hacia el frente. Cada vez que

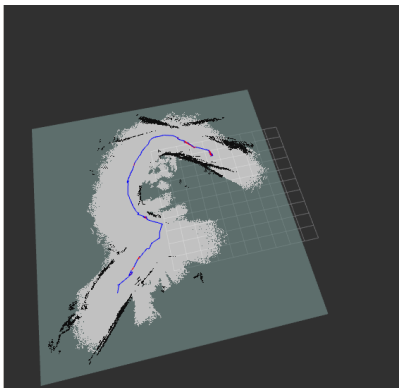

Figura 54 Mapa generado con la segunda técnica de mapeo

el robot avanza un poco este se gira sobre su eje en dirección a la pared a la que esta pegado para escanearlo. Para poder escanear las dos paredes lo que se hace es escanear una pared en el recorrido de ida y la pared opuesta en el recorrido de vuelta.

Como se puede ver en la Figura 54 esta técnica de mapeo tampoco es adecuada debido a que se pierde por completo la forma del mapa este obtiene una curva que no tiene y adicionalmente los bordes del mapa se pierden por completo lo cual tiene sentido ya que la forma del mapa no es congruente con el ambiente en la vida real.

Finalmente se intentó con una tercera técnica de mapeo que consiste en mover el robot solo por el centro del pasillo avanzando solo cuando este está mirando hacia el frente. Cada vez que el robot avanza un poco en el pasillo este se gira sobre su eje en contra de las manecillas del reloj para que escanee la pared de la izquierda se devuelve a su posición original y después se gira sobre su eje a favor de las manecillas del reloj para que escanee la pared de la derecha. Una vez escaneadas las dos paredes el robot se gira sobre su eje para que quede mirando hacia el frente y se avanza.

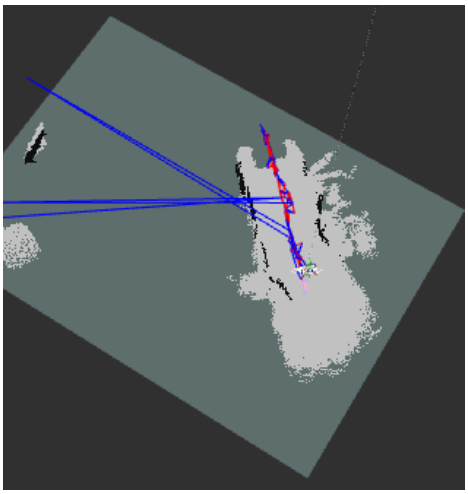

Figura 55 Mapa generado con la tercera técnica de mapeo

Como se puede ver en la Figura 55 el mapa generado con la tercera técnica de mapeo es el mejor hasta el momento debido a que este si tiene las características propias del ambiente pues el pasillo mapeado es recto. Algunas características de este mapa no son las correctas como, el recorrido del robot, ya que a mitad de camino este presenta dos perturbaciones en las cuales parece que hubiese salido del mapa, o la longitud del mapa ya que este es más largo en la vida real. Para mejorar estos mapas se van a variar los parámetros de odometria visual mencionados al principio de esta sección, manteniendo la tercera técnica de mapeo.

6.3 RESULTADOS

El primer parámetro que se varió es el parámetro **Odom/Strategy.** Este parámetro determina que técnica se utiliza para reconstruir el mapa en memoria, como en las técnicas de mapeo se mantuvo la técnica F2M se vuelve a tomar el mapa con la misma técnica cambiando la técnica de mapeo a F2F.

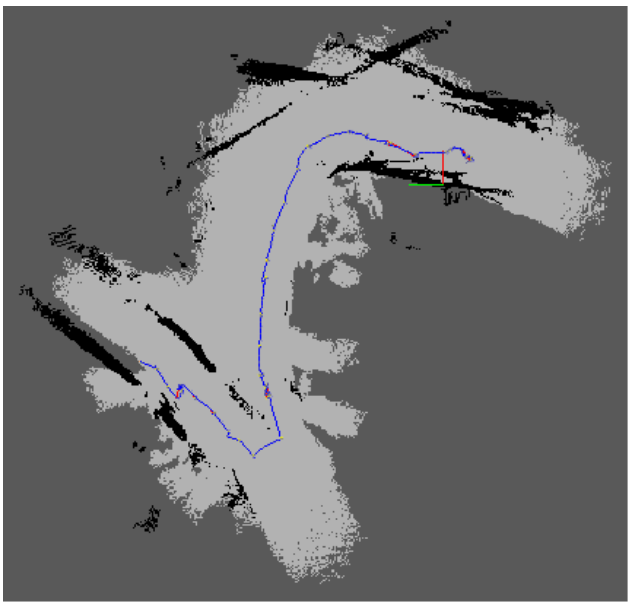

Figura 56 Mapa generado con Odom/Strategy F2F

Figura 57 Mapa 3D generadon con Odom/Strategy F2F

El mapa generado cambiando la estrategia de mapeo se puede ver en la Figura 56. Este mapa vuelve a tener los errores presentados en la Figura 54 ya que la forma del mapa no es acorde a la forma del pasillo en la vida real. Adicionalmente se puede ver que las paredes no están bien definidas ya que parece que hay paredes montadas una encima de otra generando una cruz entre ellas, lo cual se puede ver tanto en el mapa 2D como en el mapa 3D presentado en la Figura 57. De igual forma el pasillo no está acotado por las paredes como debería ser si no que solo hay

bordes definidos en algunas partes del mapa. Teniendo en cuenta las fallas presentadas en este mapa se define como estrategia optima de mapeo la estrategia F2M.

El siguiente parámetro que se varió el parámetro de **Vis/MaxFeatures** que varía el número máximo de features que se toman por cada frame. Ya que el parámetro por defecto es de 1000 features lo primero que se va a hacer es disminuir considerablemente el número de features por frame.

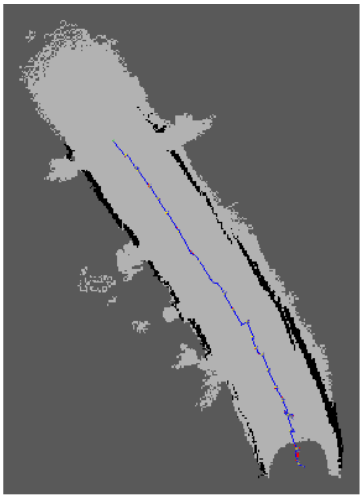

Figura 58 MaxFeatures con valor de 1000

Como se puede ver en la Figura 58 el mapa generado tiene una mejor calidad en comparación a los mapas generados anteriormente primero porque estos si son rectos acordes con la forma del pasillo. Segundo porque en estos si se evidencia de forma clara el contorno que representa las dos paredes del pasillo, que fueron escaneadas durante la trayectoria ida y vuelta del robot (trayectoria azul). Aun así, existen factores a mejorar en el mapa, el primero es que la pared de la derecha parece estar antes de que termine el piso, lo cual se ve evidenciado en los bordes grises por fuera de las líneas negras. Lo segundo que se puede evidenciar es el bajo número de loop closures presentados en el mapa (líneas rojas), pues a lo largo del mapa solo se pueden ver 2 puntos al principio en los cuales hay 2 pequeños puntos de loop closure.

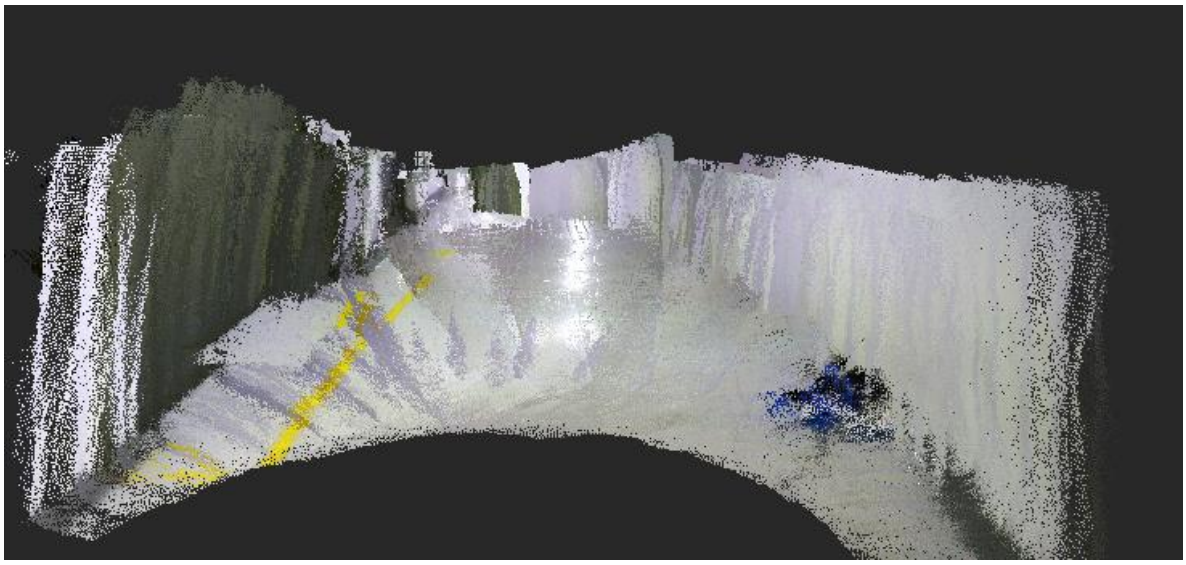

Figura 59 Mapa 3D del pasillo con MaxFeatures 100

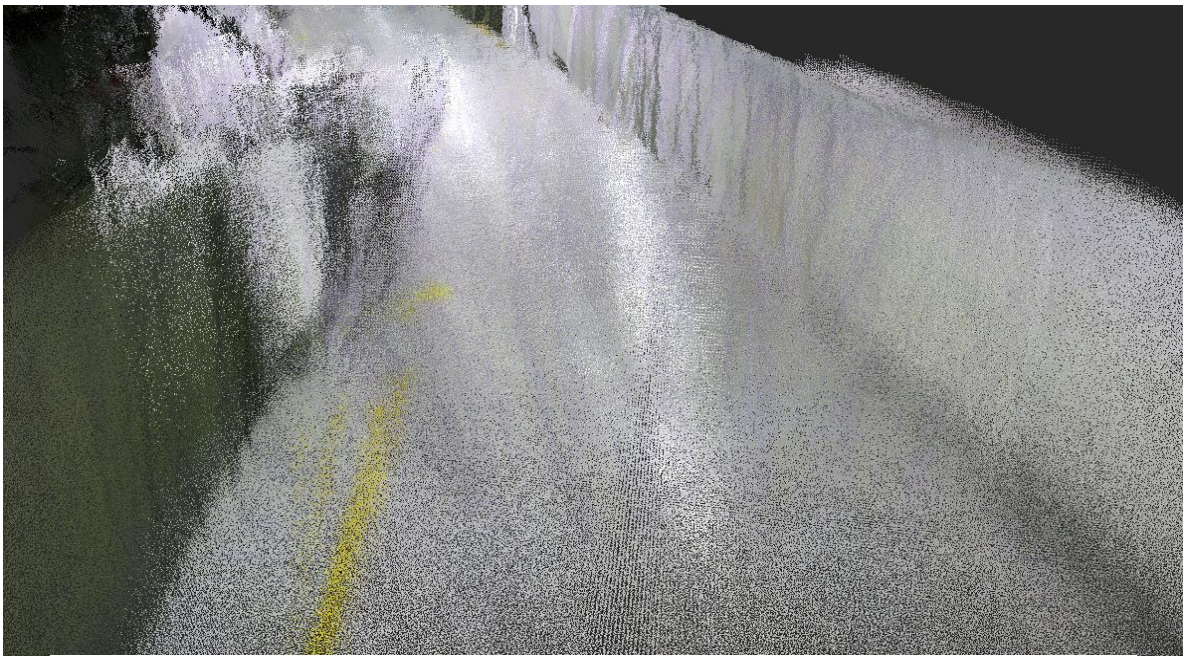

Figura 60 Mapa 3D del pasillo con MaxFeatures 100

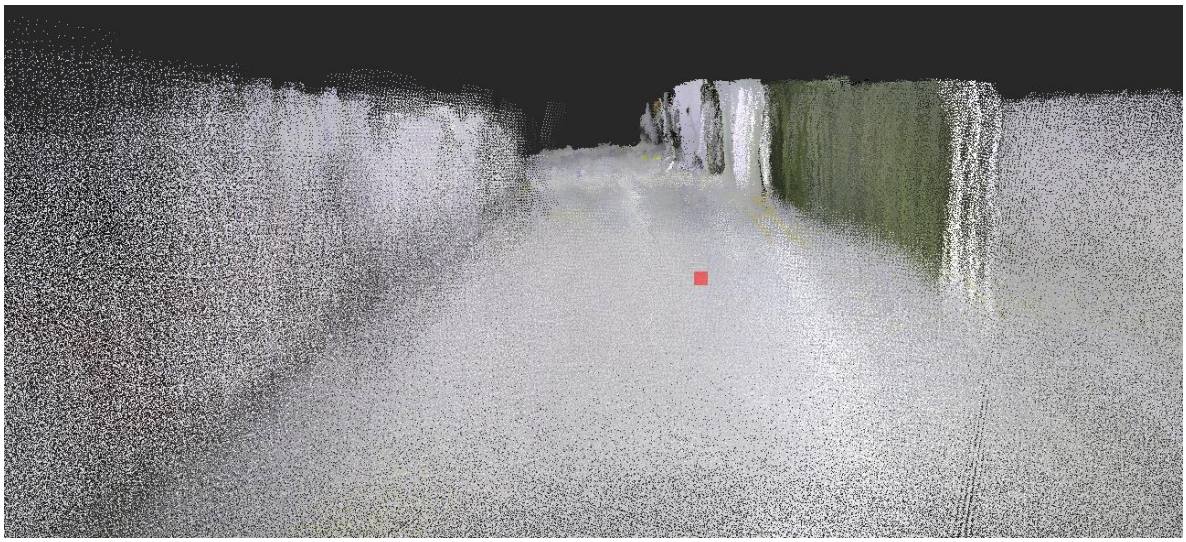

Figura 61 Mapa del pasillo con MaxFeatures 100

La Figura 59 la Figura 60 y la Figura 61 son capturas secuenciales el mapa 3D del pasillo, es decir cada foto se fue tomando a medida que el robot avanza su recorrido por el pasillo. Como se puede ver el mapa es bastante acorde a la realidad pues se pueden distinguir los diferentes objetos existentes en el pasillo que son las paredes y las puertas, y el ancho del pasillo es congruente con el del pasillo mapeado.

A lo largo del recorrido es posible notar varios defectos en el mapa, por ejemplo, en la Figura 60 la pared a la izquierda se puede ver una parte borrosa en el lugar donde hay una pared de vidrio. La Figura 61 es la continuación de lo que se ve al final de la Figura 59, si bien parece que la puerta estuviese al final del pasillo está realmente se encuentra a los lados. La explicación de porque ocurren estos fenómenos en el mapa 3D se dará en la siguiente sección de este capítulo.

Teniendo en cuenta los resultados presentados lo siguiente que se hizo fue aumentar el valor del parámetro **Vis/MaxFeatures** para ver como esto afectaba el mapa.

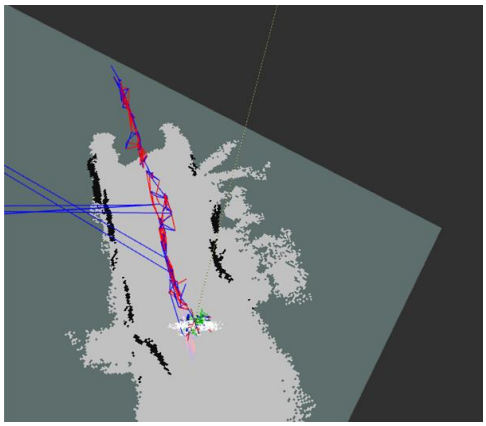

Figura 62 MaxFeatures con valor de 5000

Como se puede ver en la Figura 62 la calidad del mapa disminuyó a comparación del mapa que tiene un valor de 1000 en **Vis/MaxFeatures**, ya que este mapa no tiene la longitud correcta y los bordes no están bien definidos. Sin embargo, algo que es muy importante resaltar de este mapa es que el número de loop closures (ruta en rojo) es mucho mayor ya que a lo largo de todo el recorrido del robot se ven los loop closures. Esto se puede atribuir a que como se están tomando un mayor número de Features por frame es más sencillo encontrar las features que se comparten entre frame y frame a medida que el robot avanza.

El último parámetro que se varió fue el parámetro **Vis/MinInliers**, este parámetro dicta cual es el mínimo de inliers que debe tener una imagen para ser aceptada como una transformación en el mapa. Como el valor por defecto de este parámetro es de 20 lo primero que se hizo fue aumentar considerablemente el valor de este parámetro.

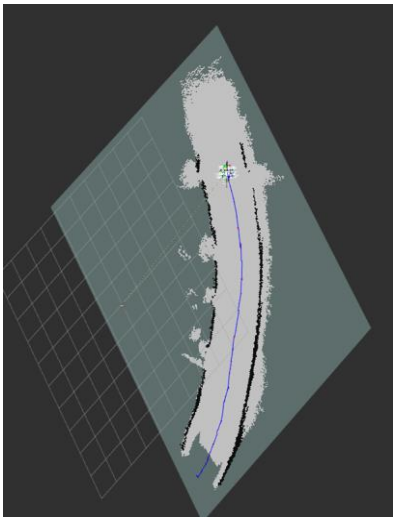

Figura 63 MinInliers con valor de 100

Como se puede ver en la Figura 63 este el mejor mapa hasta el momento en términos de definición de bordes y de forma. Los bordes a ambos lados del pasillo están claramente definidos y se puede ver de forma clara los espacios en donde se presentan ventanas transparentes pues es ahí donde se pierde un poco la forma sobre todo al lado izquierdo del pasillo. Es importante notar que aun cuando los bordes están bien definidos se sigue presentando el fenómeno visto en la Figura 58 en donde la pared de la derecha parece estar antes de que termine el piso por lo cual hay bordes grises por fuera de las líneas negras. En términos de forma, aunque este mapa presenta la longitud correcta si es posible notar que a medida que el robot avanza se va perdiendo la forma del pasillo ya que este se va curveando al igual que pasaba con la primera técnica de mapeo, aun así, la curva mostrada en este mapa es menor.

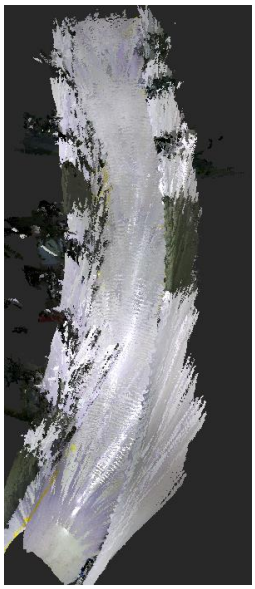

Figura 64 Vista superior del mapa en 3D MinInliers con valor de 100

En la Figura 64 provee una mejor visión de los objetos presentes en el mapa, aquí se puede ver que una de las razones por la cuales se pueden ver márgenes grises por fuera de los bordes delimitados en el mapa en 2D es por el la forma en que se superponen las imágenes tomadas por el robot al generar el mapa en 3D.

Finalmente se disminuyó el valor de este parámetro para ver de que forma esto afecta el mapa.

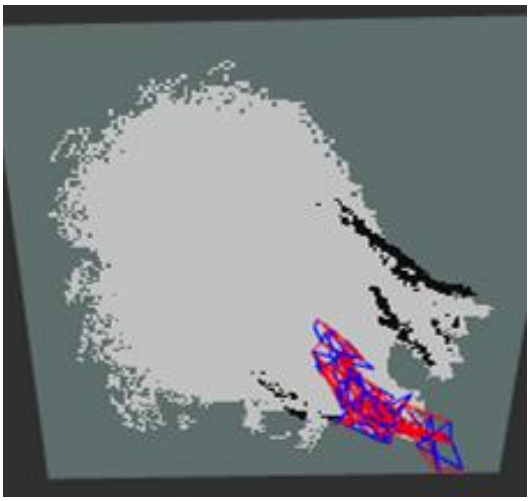

Figura 65 MinInliers con valor de 5

Como se puede ver en la Figura 65 el parámetro de **Vis/MinInliers** afecta de forma significativa la forma final que tiene el mapa. El tener un valor demasiado bajo en este parámetro daña por completo la forma que tiene el mapa ya que no se evidencia ningún tipo de forma en este mapa, aun cuando el robot hizo todo el recorrido de ida y vuelta. No existe ninguna definición de los bordes y parece que el robot solo hubiese recorrido una pequeña porción del pasillo.

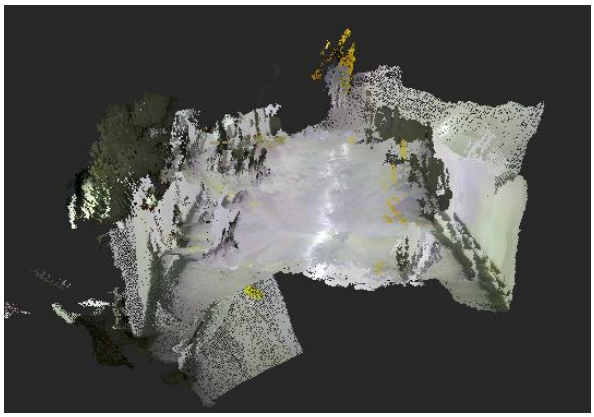

Figura 66 Mapa 3D del pasillo con MinInliers 5

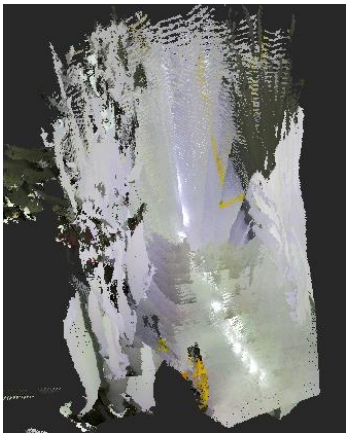

Figura 67 Vista superior del mapa 3D de con MinInliers 5

La Figura 66 presenta la vista del mapa con MinInliers en el valor de 5 desde el punto de vista del robot y la Figura 67 presenta una vista superior del mismo mapa. Ambas figuras corroboran lo que se ve en el mapa 2D analizado anteriormente donde no se genera un mapa completamente distorsionado y las imágenes tomadas por el robot están montadas unas sobre otras.

6.4 LIMITACIONES ENCONTRADAS EN EL AMBIENTE DE PRUEBAS

La principal limitación encontrada en este ambiente de pruebas está en la distorsión que se presenta en la mayoría de los mapas aun en aquellos que fueron ellos con la técnica de mapeo óptima para este ambiente. Si bien parte del error se le puede atribuir al error acumulado de la metodología SLAM (mencionado en el capítulo anterior) el error se le debe atribuir principalmente a la cámara de profundidad usada para estas pruebas.

Cómo se mencionó en el capítulo V la cámara de profundidad del robot humanoide Pepper es un modelo ASUS XTION, esta cámara de profundidad tiene dos grandes fallas, la primera es la forma como la imagen resultante de la cámara 3D se ve afectada por la luz del ambiente y la segunda es la falla de la imagen a profundidad de la cámara [16]. La falla causada por la luz genera que la cámara 3D del robot no pueda calcular de forma correcta la profundidad de aquellos objetos que tienen una sombra, como consecuencia no es posible delimitar bien los bordes que rodean estos objetos. Esto mismo aplica para objetos con superficies especulares, en este tipo de objeto la cámara pierde la percepción de profundidad y los mapea sin contorno.

La segunda falla de la cámara hace que los mapas en 3D generados por la cámara no tengan una forma acorde al ambiente que está siendo mapeado, ya que la profundidad de los objetos en la escena no es la correcta. Como consecuencia de la falla en la cámara de profundidad los objetos que son mapeados terminan teniendo una representación distorsionada en el espacio 3D, es decir en el mapa

resultante los objetos que en la realidad son planos terminan con una representación en forma de onda. Aun cuando la falla se presenta para todo tipo de objetos esta es más evidente en pisos ventanas y paredes. Es muy importante resaltar que esta distorsión empeora a medida que los objetos se alejan del sensor.

Partiendo de esta información es posible entender porque la tercera técnica de mapeo fue la que obtuvo mejores resultados, en la primera técnica de mapeo el movimiento en zigzag hace que una parte de la pared se mapeada cuando el robot está más lejos. Este movimiento constante hace que el error se acumule y como resultado final el mapa termina con una forma ondulada. De igual forma esto explica porque en mapas como el presentado en la Figura 63 que fueron generados con la técnica de mapeo óptima siguen presentando una leve curvatura.

CAPITULO VII CONCLUSIONES Y TRABAJO A FUTURO

7.1 DISCUSIÓN DE RESULTADOS Y BALANCE GENERAL DEL PROYECTO

A modo de conclusión el desarrollo de este proyecto permitió generar una documentación de varios parámetros de la librería donde se describe que factores se ven afectados con la modificación de cada uno de ellos. Adicionalmente los parámetros relacionados con odometria visual cuentan con pruebas en donde se puede ver cómo afectan los mapas que se generan en dos ambientes con características distintas.

A lo largo del proyecto las métricas cualitativas utilizadas para comparar los mapas unos entre otros fueron: la semejanza entre el mapa generado y el ambiente real en términos de forma, la definición de los bordes de los objetos en el mapa y la cantidad de loop closures que se hicieron durante el mapeo. Esta última métrica solo se tuvo en cuenta en el segundo ambiente de mapeo ya que los resultados presentados en el primer ambiente no fueron lo suficientemente buenos para tenerlos en cuenta.

Durante la simulaciones se obtuvieron diversos resultados en donde solo fue posible visualizar el funcionamiento de los parámetros para el primer ambiente de simulación. De esto se puede deducir que para lograr un buen mapeo en simulación es necesario usar un ambiente abierto ya que cerrar el ambiente genera problemas en el mapa. De igual forma teniendo en cuenta la falta de resultados en el tercer ambiente de simulación se puede deducir que para obtener cambios significativos en los mapas es necesario tener un entorno que contenga elementos de diversas características y que no sea simétrico.

El primer ambiente de experimentación con el robot humanoide Pepper (ambiente controlado) no dio los resultados esperados ya que no fue posible limitar el mapeo a el ambiente deseado. En el segundo ambiente de experimentación con el robot humanoide Pepper (ambiente no controlado), donde se obtuvieron mejores resultados, se intentó usar una última métrica que se basó en determinar en cuales mapas el robot tenía una capacidad para ubicarse una vez terminado el mapeo. Sin embargo, en ninguno de los mapas presentados fue posible lograr que el robot se ubicara.

Teniendo en cuenta los resultados encontrados en el capitulo VI es posible concluir que a la hora de mapear en espacios abiertos es muy importante encontrar una técnica de mapeo adecuada antes de empezar a variar los parámetros. Esto con el fin de lograr un buen mapeo en la experimentación. Partiendo de los resultados encontrados es posible decir que los parámetros más significativos para generar un buen mapa son los parámetros **Vis/MaxFeatures** y **Vis/MinInliers.**

7.2 TRABAJO A FUTURO

Teniendo en cuenta que este proyecto no alcanzo a cubrir las pruebas necesarias para todos los parámetros documentados en ambos ambientes a futuro es posible

continuar con el desarrollo de este explorando los parámetros documentados sin pruebas. De igual forma teniendo en cuenta las dificultades presentadas en el primer ambiente de experimentación con el robot humanoide Pepper es recomendable repetir estas pruebas limitando el movimiento de la cabeza del robot. Como se mencionó al final del capítulo V el continuo movimiento de la cabeza del robot (donde se encuentran las cámaras) generó errores de mapeo.

Otra mejora que se puede hacer para aumentar la calidad de los mapas es usar una cámara diferente a la que tiene el robot humanoide Pepper la cual se puede poner en la cabeza del mismo, los mapas generados con esta cámara pueden ser probados posteriormente por el robot.

BIBLIOGRAFÍA

[1] R. Hanabusa, Y. Nagase, S. Mitsui, T. Satake and N. Igo, "3D Map Generation for Decommissioning Work," 2020 3rd International Conference on Intelligent Autonomous Systems (ICoIAS), Singapore, 2020, pp. 46-50, doi: 10.1109/ICoIAS49312.2020.9081847.

[2] M. Labbe, F. Michaud, “RTAB-MAP as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation”, journal of Field Robotics, vol. 36, no. 2, pp. 416-446, 2019

[3] Civera J., Lee S.H. (2019) RGB-D Odometry and SLAM. In: Rosin P., Lai YK., Shao L., Liu Y. (eds) RGB-D Image Analysis and Processing. Advances in Computer Vision and Pattern Recognition. Springer, Cham.

[4] RTAB-Map Real-Time Appearance-Based Mapping Recuperado el 22 de julio de 2021 https://introlab.github.io/RTAB-Map/

[5] Kinect for Windows Sensor Components and Specifications Recuperado el 22 de julio de 2021 https://docs.microsoft.com/en-us/previous-versions/windows/kinect-1.8/jj131033(v=ieb.10)

[6] Clear path robotics Recuperado el 22 de julio de 2021 https://clearpathrobotics.com/turtlebot-2-open-source-robot/

[7] Robotis Turtlebot 3 Recuperado el 22 de julio de 2021 https://www.robotis.us/turtlebot-3-waffle-pi/

[8] Aldebaran 2.0.6.8 documentation Recuperado el 22 de julio de 2021 http://doc.aldebaran.com/2-0/family/juliette_technical/video_juliette.html

[9] National Ocean service What is lidar? Recuperado el 22 de julio de 2021 https://oceanservice.noaa.gov/facts/lidar.html

[10] SLAM (Simultaneous Localization and Mapping) Recuperado el 22 de julio de 2021 https://www.mathworks.com/discovery/slam.html#:~:text=SLAM%20(simultaneous%20localization%20and%20mapping)%20is%20a%20method%20used%20for,map%20at%20the%20same%20time.&text=Engineers%20use%20the%20map%20information,path%20planning%20and%20obstacle%20avoidance.

[11] *Reference Data for Radio Engineers*, ITT Howard W.Sams Co., New York, 1977, section 30

[12] Recuperado el 22 de julio de 2021 https://www.elmedia-video-player.com/frame-by-frame-video-player-mac.html

[13] Microsoft academic Key Frame Definition Recuperado el 22 de julio de 2021 https://academic.microsoft.com/topic/2780139006/publication/search?q=Key%20fr

ame&qe=And(Composite(F.FId%253D2780139006)%252CTy%253D%270%27)&f=&orderBy=0

[14] Lindeberg, T. (2008). Scale-Space. In Wiley Encyclopedia of Computer Science and Engineering, B.W. Wah (Ed.). https://doi-org.ezproxy.uniandes.edu.co:8443/10.1002/9780470050118.ecse609

[15] Detección de características (visión por computadora) Recuperado el 22 de julio de 2021 https://es.wikipedia.org/wiki/Detecci%C3%B3n_de_caracter%C3%ADsticas_(visi%C3%B3n_por_computadora)

[16] Z. Bauer, F. Escalona, E. Cruz, M. Cazorla and F. Gomez-Donoso, "Refining the Fusion of Pepper Robot and Estimated Depth Maps Method for Improved 3D Perception," in *IEEE Access*, vol. 7, pp. 185076-185085, 2019, doi: 10.1109/ACCESS.2019.2960798.

[17] Learn TurtleBot and ROS Recuperado el 22 de julio de 2021 https://learn.turtlebot.com/2015/02/03/3/

[18] Robotis e- manual Recuperado el 22 de julio de 2021 https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/

[19] Pepper el futuro de la robótica nos habla Recuperado el 22 de julio de 2021 https://aliverobots.com/robot-pepper/