Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

"*Exploración Autónoma de Robots Móviles Basada en Curvas B-Splines*"

Tesis Profesional

Que para obtener el título de:

Ingeniero en Tecnologías de la Información

Presenta:

Esteban Calixto Cruz

Asesores:

Dr. Abraham Sánchez López,

Dr. Alfredo Toriz Palacios

Puebla, México Enero, 2021

Agradecimientos

Principalmente a mi familia y mi novia por el apoyo que me han brindado a lo largo de mi vida, tanto en el aspecto académico como en lo personal, es gracias al esfuerzo de todos que he logrado alcanzar una nueva meta al culminar mis estudios universitarios y llegar a ser la persona que soy hoy en día.

Al Dr. Abraham Sánchez López, por todo su esfuerzo dedicado a compartir su conocimiento y su apoyo durante mi formación académica, mostrándome con el ejemplo, a ser un profesional dedicado y responsable, todo esto se ve reflejado en el presente trabajo de tesis bajo su dirección. De igual forma, agradezco su amistad que me ha ofrecido todos estos años.

A mis profesores, que se esforzaron por compartir sus conocimientos y pusieron a prueba mis habilidades, con el único objetivo de hacerme crecer y tener una correcta formación académica, en especial al M.C. Juan Carlos Conde Ramírez y al Dr. Alfredo Toriz Palacios.

A mis amigos Laura, Albert, Juan Pablo y Jiuber, que conocí a lo largo de mi vida universitaria, que me enseñaron el valor del trabajo en equipo, me ofrecieron su amistad y apoyo siempre que lo necesite. Con los cuales compartí muchas experiencias, de aprendizaje y diversión, que nunca olvidare y a pesar de los obstáculos, siempre logramos salir adelante juntos, por esto y más, les agradeceré siempre.

Índice

Índice de Figuras

Capítulo 1

Introducción

En la actualidad la robótica enfrenta grandes retos, dentro de los cuales, uno de los más desafiantes es el obtener mecanismos robustos y eficientes para modelar ambientes de creciente complejidad, haciendo uso de robots móviles para su exploración. Este problema de localización y mapeo simultáneo es conocido como SLAM, el cual nace de la necesidad de construir un modelo del entorno y estimar al mismo tiempo la ubicación del robot. Por lo anterior, es necesario enfocarse en la navegación de robots móviles, la cual permite desarrollar tareas robustas en donde es preciso desplazarse en ambientes complejos sin ayuda o supervisión humana, para esto, se hace uso de las capacidades para sensar y moverse sin colisionar con obstáculos, existentes en el ambiente, que se pueden integrar a los robots. Muchos enfoques han tratado con el problema de SLAM sin tener un éxito concreto, dado que se limitan a extraer características geométricas y representar sus posiciones en el mapa o discretizar el espacio en células y clasificar cada una como ocupada o vacía, lo que provoca que se restringa el campo de aplicación a ambientes con geometrías bien definidas o de dimensiones limitadas, para ello, se han generado nuevas propuestas para dar solución al problema de asociación de datos para SLAM, destinado a la construcción de ambientes complejos, como lo es la incorporación de curvas B-Splines como una herramienta para la descripción de obstáculos con geometrías complejas.

En el laboratorio de MOVIS se han realizado proyectos referentes a la planificación de movimientos y trayectorias. Así mismo, en los últimos 15 años se han propuesto estrategias para la navegación y exploración con robots móviles, dichas propuestas han sido implementadas, en su mayoría, a nivel de simulación. Sin embargo, los resultados obtenidos han sido satisfactorios en las diversas áreas de aplicación con robots en entornos reales.

El documento de tesis está dividido de la siguiente manera:

El capítulo 2 contiene un resumen del estado del arte, con el cual, se presentan los trabajos previos que forman un precedente para el trabajo que se realiza, los cuales abarcan diferentes tópicos como lo son las alternativas que se han tomado para la solución de problemas de SLAM, las bases de implementación e introducción de las curvas B-Splines dentro de los problemas de exploración, entre otros. El Capítulo 3 es el capítulo central de la tesis, en él se presenta la definición de la estrategia propuesta y los recursos utilizados para su implementación, de igual forma, se detallan los fundamentos que se utilizan para la localización del robot, el uso del algoritmo de filtro de partículas y la gestión de los datos utilizando curvas B-Splines. En el Capítulo 4 se presentan los resultados obtenidos de la implementación de la estrategia propuesta bajo diferentes entornos desconocidos para el robot, esto a nivel de simulación, dentro de los escenarios se presentan situaciones diferentes donde se pondrá a prueba el algoritmo de exploración integrada basado en SRT que se propone y el resultado de obtener las características del mapa utilizando las propiedades de las curvas B-Splines. Finalmente, en el capítulo 5, se establecen las conclusiones y el trabajo a futuro que se puede realizar para continuar con el tema central de la tesis.

En este proyecto de tesis se propone desarrollar un sistema robótico con el objetivo de realizar la exploración autónoma, auto localización y mapeado de ambientes desconocidos con ayuda de un robot móvil, en este caso el robot diferencial Pioneer P3-DX, teniendo como principal característica la incorporación de curvas B-Splines como medio para describir obstáculos con geometrías complejas dentro del entorno de exploración, y utilizar dicha información contenida en ellas, para encontrar puntos característicos del ambiente que puedan ser asociados. Las estrategias para el desarrollo del sistema robótico serán presentadas más adelante.

Capítulo 2

Estado del arte

Un aspecto fundamental para la exploración en ambientes desconocidos con robots móviles, es precisamente el hecho de conocer las características que posee tanto el robot móvil como el entorno a explorar, al igual que sus condiciones, el cual puede ser estático o dinámico. A partir de dichas características, es posible recabar información que puede ser eficazmente aprovechada para la localización del robot, y, en consecuencia, en la calidad del mapa que se obtiene al finalizar la exploración. La localización y mapeo simultáneo (SLAM) es un problema complejo en la investigación de la robótica móvil, dentro del contexto del problema, un robot móvil explora y censa una región desconocida, además construye un mapa y se localiza en él.

2.1 Localización

Para realizar la exploración de un entorno determinado la mayoría de los robots viajan a través de él. Esto significa que algunos comandos de movimiento tienen que ser enviados a los actuadores de movimiento del robot para que este pueda desplazarse, a medida que el robot se mueve a través del entorno, se da lugar a un problema conocido como errores odométricos. Los errores odométricos hacen referencia a un desface entre el movimiento deseado especificado a través de los comandos de control y el movimiento real logrado por

los actuadores de movimiento. El problema de la localización autónoma se ha investigado por más de dos décadas, en dicho tiempo, se han generado una variedad importante de paradigmas que buscan determinar la posición y orientación del robot con respecto a los objetos en el entorno. Una vez que se adquiere la información referente a la pose del robot, por localización, se utiliza como marco de referencia para el mapeo, de esta forma, se interpretan y localizan las observaciones del robot y se construye un mapa a partir de ellas. Con lo anterior, se establece que la localización y el mapeo son interdependientes.

De igual manera, se reafirma que la navegación es una de las más desafiantes competencias requeridas en un robot móvil y para que esta tenga éxito se deben contemplar cuatro bloques de construcción de navegación [1]. Uno de los bloques es el mencionado previamente, la **localización**, donde el robot debe determinar su posición en el ambiente, por lo cual, también debe ser capaz de interpretar sus sensores para extraer los datos significativos que están presentes en el ambiente explorado, esto se conoce como **percepción**, mientras el robot realiza la exploración debe decidir cómo actuar con el fin de cumplir los objetivos definidos inicialmente esto se define como **conocimiento** y finalmente se debe establecer el **control de movimientos** para que el robot pueda modular las salidas de sus motores para alcanzar la trayectoria deseada [2].

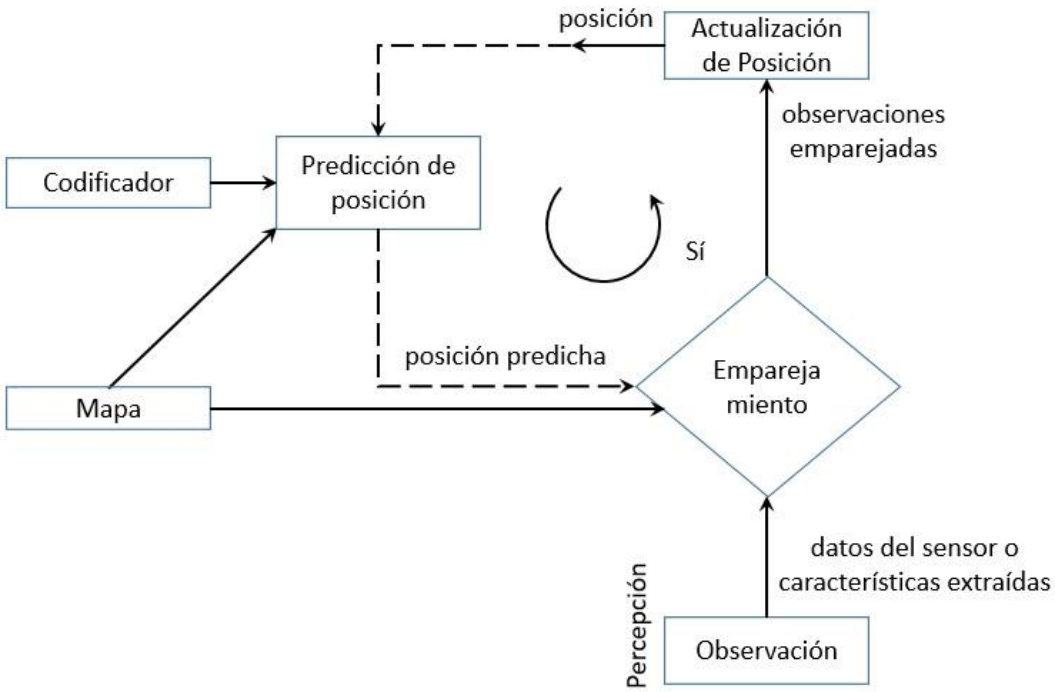

Figura 2.2.1 Esquema general para localización de robots móviles.

Como se ha mencionado, los problemas odométricos hacen imposible que el rendimiento perfecto del robot y el conocimiento se logren bajo un entorno real, debido a que mientras se desplaza en el entorno el robot pierde información acerca de su ubicación. Para evitar que la incertidumbre de la posición crezca de forma indiscriminada, el robot debe localizarse en relación con su mapa de entorno, para esta tarea, puede hacer uso de sus sensores y realizar observaciones de su entorno. De tal manera, la información proporcionada por la odometría del robot, más la información recabada de las observaciones, se pueden combinar, para abordar de mejor forma, el problema fundamental de estimar la ubicación del robot. En [1] la solución a este problema se plantea a través de técnicas probabilísticas, las cuales se basan en una representación discreta de posiciones robóticas, a las cuales se les asigna un valor de probabilidad, el cual representa la creencia de que el robot se encuentre efectivamente en dicha posición. En resumen, el objetivo principal es calcular la probabilidad de cada posición del robot de manera individual dada el codificador y el sensor que el robot ha recogido, y de esta manera, tener un conocimiento más realista de la posición que puede adoptar el robot durante los diferentes momentos de la exploración.

La localización probabilística es un algoritmo probabilístico, que permite que las incertidumbres que surgen de los modelos de movimiento incierto y las lecturas ruidosas de los sensores se tengan en cuenta de una manera basada en principios. El filtrado de Kalman, es una forma de estimación probabilística, que utiliza una representación de densidad de probabilidad gaussiana, otros métodos de estimación probabilística son los métodos bayesianos, que primordialmente incorporan conocimiento previo para poder estimar modelos útiles dentro de un espacio muestral y de este modo estimar parámetros que provengan de la experiencia y el filtrado de partículas, estos métodos permiten manejar distribuciones más generales.

2.2 SLAM

Existen ocasiones en las que se conoce de antemano el entorno en el cual se desplazará un robot, sin embargo, en la mayoría de los casos el agente robótico deberá moverse en un entorno totalmente desconocido, donde no se tiene acceso a un mapa del entorno, y es en ese momento cuando los problemas de SLAM (Simultaneous Location and Mapping) surgen. El objetivo del proceso de SLAM es usar el entorno para actualizar la posición del

robot, ya que no se puede confiar directamente en la odometría, por ello, se utilizan escaneos laser del entorno para corregir la posición del robot.

Un aspecto que podemos establecer, sin ninguna duda, es la necesidad de que los robots conozcan su localización, por lo tanto, un mapa es fundamental para que esta tarea sea más fácil. Al proporcionar al robot de un contexto espacial más elaborado se puede lograr desarrollar un comportamiento más inteligente, por esta razón, los mapas permiten a los robots ir más allá de un comportamiento reactivo. En la mayoría de las aplicaciones el robot tiene que estar equipado con sensores que le permitan observar su entorno y construir un mapa para sí mismo (mapeo). Si asumiéramos un conocimiento perfecto de la ubicación del robot, el principal reto del mapeo es realizar la descripción más precisa de la realidad física basada en las lecturas de los sensores, incluso con el ruido asociado al sistema de medición [3].

En general, el mapa y la información asociada a la pose del robot nos permiten definir el contexto en cual se pueden relacionar observaciones, decisiones y acciones que deberá emprender el robot a lo largo del tiempo de exploración. De esta manera, al aplicar técnicas de SLAM, es que se pude realizar con éxito el proceso de localización autónoma de una forma completa, donde un robot móvil se encuentra en un lugar desconocido perteneciente a un medio ambiente igualmente desconocido, todo esto debido a que SLAM no necesita un conocimiento previo del entorno. En [4] se afirma que la definición de un método robusto de SLAM dotaría a los robots de una verdadera autonomía, ya que, en la actualidad, los métodos de SLAM se han limitado a plataformas especializadas, entornos pequeños y ciertas tecnologías de sensores. De esta manera las investigaciones se han centrado en la necesidad de encontrar una solución robusta para SLAM, que sea funcional en los diferentes modelos de robots y que permita la construcción de mapas grandes y precisos de los entornos, teniendo en consideración que, la solución deberá ejecutarse en tiempo real y trabajar bajo limitaciones en la memoria de los robots.

Los algoritmos de SLAM puede ser aplicado de forma online u offline. El online tiene como objetivo que el robot sea capaz de procesar la información mientras navega en el ambiente, en otros términos, se realiza una estimación sobre la posición posterior momentánea, que adopta el robot, en apoyo con el mapa:

$$p(x_t, m|z_{1:t}, u_{1:t}) \tag{2.1}$$

Bajo este contexto x_t corresponde a la pose del robot en el tiempo t, m es el mapa y $z_{1:t}$ y $u_{1:t}$ hacen referencia a las medidas y los controles, respectivamente. Por otra parte, en el offline el proceso de SLAM es realizado sobre un conjunto de datos que se recabaron con anterioridad en algún robot, que comparte características en cuanto a sus sensores y odometría. En [5] se hace mención que, un algoritmo de SLAM puede llevar una representación o mapa de su ambiente en diversas maneras. Dada la variedad de mapas, es fundamental realizar un estudio de sus ventajas y desventajas. De tal manera, que al realizar la elección de un mapa se deberán tener en cuenta, según [2], lo siguiente:

- ¿Es necesario mantener el mapa en nociones métricas?
- ¿Cuál es el uso que se le dará al mapa?
- ¿El entorno es dinámico?
- ¿De qué sensores se dispone?
- ¿De cuánto poder de cómputo se dispone?

2.2.1 Filtro de Kalman extendido

Retomando la historia de la aplicación de algoritmos SLAM se encuentra que el más influyente de estos se basa en el filtro de Kalman extendido (EKF), el cual es una extensión del filtro de Kalman, donde las ecuaciones no lineales son linealizadas. Una de las primeras soluciones que aplican este enfoque se presenta en [6], donde el proceso de construcción del mapa se realizó considerándose como una extensión de la tarea de localización. En este trabajo tanto el robot como la construcción del mapa se realizan simultáneamente, para ello, un único vector de estado es utilizado, y de esta forma, se representan todas las variables del sistema en las que se está interesado. Inicialmente los trabajos se desarrollaban bajo la base de Kalman SLAM, sin embargo, tras el avance de investigaciones aparecieron algoritmos más sofisticados con técnicas para extraer y representar características presentes en el entorno, algoritmos para realizar la fusión de diferentes tipos de sensores,

métodos para mejorar la asociación de datos y técnicas que permiten reducir el costo computacional y mejorar la consistencia del filtro. A partir de todas las mejoras que surgieron del clásico Kalman SLAM surge EKF, siendo uno de los enfoques de SLAM más populares utilizados en la actualidad.

En el EKF-SLAM, el mapa es un gran vector de sensores de apilamiento y los estados de puntos de referencia, se modela por una variable gaussiana. Este mapa, también llamado mapa estocástico, es mantenido por el EKF a través de los procesos de predicción, por medio del movimiento de los sensores, y corrección dadas las observaciones realizadas por los sensores sobre los puntos de referencia, que se encuentran en el entorno tras un mapeo previo. El filtro de Kalman extendido asume que el modelo dinámico del sistema y el modelo de observación son establecidos por funciones no lineales f y h, respectivamente:

$$X_{k+1}^{r} = f(X_k^r, U_k) + w_k \tag{2.2}$$

$$Z_k = h(X_k^r) + v_k \tag{2.3}$$

Donde X_k^r son los estados del sistema, mientras que U_k representa los controles de entrada. Z_k son las mediciones de los puntos de referencia, con $k \in N_0$ que representa el tiempo discreto. Por último, w_ky v_k son la representación de los ruidos de entrada que tiene el sistema, dichos ruidos, tiene las características de ser de tipo gaussiana y con media cero.

El EKF-SLAM es considerado, por la mayoría de los autores, como un proceso de 3 etapas:

1. **Estado de Predicción:** Hace referencia al movimiento que realiza el robot, es necesario estimar donde estará el robot en un tiempo k en el futuro, esto es realizado a partir de estimaciones incrementales, donde la distribución resultante, después de que el robot se desplaza, es calculada en base a el estado X_k previo del robot. Sin embargo, en medida que el robot se mueve, la incertidumbre de la posición crece y la correlación con las características del mapa disminuye.

2. **Estado de Actualización:** Esto sucede en el momento que una característica es re-observada, esta etapa se aplica con el objetivo de reducir la incertidumbre y

mejorar la estimación del estado en general. La estimación de la pose del robot será cada vez menos precisa, dada la incertidumbre del actuador, esto mientras no se produzcan re-detecciones históricas.

3. **Añadir Nuevos Puntos de Referencia:** Se trata de añadir la información que relaciona viejos puntos de referencia con los nuevos. La información de la pose del robot es indispensable para agregar la nueva información.

En la parte inicial del algoritmo se cuenta con la posición del robot y una matriz de covarianza, la cual representa la incertidumbre de pose, por lo tal, ningún punto de referencia se ha añadido al mapa. Al detectarse un nuevo punto de referencia el vector de estado y la matriz de covarianza se extienden por medio del proceso de inicialización denominado aumento de estado. El algoritmo estándar de Kalman se resume de forma compacta en la figura 2.2.1.

Algoritmo del Filtro de Kalman Extendido $(\hat{\boldsymbol{x}}_{\boldsymbol{k-1}}, \boldsymbol{P}_{\boldsymbol{k-1}}, \boldsymbol{u}_{\boldsymbol{k}}, \boldsymbol{z}_{\boldsymbol{k}})$

---

1. $\hat{X}_k^- \leftarrow f\left(\hat{X}_{k-1}, u_k, 0\right)$
2. $P_k^- \leftarrow A_k P_{k-1} A_k^T + Q_k$

3. $K_k \leftarrow P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}$
4. $\hat{X}_k \leftarrow \hat{X}_k^- + K_k \left(z_k - h\left(\hat{X}_k^-, 0\right)\right)$
5. $P_k = (I - K_k H_k) P_k^-$

6. $\hat{X} \leftarrow [\hat{X} + x_N y_N]^T$
7. $P^{N+1N+1} \leftarrow J_x \, P_k \, J_x^T + J_z R J_z^T$
8. $P^{rN+1} \leftarrow P^{rr} J_x^T$
9. $P^{N+1r} \leftarrow (P^{rN+1})^T$

Figura 2.2.1 Algoritmo EKF.

Dentro del contexto del algoritmo EKF, $\hat{X}$ representa los estados del sistema, P es la matriz de covarianza del sistema, A es el jacobiano del modelo de predicción, u es la entrada de control, z son las observaciones, Q representa la incertidumbre asociada, R es la covarianza de la observación, H es el jacobiano del modelo de medición h, K es la ganancia de Kalman, J_z es también el jacobiano del modelo de predicción, pero con respecto al modelo de medición.

Analizando la estructura del algoritmo se encuentra que, en las dos primeras líneas, se define la **etapa de predicción**, en la línea 1 se describe el proceso asociado al movimiento del robot en el cual se incorpora el control u_k, mientras que la línea 2 denota la incertidumbre asociada a P_k^-, estas estimaciones solo afectan la pose estimada del robot. De las líneas 3 a 5 se lleva a cabo **la etapa de actualización,** en la línea 3 se calcula la ganancia de Kalman K_k incorporando la incertidumbre de la medición, asociada a la observación actual z_k. Por su parte, las líneas 4 y 5 denotan el proceso en el cual la matriz de ganancia de Kalman propaga cambios a lo largo de la estimación del mapa. Finalmente, en las líneas 6 a 9 se establece el proceso de **añadir nuevos puntos de referencia**, en la línea 6 el vector de estado X es actualizado con un nuevo punto de referencia, y para terminar se añade la covarianza para el nuevo hito a la a matriz de covarianza del sistema P, esto en las líneas 7 y 9.

EKF-SLAM tiene la particularidad de mantener la incertidumbre total durante todo momento, mientras se realiza la construcción online del mapa. Se debe puntualizar que el proceso de localización tiende a volverse más difícil y menos preciso cuando pocos puntos de referencia están involucrados en el, esto genera un efecto domino que provoca que la asociación de datos también sea más difícil por las incertidumbres que deben superarse. En general EKF-SLAM es una solución importante adoptada en diferentes investigaciones, sin embargo, deben considerarse las limitaciones que presenta como que:

- La distribución gaussiana asumida para el estado del sistema no corresponda a la realidad, ya que el EKF hará que las estimaciones de los momentos de la distribución se vayan degenerando con el tiempo, esto producirá valores optimistas para la matriz de covarianza del mapa obteniendo como resultado inconsistencias, tal como se plantea en [7].

- El costo computacional crece al cuadro a partir del número de objetos contenidos en el mapa. Esto debido a la necesidad de actualizar la matriz completa de covarianza del mapa después de cada medición, esto es una limitante para su aplicación en un entorno real.

Como se ha establecido el EKF-SLAM es una solución que ha sido adoptada y se ha popularizado su aplicación, si bien es importante entender que no es un proceso simple para implementarse este dota de ventajas significativas para atacar el problema de SLAM. De igual manera, se deberán considerar las limitantes presentadas, dado que ambos problemas se vuelven críticos en escenarios grandes.

2.2.2 Filtro de Partículas

Los filtros de partículas son modelos matemáticos, los cuales, representan la distribución de probabilidad como un conjunto de partículas discretas en el espacio de estados [8]. El principal objetivo de este método es rastrear una variable de interés, típicamente no gaussiana y potencialmente multi-modal en las funciones de distribución de probabilidad.

El filtro de partículas (FP) es utilizado para estimar el estado de un sistema que sufre cambios a través del tiempo. Fue propuesto por N. Gordon, D. Salmond y A. Smith en 1993 [9], bajo el nombre de Filtro Bootstrap, con la intención de implementar Filtros Bayesianos recursivos mediante el método de Monte Carlo, el cual tiene una especial aplicación en los problemas que tienen una difícil solución por métodos exclusivamente analíticos o numéricos, pero que dependen de factores aleatorios o se pueden asociar a un modelo probabilístico artificial.

En concreto, el FP representa la densidad a posteriori mediante un conjunto discreto de N partículas $(m_1, \dots, m_N)$ y sus probabilidades asociadas $(\pi_1, \dots, \pi_N)$, el principal inconveniente de los filtros de partículas es que escalan exponencialmente con la dimensión del espacio de estado. Sin embargo, su habilidad y ventaja para modelar distribuciones multimodales, por un conjunto de muestras, es una ventaja que los coloca por encima de otros filtros. El filtro de Kalman, como se ha mencionado, se restringe a distribuciones gaussianas.

El algoritmo de FP consiste en tres etapas:

1. **Muestreo:** Crea una nueva generación $S_t^{'}$ de partículas basada en el conjunto previo S_{t-1}.

2. **Ponderación de Importancia:** En esta etapa se calcula una importancia de peso para cada muestra del conjunto $S_t^{'}$.

3. **Remuestreo:** Toma N muestras del conjunto $S_t^{'}$. De tal manera que, la probabilidad para tomar una partícula es proporcional a su peso. El nuevo conjunto S_t es dado por las partículas tomadas.

Las etapas antes mencionadas están representadas en el algoritmo del FP, el cual se muestra en la figura 2.2.2:

Algoritmo del Filtro de Partículas

$\boldsymbol{Entrada}$: $Conjunto\ de\ muestras S_{t-1} y\ datos\ d.$

1. $S_t^{'} = \emptyset$
2. $\boldsymbol{para}\ i = 1\ hasta\ N\ hacer$
3. $\quad tomar\ \hat{s}\ \pi(s_t |\ s_{t-1}^{[i]}, d)$
4. $\quad \widehat{w} = \eta[p(\hat{s} \mid s_{t-1}^{[i]}, d)][\pi(\hat{s} \mid s_{t-1}^{[i]}, d)]^{-1} \qquad donde\ \eta\ es\ un\ normalizador$
5. $\quad S_t^{'} = S_t^{'} + (\hat{s}, \widehat{w})$
6. $\boldsymbol{fin}$
7. $S_t = \emptyset$
8. $\boldsymbol{para}\ j = 1\ hasta\ N\ hacer$
9. $\quad tomar\ una\ muestra\ s_t^{[i]}\ de\ S_t^{'}. Entonces\ s_t^{[i]}\ es\ tomada\ como\ probabilidad\ w_t^{[i]}$
10. $\quad S_t = S_t + (s_t^{[i]}, \frac{1}{N})$
11. $\boldsymbol{fin}$
12. $\boldsymbol{regresar}\ S_t$

Figura 2.2.2 Algoritmo del filtro de partículas.

El filtro es una alternativa interesante y en diversos estudios se ha incorporado para la solución de problemas de SLAM, siendo el algoritmo de FastSLAM es más popular. Esta es una solución estocástica de SLAM que involucra tres conceptos importantes:

- **Rao-Blackwellisation:** La aproximación resulta de dividir la distribución conjunta del espacio de estado en una parte muestreada, en la que los estados de la pose del robot se representan por medio de partículas, y los estados de referencia se estiman analíticamente mediante filtro de Kalman, la partición de estado está definida como:

$$
\begin{aligned}
& p\left(X_{v0:k}, m \mid Z_{0:k}, U_{0:k}\right) \\
& = p\left(X_{v0:k} \mid Z_{0:k}, U_{0:k}\right) p\left(m \mid X_{v0:k}, Z_{0:k}, U_{0:k}\right) \\
& = p\left(X_{v0:k} \mid Z_{0:k}, U_{0:k}\right) p\left(m \mid X_{v0:k}, Z_{0:k}\right)
\end{aligned}
\tag{2.4}
$$

  Donde $X_{v0:k}$ hace representa el histórico de posiciones del vehículo, m es el conjunto de puntos de referencia, $Z_{0:k}$ es una secuencia de mediciones, por su parte, $U_{0:k}$ es una secuencia, pero de entradas de control.

- **Independencia Condicional:** Esta parte hace referencia a que las características en SLAM se correlacionan debido a la incertidumbre en la posición del robot. En un ámbito utópico donde la trayectoria del robot fuera perfectamente conocida cada característica podría ser estimada independientemente, esto es representado en la figura 2.2.3 donde se denota la trayectoria S_k de los controles U del robot, las características del mapa k se establecen como Z, y son condicionalmente independientes. La importancia práctica de la independencia condicional entre las características es que, dada una partícula de la trayectoria del robot, podemos estimar cada una de las características de forma independiente, lo que genera un algoritmo con un costo lineal en el número de partículas.

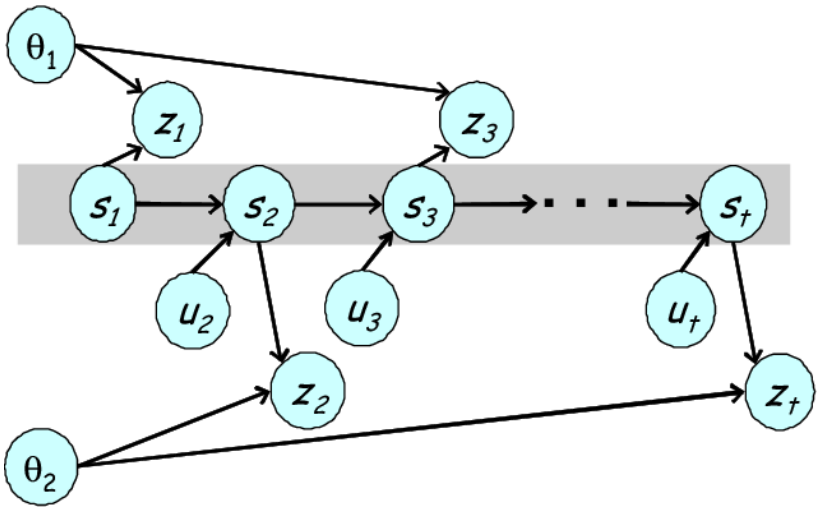

Figura 2.2.3 Dependencias probabilísticas entre variables SLAM en una red bayesiana.

- **Re-muestreo:** En esta etapa se le permite al robot concentrar partículas en altas regiones de probabilidad pertenecientes a la distribución. Esto es de utilidad en distribuciones que tienden a evolucionar en el tiempo, correspondientes a estados dinámicos.

Al ser un filtro bayesiano, el algoritmo FastSLAM se conforma en los pasos de predicción y actualización como se explicó en el EKF-SLAM [3]. En este proceso cada partícula se comporta de la forma:

$$X_k^{[m]} = (X_k^{[m]}, M_{1,k}^{[m]}, P_{1,k}^{[m]}, \dots, \mu_{N.k}^{[m]}, P_{N,k}^{[m]}) \tag{2.5}$$

Donde $[m]$ indica el índice de la partícula, $X_k^{[m]}$ es la trayectoria estimada, y tanto $\mu_{N.k}^{[m]}$ como $P_{N,k}^{[m]}$ son la media y la covarianza del valor gaussiano respectivamente, que representa el enésimo hito o punto de referencia asociada a la partícula m^{th}. De tal manera, en el paso de predicción del FastSLAM la distribución de probabilidad, de la siguiente posición del robot, se obtiene a partir de la distribución previa perteneciente al tiempo $k-1$, al igual que nuevas poses, todo esto utilizando el más reciente comando de movimiento u_k como:

$$X_k^{[m]} = p\,(x_k \mid X_{k-1}^{[m]}, u_k) \tag{2.6}$$

Sin embargo, cuando la observación es más precisa en relación con el ruido generado por el vehículo al desplazarse aparece un gran problema, dado que la predicción se realiza solamente con el control de movimiento. Al observar esta situación se adoptó una propuesta donde las poses se muestrean bajo la considerando el movimiento u_k y la medida z_k. Esta conclusión fue llamada FastSLAM 2.0:

$$X_k^{[m]} = p\ (x_k \mid X_{k-1}^{[m]}, u_k, z_k, n_k) \tag{2.7}$$

Las variables $n_k = n_1, \dots, n_k$ son variables de asociación de datos, en la que cada n_k especifica la identidad del punto de referencia observado en un instante k.

Ahora en el paso de actualización al obtener una nueva medición z_k de una característica n, cada partícula de la trayectoria es ponderada de acuerdo a la medición esperada y la medición real:

$$w_k^{[m]} = N(z_k \mid x_k^{[m]}, (\mu_{N.k}^{[m]}, P_{N,k}^{[m]})) \tag{2.8}$$

Donde el factor $w_k^{[m]}$ es llamado importancia del peso de la partícula, en otras palabras, es el cociente de la división de la distribución del objetivo entre la distribución propuesta.

2.3 Incorporación de curvas B-Splines

Como se ha mencionado a lo largo del presente capítulo, la solución al problema de SLAM ha sido objeto de importantes investigaciones en las últimas décadas. No obstante, las limitaciones para encontrar una solución completa a esta problemática también continúan existiendo, una de las más importantes es la asociada con el modelado geométrico de entornos arbitrariamente complejos. En [10] un algoritmo de SLAM basado en curvas Spline es propuesto como herramienta de representación del entorno. La principal diferencia entre este enfoque y los algoritmos tradicionales de SLAM, es que no se confía en una geometría específica. Esto permite modelar el ambiente realizando sus adecuaciones a los procesos de **segmentación y asociación de datos**, posteriormente a través de la aproximación clásica del Filtro de Kalman Extendido (EKF), se logra abordar el problema de SLAM.

2.3.1 Segmentación y ajuste de datos

El enfoque presentado por el Dr. Pedraza en [10], nace de la idea de usar un sensor láser que permita recabar información perteneciente al ambiente, con lo cual se obtiene un conjunto de m puntos de datos en el espacio $\mathbb{R}^2$ ligados entre sí por la lógica establecida por el sensor al realizar su barrido. La intención es simple, aprovechar la información recabada al procesarla con el método EKF-SLAM, el primer paso consiste en segmentar los conjuntos de mediciones, que representan los objetos físicos que son detectados.

El siguiente paso es obtener un conjunto inicial de $m-1$ vectores que conecten dos puntos de datos consecutivos obtenidos por el láser:

$$P_i = d_i - d_{i-1} \tag{2.9}$$

La siguiente parte del proceso consiste en realizar dos tipos de comparaciones con el objetivo de generar los vectores finales de puntos que representaran los objetos en el mapa.

- **Primera Comparación:** Se realiza con el motivo de buscar elementos lo suficientemente cercanos, de tal forma que, estos puedan pertenecer al mismo elemento en el entorno.

$$\max(\| P_i \|, \| P_{i+1} \|) \leq \eta \cdot m(\| P_i \|, \| P_{i+1} \|) \tag{2.10}$$

- **Segunda Comparación:** Tiene el fin de buscar un ángulo α_i formado por dos vectores P_iy P_{i+1}, de tal manera, que ese ángulo puede sugerir que ambos puntos pertenecen al mismo objeto dentro del entorno.

Los autores establecen para $\eta \in [1.5, 2]$ y para $a_{\max} \in [0, \frac{\pi}{4}]$ como valores adecuados, dentro de las respectivas comparaciones.

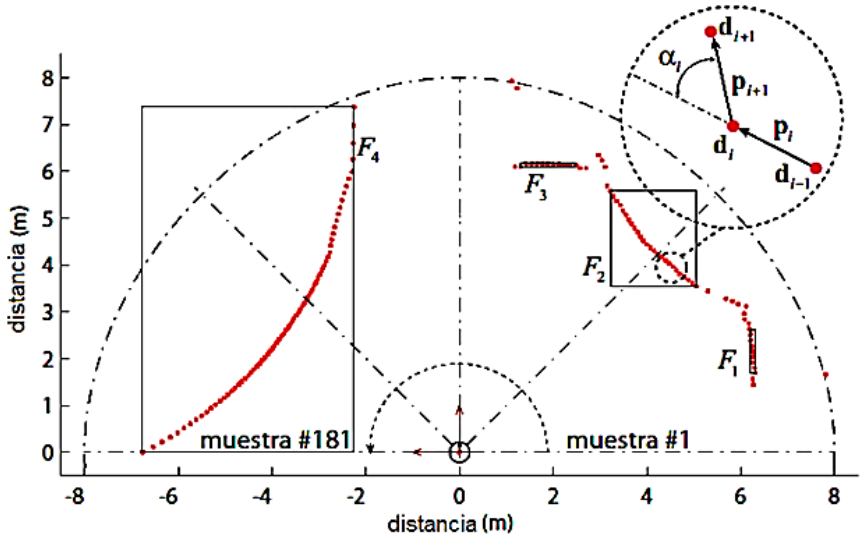

Figura 2.3.1 Criterio de segmentación de datos brutos provenientes del sensor láser.

De igual manera, una consideración adicional fueron ángulos demasiados cerrados como un criterio de separación de características. Finalmente, cada uno de los obstáculos obtenidos en el proceso son ajustados a curvas B-Splines de tercer grado [11]:

$$D_k = C(t_k) = \sum_{i=0}^{n} N_{i,p}\,(t_k) X_i \qquad for\ 0 \le k \le m. \tag{2.11}$$

2.3.2 Asociación de datos

Con el proceso de segmentación finalizado, el siguiente paso consiste en establecer una correspondencia entre cada uno de los m segmentos obtenidos en el instante k y una (o ninguna) de las n características contenidas en el mapa que está siendo construido [12]. Lo primero que se realiza es una asociación que se denomina “burda” (Figura 2.3.2a), donde los puntos de control de cada segmento de curva se comparan con los puntos de control de las características almacenadas en el mapa, por medio del siguiente criterio:

$$\min\left(dist\left(X_{m,i}, X_{o,j}\right)\right) \le d_{min} \qquad \begin{cases} i = 1 \ldots n_{m,} \\ j = 1 \ldots n_o \end{cases} \tag{2.12}$$

Si la distancia entre los puntos de control de la Spline en el mapa y los puntos de control de la Spline observada es mayor al umbral d_{min}, se descarta la asociación. En caso contrario, la el proceso de asociación se ejecuta (Figura 2.3.2b):

- **Paso Uno**: Se considera como punto a uno de los extremos de la curva observada, mientras que b, es el punto más cercano a a en la B-Spline contenida en el mapa. Si b resulta ser un extremo de la B-Spline entonces es necesario calcular el punto c, siendo este el punto más cercano a b en la B-Spline observada.

- **Paso Dos:** El proceso se repite utilizando el otro extremo de la B-Spline observada (punto d en la Figura 2.3.2b). Este punto es asociado a con el punto e de la B-Spline en el mapa.

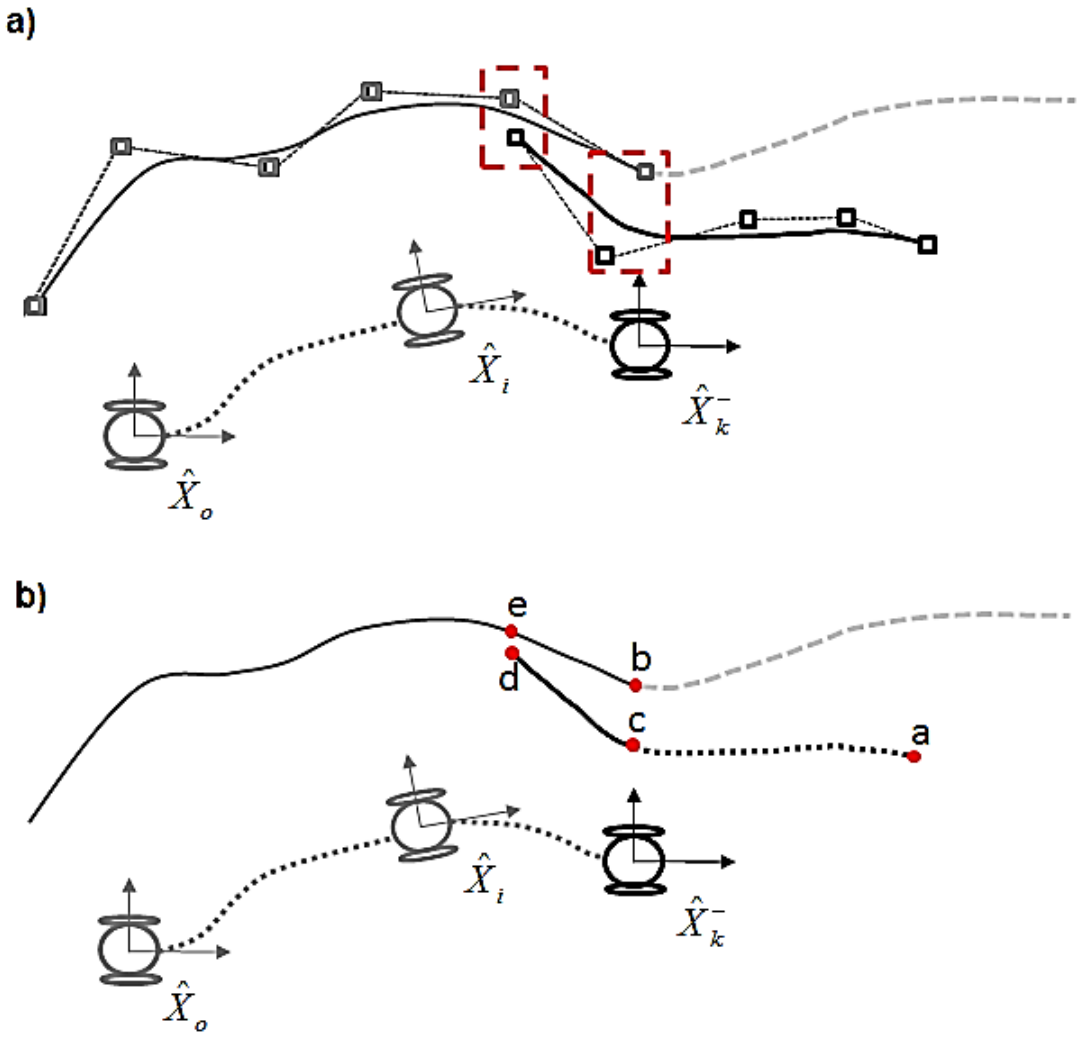

Figura 2.3.2 Concordancia entre las curvas.

Como parte final del procedimiento, se hace uso del EKF el cual necesita una expresión matemática que permita utilizar la asociación de datos bajo el contexto de la localización, para que, de esta forma, se logre predecir las mediciones que se esperan obtener por el sensor dada la pose del robot y el conocimiento del ambiente en el instante k (Figura 2.3.3).

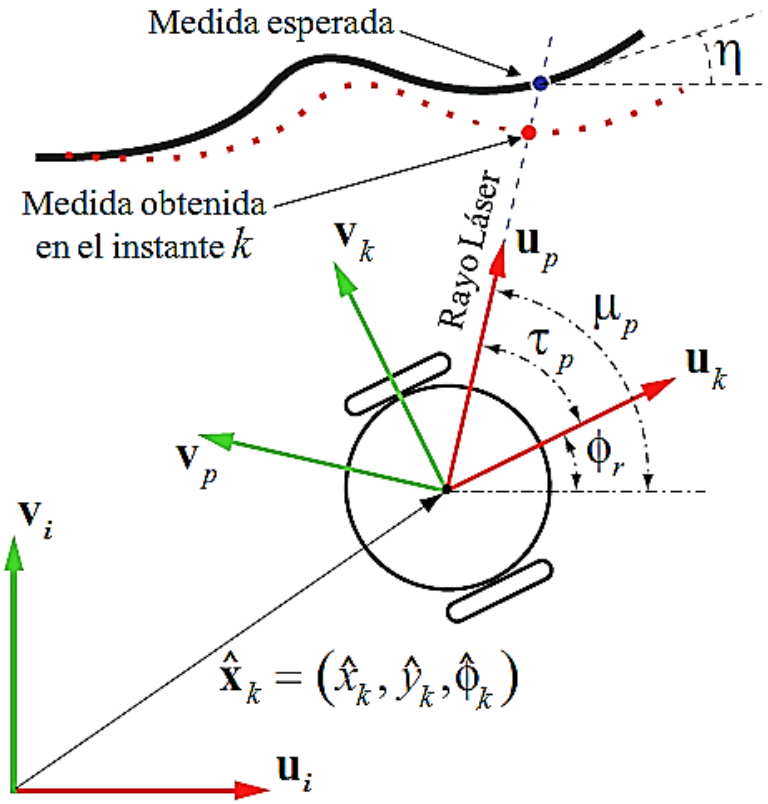

Figura 2.3.3 Modelo de observación.

Capítulo 3

Estrategia Propuesta

En el presente capítulo, se exponen las estrategias utilizadas para cumplir con el objetivo de la tesis. Se inicia con el detalle de las herramientas de software que permiten implementar, visualizar y evaluar el desempeño del sistema desarrollado, dentro de este ámbito, se incluye la especificación y características del entorno de pruebas, el robot y sensor que fueron empleados. Además, se procede a describir a profundidad la utilidad de las curvas B-Splines para atacar el problema de SLAM, debido a que sus propiedades matemáticas, son apropiadamente aplicables para recabar información acerca de los diferentes elementos que se encuentren dentro del área de exploración del robot. También, se explica la aplicación del Filtro de Partículas, que es el método usado en este trabajo para estimar el estado del sistema robótico a medida que realiza el proceso de exploración. Por último, se presenta la estructura de la propuesta que da solución al tema central de la tesis.

3.1 Herramientas de Desarrollo

Es importante realizar una presentación de las herramientas que se incorporan a la solución, de esta manera, se tendrá un panorama más extenso de las aplicaciones y la relevancia del sistema obtenido. Durante esta sección se explicará a detalle el sistema

operativo ROS (Robot Operating System), el simulador de robots Gazebo, el visualizador de datos rviz, el sensor láser Hokuyo y el robot Pioneer P3-DX.

3.1.1 Sistema Operativo Robótico (ROS)

ROS [13] es un marco de trabajo flexible orientado a escribir software para sistemas robóticos. Se conforma de una colección de herramientas, bibliotecas y convenciones cuyo objetivo consiste en simplificar la tarea de generar un comportamiento robótico complejo y robusto para una amplia variedad de plataformas robóticas. ROS es un software de código abierto y opera sobre sistemas basados en Linux, actualmente cuenta con una variedad importante de distribuciones que son actualizadas de manera periódica. ROS fue desarrollado originariamente en 2007 bajo el nombre de **switchyard** por el Laboratorio de Inteligencia Artificial de Stanford, con la intención de dar soporte al proyecto STAIR2. El gráfico de procesos de ROS es una red de procesos **peer-to-peer**, potencialmente distribuidos en máquinas, que se acoplan de una forma sencilla por medio de la infraestructura de comunicación de ROS, siendo su objetivo primario [14] soportar el re-uso de código en la investigación y desarrollo de la robótica.

Este framework es una estructura distribuida de procesos llamados **Nodes** (Nodos), los cuales se pueden agrupar en paquetes y pilas, que fácilmente pueden ser intercambiados, compartidos y distribuidos. Este diseño va desde el nivel del sistema de archivos hasta el nivel comunitario, permitiendo tomar decisiones independientes de desarrollo e implementación, para comprender mejor lo anterior se establecen las siguientes definiciones [2] complementadas con la Figura 3.1.1:

- **Nodo:** Es un proceso de ROS, es el responsable de realizar cálculos computacionales, así como de procesar los datos recibidos por los sensores.

- **Nodo Maestro:** Se encarga de gestionar y coordinar los nodos para que se pueda realizar una correcta comunicación.

- **Servidor de Parámetros:** Es un repositorio multivariado compartido, este se ejecuta dentro del nodo maestro y es usado por los nodos para almacenar y recuperar datos relevantes, llamados parámetros.

- **Mensajes:** Son estructuras de datos simples que constan de campos tipificados (enteros, punto flotante, booleanos, etc.).

- **Temas:** También llamados tópicos, son las vías de transporte con semántica de publicador/suscriptor entre los nodos. Cada tema posee un nombre único, los nodos publican mensajes de un tipo específico y estos pueden ser consultados por otros nodos por medio de una suscripción.

- **Servicios:** El modelo Publicador/Respuesta es realizado mediante servicios, estos son definidos a partir de un par de mensajes, uno para la petición y otro para la respuesta.

- **Bolsas:** Son un formato utilizado para almacenar datos de mensajes de ROS. Es un mecanismo importante para el desarrollo de algoritmos y la realización de pruebas.

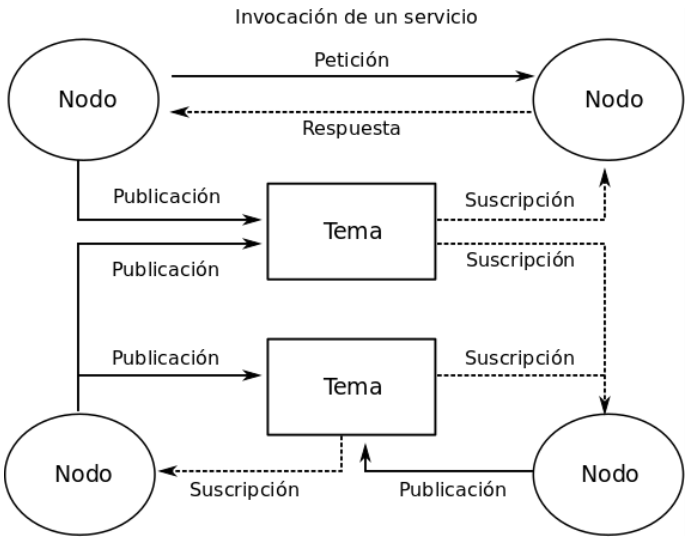

Figura 3.1.1 Diagrama de la comunicación entre nodos en ROS.

ROS tiene características que lo hacen un framework sumamente fuerte y útil, por ejemplo, el código escrito para ROS se puede usar con otros frameworks o estructuras de desarrollo, además es fácil de implementar en cualquier lenguaje de programación, entre los más populares se encuentran Phyton, C++ y Lisp, de igual forma, se cuenta con bibliotecas experimentales para Java y Lua. Como se puede intuir ROS es apropiado para

grandes sistemas de rutinas de ejecución y para grandes procesos de desarrollo, esta es la razón por la que se eligió para este proyecto, aplicando su versión Kinetic.

3.1.2 Gazebo

La simulación robótica es una herramienta esencial cuando se trabaja en proyectos con sistemas robóticos, un simulador bien diseñado permite probar rápidamente algoritmos, diseñar robots, realizar pruebas de regresión y entrenar el sistema de Inteligencia Artificial (IA) utilizando escenarios realistas. Gazebo [15] es un simulador de entornos 3D, cinemático, dinámico y multi-robot, permite realizar simulaciones de robots articulados en entornos complejos, interiores o exteriores, realistas y tridimensionales, de igual manera, brinda la opción de diseñar robots de forma personalizada, crear mundos virtuales usando sencillas herramientas CAD e importar modelos ya creados. Este simulador surge de la necesidad de simular robots en entornos exteriores, bajo diversas condiciones, el desarrollo de Gazebo comenzó en el otoño de 2002 en la Universidad del Ser de California, siendo sus creadores originales el Dr. Andrew Howard y su estudiante Nate Koenig.

Gazebo puede ser sincronizado con ROS, de tal forma que, los robots emulados publiquen la información de sus sensores en nodos, así como implementar una lógica y un control que den ordenes al sistema robótico. Las características más importantes que tiene Gazebo son:

- Es compatible con ROS, se puede ejecutar el simulador gracias a que forma parte de su bundle *“ros-kinetic-desktop-full”*.

- Cuenta con una simulación realista de la física de los cuerpos rígidos. Los robots son capaces de interactuar con el mundo, ya sea por medio de su desplazamiento, al tomar o empujar objetos, y a su vez el mundo interactúa con ellos.

- Alta calidad en los gráficos, que permite desarrollar y visualizar los modelos con un gran nivel de detalle.

- Gazebo tiene la capacidad de desarrollar y simular modelos de robots propios o incluso cargarlos en tiempo de ejecución.

- Como se ha mencionado, el simulador, provee la posibilidad de crear escenarios (mundos) de simulación, variando las características de los contactos con el suelo, los obstáculos e incluso los valores de la gravedad en las tres dimensiones.

- Contiene diversos plugin para añadir sensores al modelo del robot y simularlos, como sensores de odometría, de fuerza, de contacto, lásers y cámaras estéreo.

- Es multiplataforma, de modo que puede ser usado en Unix como en Windows.

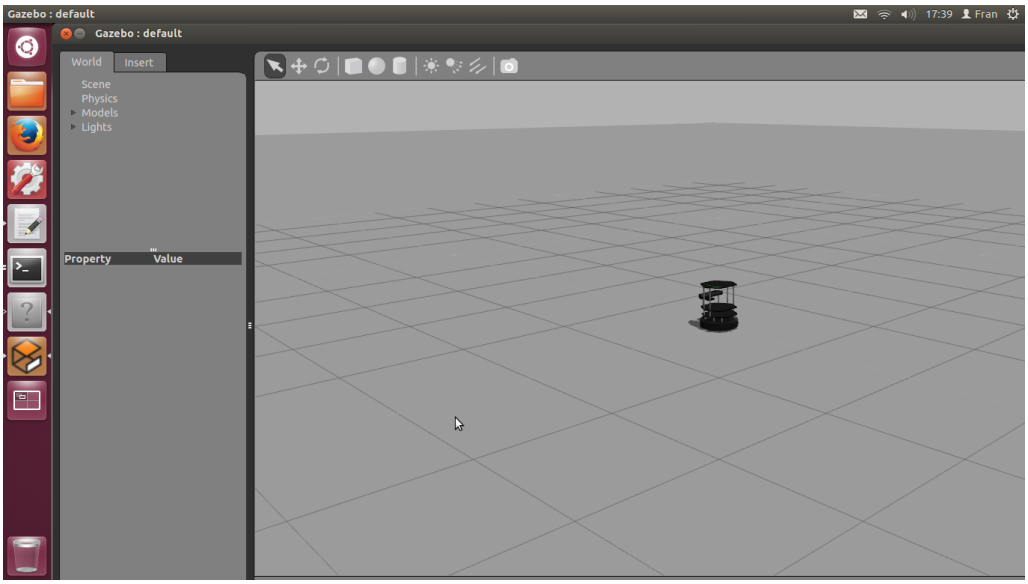

Figura 3.1.2 Simulador Gazebo.

Todos los mundos y modelos son descritos en archivos SDF (Simulation Description Format), el cual es un archivo en formato XML que describe objetos y entornos para simuladores, visualizadores y control de robots. Los componentes de los modelos SDF son [16]:

- **Enlaces (Links):** Contiene las propiedades físicas del cuerpo del modelo, puede ser una rueda o un eslabón de la cadena articular. Cada enlace puede contener muchos elementos visuales y de colisión, la mejor opción es siempre reducir el número de enlaces presentes en el modelo con el fin de mejorar el rendimiento y la estabilidad. A su vez los enlaces pueden conformarse de una serie de elementos.

- **Elemento de Colisión:** Encapsula una geometría que se utiliza para comprobar la colisión. Puede ser desde una forma simple, hasta una malla triangular, un enlace puede contener muchos elementos de colisión.

- **Elemento Visual:** Permite visualizar partes de un enlace, el cual puede contener 0 o más elementos visuales.

- **Elemento Inercial:** Describe las propiedades dinámicas presentes en el enlace, como lo son la masa y la matriz de inercia rotacional.

- **Sensor:** Recopila datos del mundo para el uso de plugins. Un enlace puede contener 0 o más sensores.

- **Elemento Luminoso:** Describe una fuente luminosa conectada al enlace, el cual, puede contener 0 o más luces.

- **Articulaciones (Uniones):** Permite conectar dos enlaces. Se establece una relación padre e hijo junto con otros parámetros como el eje de rotación y los límites de las articulaciones.

- **Plugins:** Es una biblioteca compartida, creada por terceros, que tiene la funcionalidad de controlar un modelo.

En resumen, Gazebo ofrece la posibilidad de simular con precisión y eficiencia poblaciones de robots en entornos complejos, teniendo la capacidad de comunicarse adecuadamente con ROS para la operabilidad del sistema robótico en cuestión.

3.1.3 Rviz

Rviz es una herramienta de visualización en 3D para aplicaciones de ROS [14], donde se pueden representar robots, nubes de puntos, datos de sensores, etc., esta herramienta no solo permite desarrollar herramientas propias, sino que también reutilizar otras ya

existentes. Rviz funciona leyendo e interpretando los diferentes datos contenidos en los mensajes de ROS, por lo cual, es necesario tener un generador externo de estos mensajes, como lo puede ser un robot real o simulado. De igual manera, Rviz usa la información de la librería tf para mostrar el conjunto completo datos que son recopilados por los sensores, visualizar todos estos datos permite analizar rápidamente todo lo que el robot observa y ayuda a identificar problemas como una mala configuración de los sensores o imprecisiones en el modelo.

Si nos centramos a nuestra problemática Rviz nos ayudará a visualizar los datos recabados por el sensor Hokuyo que se encuentra adaptado al robot de prueba, de esta manera, la herramienta podrá detectar los mensajes de ROS pertenecientes al tópico del láser para poder visualizarlos, tal como se muestra en la Figura 3.1.3.

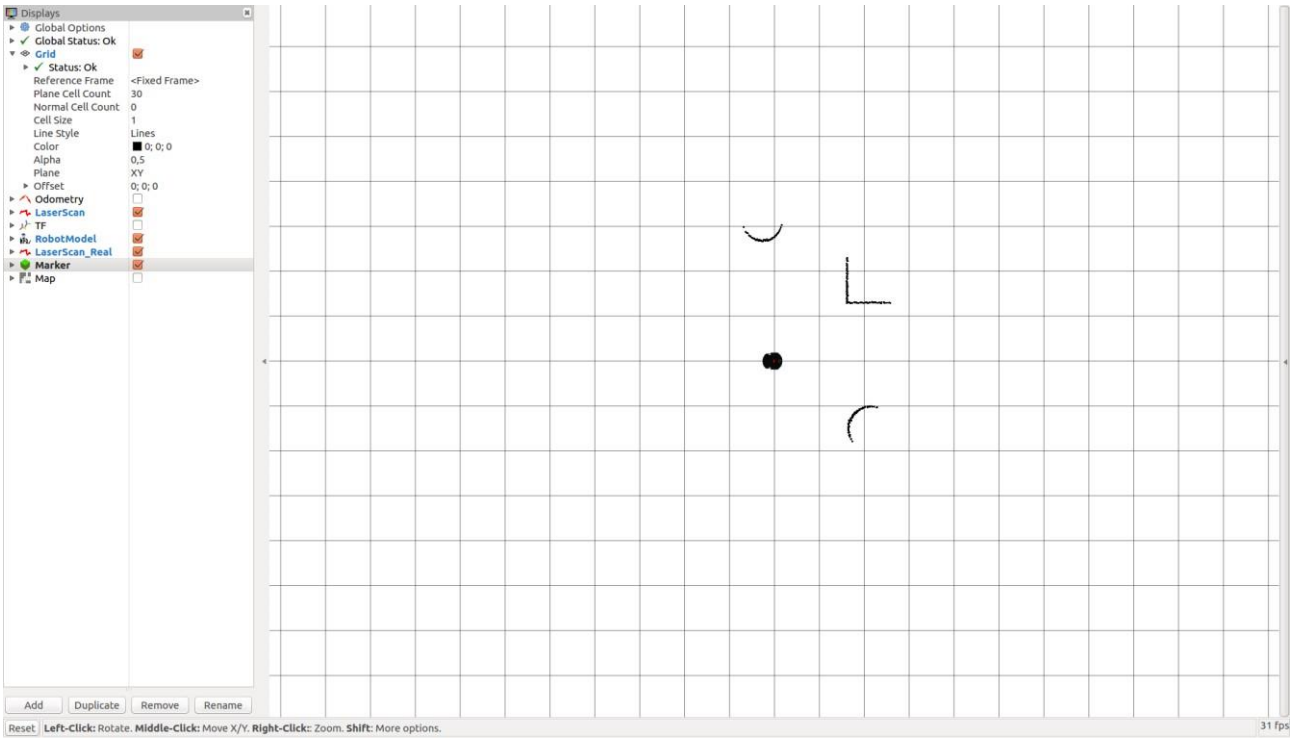

Figura 3.1.3 Visualización de las lecturas del láser Hokuyo en Rviz.

Dentro de Rviz existen do marcos de coordenadas:

- **Marco Fijo:** Es utilizado para denotar el marco del “mundo”, de tal manera, que este marco se utilizará para cada una de las transformaciones que se realicen. Si no se realiza una correcta asignación de este marco es posible que no se visualice ningún tipo de información en Rviz.

- **Marco Objetivo:** Es utilizado como referencia para la vista de la cámara de Rviz, se puede establecer el mapa como referencia y se verá como el robot se desplaza a través de él, en caso contrario, si el robot es el marco objetivo este permanecerá fijo mientras el mapa se mueve.

3.1.4 Robot Pioneer P3-DX

El robot Pioneer P3-DX es un pequeño robot de tracción diferencial de dos ruedas y dos motores, dedicado a la investigación. Es un robot móvil inteligente y es considerado uno de los más populares del mundo en su rama por su gran versatilidad, fiabilidad y durabilidad.

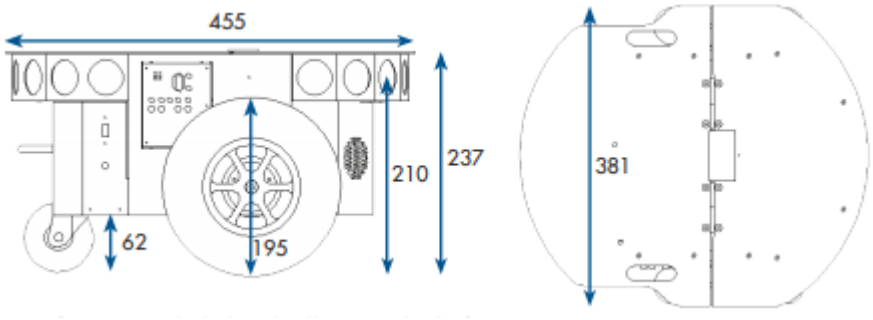

Figura 3.1.4 Dimensiones del robot Pioneer P3-DX (mm).

El robot cuenta de fabrica con un controlador integrado, motores con codificadores de 500 garrapatas, ruedas de 19 cm, cuerpo de aluminio resistente, 8 sensores ultrasónicos orientados hacia adelante, 8 sensores orientados hacia atrás (opcional) y con hasta 3 baterías intercambiables, las especificaciones se pueden observar en la Figura 3.1.4. En cuanto a su movimiento, los robots pioneros pueden realizar rotaciones (giros) hacia la izquierda y derecha, así como, avanzar y retroceder (Figura 3.1.5).

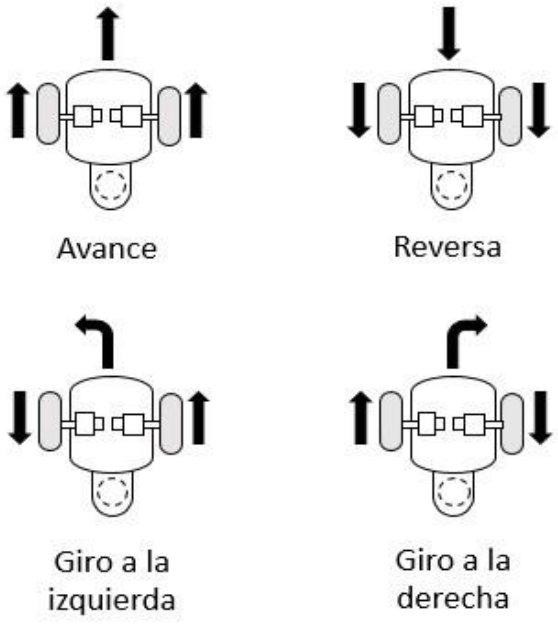

Figura 3.1.5 Especificaciones de movimiento del robot Pioneer P3-DX.

Un aspecto sumamente importante es el correspondiente al modelo cinemático del robot Pioneer P3-DX, ya que esta es una referencia esencial para la implementación de la algoritmia correspondiente. A continuación, se procederá a describir formalmente el modelo para los robots pioneros, tomando como base el trabajo de [2].

Se considera un sistema mecánico con coordenadas generalizadas $q \in C$, donde C es el espacio de configuración en R^n. Una restricción es llamada cinemática cuando solo involucra coordenadas (q) y velocidades (q') generalizadas [17]. En el centro del eje de las ruedas se ubica un punto $P = (P_x, P_y)$, el cual es el origen del marco de coordenadas local para el robot. El vector $X_R = [P_x, P_Y, \theta]$ corresponde a la especificación de la postura del robot dentro del marco de coordenadas global. A través de la forma de Pfaffian, las restricciones cinemáticas son usualmente definidas:

$$v_i(q) \cdot q' = 0 i = 1, \dots, k < n \tag{3.1}$$

Donde v_i corresponde a los k vectores linealmente independientes, los cuales se expresan de la forma:

$$A(q) \cdot q' = 0 \tag{3.2}$$

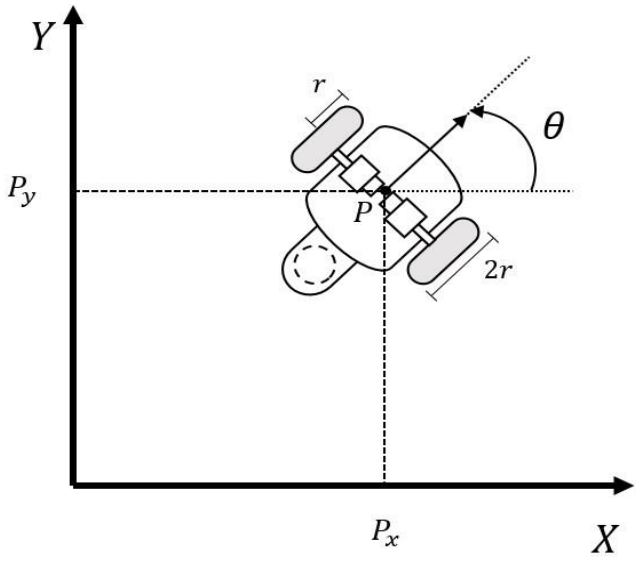

Figura 3.1.6 Diagrama del Robot Pioneer P3-DX (Configuración diferencial).

$A(q) \in R^{(m \cdot n)}$ corresponde una matriz de rango completo, por su parte $S(q) = [s_1, \dots, s_{(n-m)}]$ es un conjunto de vectores linealmente independientes en el espacio nulo de $A(q)$, lo cual se describe como:

$$A(q) \cdot S(q) = 0 \tag{3.3}$$

Si se asume que no existe un desplazamiento lateral y la velocidad de P esta en la dirección del marco local del robot, las matrices $A(q)$ y $S(q)$ que satisfacen la ecuación 3.3 son:

$$A = [-sin\theta cos\theta 0], S(q) \begin{bmatrix} cos\theta & 0 \\ sin\theta & 0 \\ 0 & 0 \end{bmatrix} \tag{3.4}$$

En términos de un vector de pseudo-velocidades $v(t) \in R^{n-m}$, conforme a las ecuaciones 3.2 y 3.3, es posible establecer la ecuación cinemática del robot como:

$$q' = S(q) \cdot v(t) \tag{3.5}$$

En el que $v(t) = [v_r(t), \omega_r(t)]$ se compone de las velocidades lineales y angulares de las ruedas del robot, denotadas por ω_i y ω_d:

$$v_r(t) = \frac{r(\omega_d + \omega_i)}{2} \; \omega_r(t) = \frac{r(\omega_d - \omega_i)}{l} \tag{3.6}$$

3.1.5 Sensor Láser Hokuyo

Al momento se ha definido el sistema operativo que se utilizará para controlar al sistema robótico Pioneer P3-DX, también se mencionó el uso de las herramientas Gazebo y Rviz para la simulación del trabajo de exploración del robot y la visualización de la información obtenida respectivamente. Ahora es importante describir las características del sensor láser Hokuyo modelo URG-04LX-UG01 (Figura 3.1.7), el cual permitirá sondear el área de exploración en busca de obstáculos, si bien los robots pioneros se encuentran equipados con sensores ultrasónicos estos carecen de exactitud en sus lecturas y poseen un menor rango de detección, en comparativa con el sensor Hokuyo. Recordemos que el problema de SLAM online requiere que la detección de obstáculos se realice con la mayor exactitud posible, de tal manera, que no existan problemas que puedan afectar la movilidad del robot o que provoquen alguna colisión con los obstáculos presentes en el entorno que lo rodea.

Figura 3.1.7 Sensor Láser Hokuyo URG-04LX-UG01 montado sobre el robot Pioneer P3-DX.

Las especificaciones técnicas del sensor se presentan a continuación:

Modelo:	URG-04LX-UG01
Fuente de poder:	De 20 a 5,600mm con una amplitud de $240°$.
Precisión:	De 60 1,000mm $\pm$ 30mm. A partir de 1,000 $\pm$ 3% de la medida.
Resolución angular:	Aproximadamente 0.36 $\left(\frac{360°}{1,024}\right)$.
Tiempo de escaneo	100ms/escaneo.
Peso	Aproximadamente 160g.

3.2 SLAM basado en curvas B-Splines

En el capítulo 2, se presentaron aportaciones importantes que se han realizado para atacar el problema de SLAM, de igual manera, se introdujo el concepto de curvas B-Splines, y como con este tipo de curvas hacen posible el especificar con mayor detalle los obstáculos que se encuentran presentes en el entorno mientras el robot lo explora. El siguiente paso, consiste en establecer la manera en que las curvas B-Splines se adaptan a la solución de SLAM desarrollada, para ello, es importante conocer sus fundamentos matemáticos. Posteriormente, se detallará la manera en que se incorporan dentro de un plan de navegación para dar solución al problema de localización y mapeo (SLAM), incluyendo en el proceso el Filtro de Partículas para estimar la posición del robot en medida que este se desplaza.

3.2.1 Especificación de curvas

En [18] se define una curva como la representación gráfica de una función, a su vez una función polinómica de grado nth se define como una serie de n potencias de una o más variables multiplicadas por coeficientes constantes (Ecuación 3.7).

$$f(x) = C_0X^N + C_1X^{N-1} + C_0X^{N-2} + \cdots + C_0X^0$$

Esto demuestra que el grado del polinomio es el valor del exponente más alto. Si se simplifica la definición de una curva polinómica se tiene que, es un gráfico de una función polinómica. Las curvas son una representación muy útil aplicadas en diferentes contextos. Por ejemplo, se pueden utilizar para definir rutas para objetos en un videojuego o también para describir funciones de velocidad, color o en nuestro caso definir atributos de objetos.

Los elementos necesarios para la construcción de curvas B-Splines son:

- **Puntos de control:** Es un conjunto de puntos que controlan la forma general de la curva.

- **Grado y orden:** Las curvas que poseen grados más altos tienen tramos que están influenciados por más puntos de control y por lo tanto son aproximaciones más suaves del polígono de control (Figura 3.2.1).

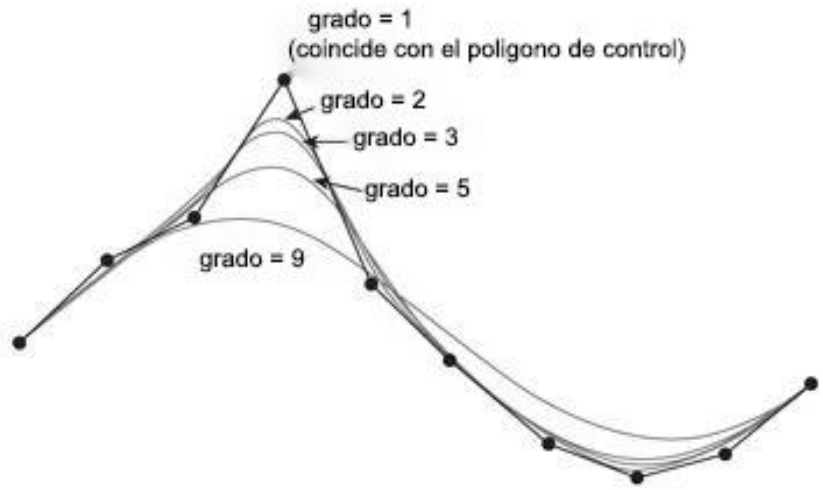

Figura 3.2.1 Comportamiento de curvas de varios grados.

Las B-Splines desacoplan el número de puntos de control N del grado de la curva $k-1$ y el rango de efecto de los puntos de control k. De esta manera, cada punto de la curva solo se ve afectado por un subconjunto de los puntos de control en lugar de cada punto de control (como en las curvas de Bezier), con esto se logra un mayor control local respecto a la forma de la curva. El orden k determina el número de puntos de control que afectan a subconjuntos de puntos de curva, por lo cual, el orden máximo de una curva es $k = N$, y el grado máximo de una curva es $N-1$. Respecto al orden mínimo es 2, lo que crea una curva de primer grado.

- **Vectores de nudos**: El propósito de un vector de nudos es describir el rango de influencia para cada uno de los puntos de control. Si se desea dibujar una curva de tercer orden que contenga cinco puntos de control los rangos de cada punto de control se describirían como $[t_0, t_3], [t_1, t_4], [t_2, t_5], [t_3, t_6], [t_4, t_7]$, transformándolo a un vector el resultado sería $[t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7]$, esto es un vector de nudos. En este caso de muestra, el vector de nudo se utiliza para describir cómo los puntos de control influyen en una curva de tercer orden con un rango paramétrico de $t_0 < t\ t_7$.

  Los elementos del vector de nudo deben ser monótonamente crecientes. Esto significa que cada elemento posterior deber ser mayor o igual que el anterior. Existen dos tipos definidos de vectores de nudo, los uniformes donde los elementos del vector se espacian uniformemente (Ejemplo vector 3.7) o los no uniformes, donde sucede lo contrario (Ejemplo vector 3.8).

$$X = [1\ 2\ 3\ 4\ 5\ 6] \tag{3.7}$$

$$X = [1\ 2\ 2\ 3\ 3\ 4] \tag{3.8}$$

- **Funciones base B-Splines:** Las funciones base para curvas B-Splines están definidas por las fórmulas de recursión de Cox-de Boor (Ecuación 3.9).

$$N_{i,1}(t) = \begin{cases} 1\ si\ x_i \le t \le x_{i+1} \\ 0\ De\ otra\ forma \end{cases}$$
$$N_{i,k}(t) = \frac{(t - x_i)N_{i,k-1}(t)}{x_{i+k-1}} + \frac{(x_{j+k} - t)N_{j+1,k-1}(t)}{x_{j+k} - x_{j+1}} \tag{3.9}$$

  Las funciones base resultante están sujetas a la restricción de que la suma de las funciones base debe ser igual a 1 (Ecuación 3.10).

$$\sum_{i=1}^{N} N_{i,k}(t) \equiv 1 \tag{3.10}$$

  Ahora se debe encontrar recursivamente las funciones base para cada orden dado, comenzando en las funciones base de primer orden. Al obtener el conjunto

de funciones base y conocer el grado de la curva se puede adaptar la ecuación 3.10 para generar la ecuación de la curva B-Spline (Ecuación 3.11)

$$P(t) = \sum_{i=1}^{N} P_i N_{i,k}\,(t) \tag{3.11}$$

Las funciones base definen la influencia de cada punto de control como una función de t. Una vez que se ha derivado las funciones base necesarias, la ubicación del punto en la curva es simplemente la suma ponderada de todos los puntos de control en un valor dado de t. Sólo que ahora, el peso de algunos de los puntos de control podría ser 0.

3.2.2 Segmentación de datos y extracción de puntos de referencia

Durante la sección 2.3 se realizó la introducción al trabajo realizado por el Dr. Pedraza, el cual, incorporo las curvas B-Splines al problema de localización y mapeo (SLAM). Se estableció la manera en que a través de un sensor láser un robot puede obtener información de su entorno y utilizarla para ubicarse. Sin embargo, antes de que los datos obtenidos puedan utilizarse por el algoritmo de localización es necesario, que dicha información, sea tratada por varios procesos.

3.2.2.1 Primera Segmentación

Para este proceso se realiza un análisis de la posición relativa de los puntos de datos. El objetivo consiste en detectar puntos que sean lo suficientemente cercanos como para pertenecer al mismo obstáculo. Si el conjunto de m puntos detectados por el láser presenta un punto de rompimiento se analiza con el criterio de Dietmayer y dependiendo la evaluación se generan subconjuntos de puntos denominados clústers. El criterio de Dietmayer se detalla a continuación.

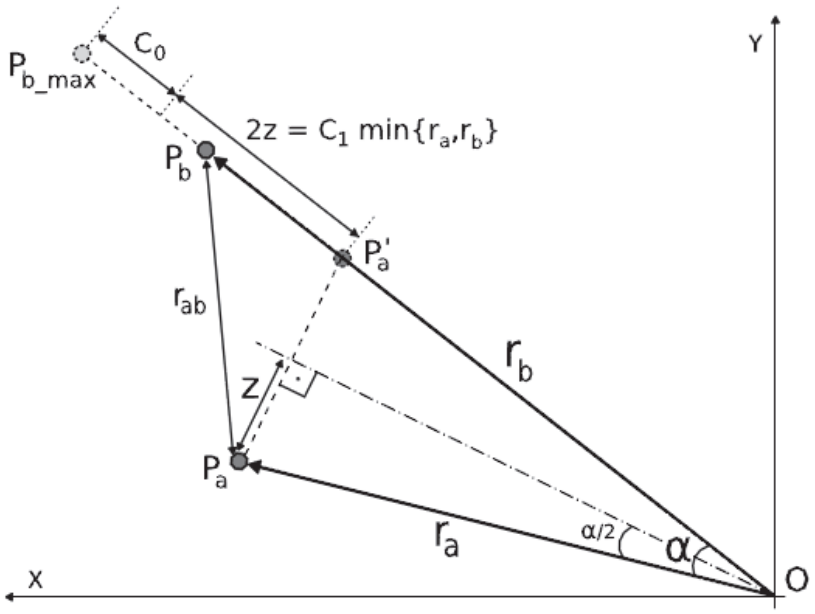

Figura 3.2.2 Criterio de Dietmayer.

P_a y P_b representan dos puntos consecutivos detectados por el láser, mientras que r_ay r_b son las distancias de estos puntos al origen de coordenadas. Dado el triángulo OP_aP_b donde r_ay r_b son conocidos y representan la resolución angular del láser, se aplica el teorema de cosenos para obtener la distancia existente entre P_a y P_b:

$$r_{ab} = \sqrt{r_a^2 + r_b^2 - 2r_a r_b \cos(\alpha)} \tag{3.12}$$

De esta forma, los clústers son formados a partir de la evaluación de la distancia P_a y P_b:

$$r_{ab} \leq C_0 + C_1 \cdot min\{r_a - r_b\} \tag{3.13}$$

Donde C_0 representa el ruido asociado al ajuste en las medidas del láser, para el sensor Hokuyo este valor fue presentado en su ficha técnica dentro de la sección 3.1.5. Por otra parte, $C_1 = \sqrt{2(1 - \cos(\alpha))}$, este valor no es tan explicado por Dietmayer, sin embargo, se puede comprenderse, al igual que $min\{r_a - r_b\}$, usando la figura 3.2.2, donde se observar que el valor de $min\{r_a - r_b\}$ es igual al valor de r_a, de tal manera que:

$$C_1 = min\{r_a - r_b\} = r_a\sqrt{2(1-\cos(\alpha))} = 2r_a\sqrt{\frac{1-\cos(\alpha)}{2}} \tag{3.14}$$

Finalmente obtenemos:

$$2z = C_1 \cdot r_a \tag{3.15}$$

De tal forma, si la desigualdad 3.13 se cumple entonces una nueva segmentación se realiza, generando nuevos clústers. Ahora con los datos sin procesar en los clústers, el siguiente paso es encontrar las características contenidas en cada uno de ellos.

3.2.2.2 Segunda Segmentación

Los segmentos obtenidos durante en la primera segmentación se someten a otra evaluación, en ella se busca detectar ángulos, formados por pares consecutivos de puntos, que se encuentren debajo de cierto umbral. El objetivo de la segunda segmentación es detectar esquinas y curvas que presenten curvaturas altas, para realizar esta evaluación se hace uso del algoritmo “Split and merge”. El algoritmo se conforma de dos partes, la primera de ellas es recursiva, y consiste en dividir los segmentos disponibles en segmentos más pequeños, mientras que la segunda se utiliza para fusionar segmentos que son casi colineales (Figura 3.2.3).

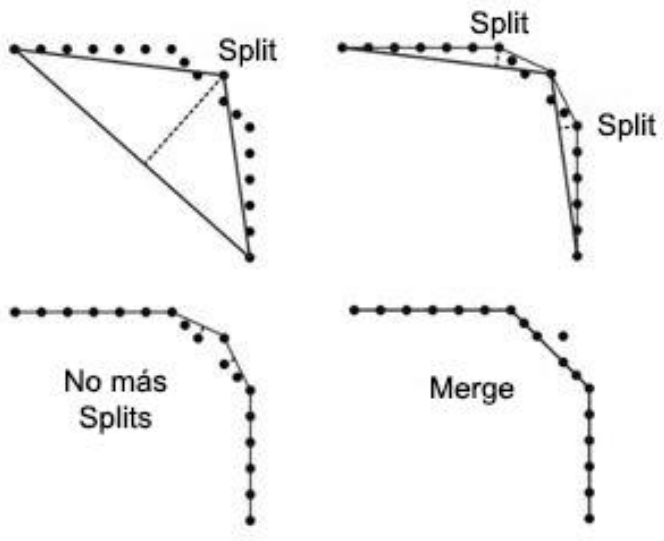

Figura 3.2.3 Comportamiento del algoritmo “Split and Merge”.

Al finalizar el algoritmo cada cluster procesado generará n sub-clústers de datos en bruto, correspondientes a las medidas recabadas por el láser, de n segmentos no-colineales. La línea contenida en estos subgrupos se aproxima utilizando el método de la media de mínimos cuadrados. La Figura 3.2.4 especifica el proceso que sigue el algoritmo de “Split and Merge”.

Algoritmo “Split and Merge”

$función\ \boldsymbol{SPLIT}(\boldsymbol{P})$
  $\textbf{Entrada}: \mathrm{P}\{p_1, \dots, p_n\}$
  $Colocar\ P\ en\ la\ lista\ L$
  $R \leftarrow línea\ que\ se\ ajusta\ a\ los\ puntos\ de\ P$
  $p_t \leftarrow punto\ con\ la\ mayor\ distancia\ T\ a\ la\ línea\ R$
  $Si(T > T_{max})\ entonces$
    $P' \leftarrow \{p_l, \dots, p_t\}$
    $P'' \leftarrow \{p_t, \dots, p_n\}$
    $L \xleftarrow{add} \textbf{SPLIT}(\mathrm{P}')$
    $L \xleftarrow{add} \textbf{SPLIT}(\mathrm{P}'')$
  $Fin\ Si$
  $\boldsymbol{Regresar\ L}$

$\boldsymbol{funci}\text{ó}\boldsymbol{n}\ \boldsymbol{MERGE}(\boldsymbol{L})$
  $Entrada: L \leftarrow lista\ de\ conjuntos\ \{P_i, \dots, P_n\}$
  $Para\ i \leftarrow 1\ to\ n - 1\ hacer$
    $Si\ P_i y\ P_{i+1}\ son\ colineales\ dentro\ de\ un\ margen\ de\ error\ entonces$
      $Q \leftarrow P_i \cup P_{i+1}$
      $R \leftarrow línea\ que\ se\ ajusta\ a\ los\ puntos\ de\ Q$
      $q_t\ punto\ \in Q\ con\ la\ mayor\ distancia\ T\ a\ la\ línea\ R$
      $Si\ T \leq T_{max}\ entonces$
        $L \xleftarrow{eliminate} P_i\ y\ P_{i+1}$
        $L \xleftarrow{add} Q$
      $Fin\ Si$
    $Fin\ Si$
  $Fin\ Para$
  $\boldsymbol{Regresa\ L}$

Figura 3.2.4 Algoritmo “Split and Merge”.

Al finalizar el algoritmo de “Split and Merge”, se procede a analizar el valor de la pendiente de cada segmento, este proceso se realiza en pares, si la diferencia entre las dos pendientes es mayor a un determinado umbral, un nuevo proceso de segmentación se realiza.

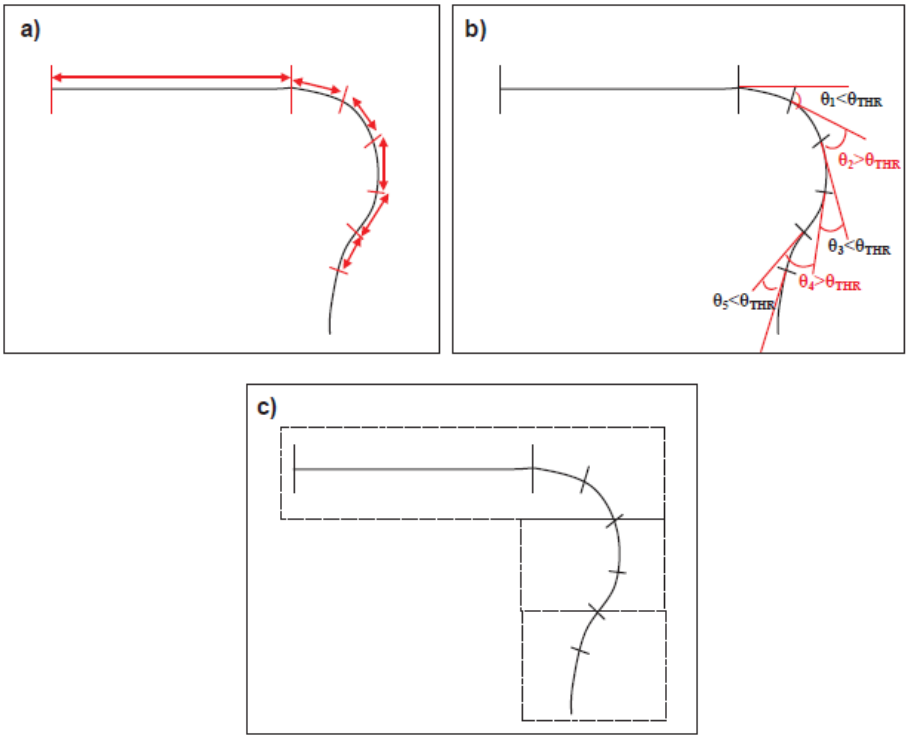

Figura 3.2.5 Segunda Segmentación.

En el inciso $a)$ de la Figura 3.2.5 se visualizan los segmentos de línea obtenidos al ejecutar el algoritmo de “Split and Merge”, en el inciso $b)$ corresponde al análisis de las pendientes de los segmentos adyacentes y el inciso $c)$ es el resultado final esperado de la segunda segmentación. El siguiente paso al finalizar la segunda segmentación corresponde a que cada uno de los segmentos obtenidos sean ajustados a curvas B-Splines de tercer grado. De esta forma, se definen los obstáculos del entorno dentro de la observación del robot en un instante k.

Con la descripción anterior, se tiene que el proceso de segmentación y extracción de puntos de referencia, consiste en que el robot Pioneer P3-DX realice una observación a través del sensor Hokuyo obteniendo información (segmentos de puntos), que pasa a la primera segmentación, donde con ayuda del criterio de Dietmayer, se generan nuevos segmentos a partir de la distancia que exista entre los puntos de la observación. A su vez estos segmentos resultantes son tratados con el algoritmo de “Split and Merge” que a través

de la evaluación de las pendientes de segmentos adyacentes pueden producir una nueva segmentación, generando los segmentos finales correspondientes a los obstáculos detectados por el sensor. Finalmente, estos segmentos son ajustados a curvas B-Splines de tercer grado representando correctamente los obstáculos presentes en el entorno que explora el robot (Figura 3.2.6).

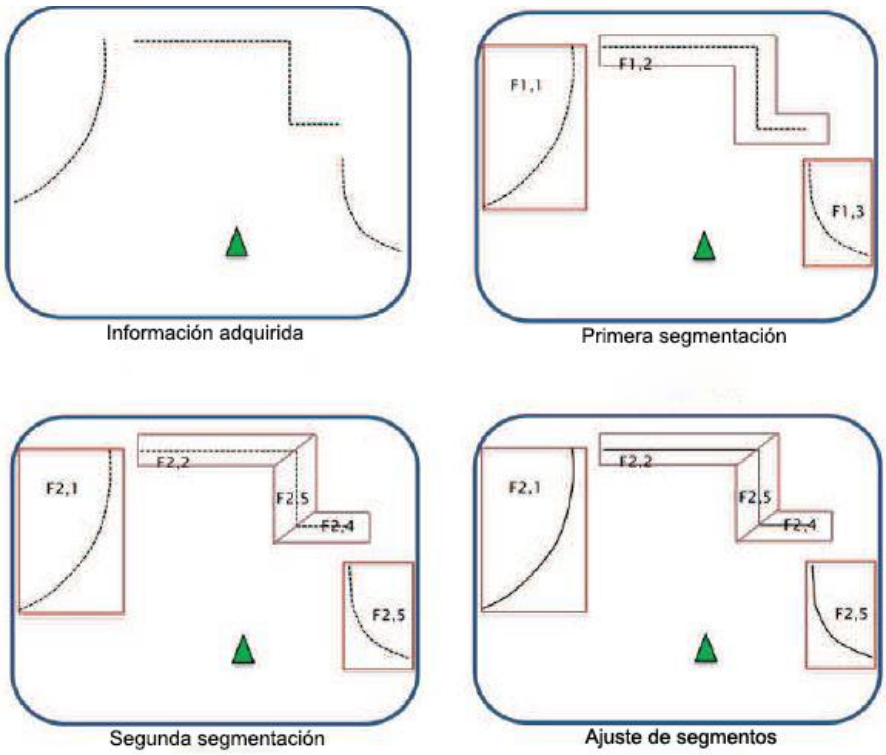

Figura 3.2.6 Proceso de segmentación de datos y extracción de puntos de referencia.

3.2.3 Asociación de puntos de referencia

En este momento la información proporcionada por el sensor ha sido segmentada, recordemos dicha información es tratada en cada observación, por lo cual, después de realizar el proceso de segmentación de datos el siguiente paso es asociar los segmentos resultantes. El objetivo de este procedimiento es igualar las características observadas de diferentes escaneos y asignar medidas que pueden originar y rechazar mediciones falsas. De esta manera, los segmentos que pertenezcan a un mismo obstáculo, pero que fueron observados en momentos de tiempo diferentes pueden unirse en un solo segmento en la representación general del entorno, este proceso es conocido como re-observación de puntos de referencia.

La primera asociación se realiza de una manera “burda”, los puntos de control de los segmentos obtenidos en el proceso de segmentación se comparan con los puntos de control contenidos en el mapa bajo el criterio 2.12 presentado en la sección 2.3.2. Donde $X_{m,i}$ y $X_{o,j}$ son los puntos de control de las Splines en el mapa y en la posición prevista respectivamente, n_m y n_0 son el número de puntos de control de las Splines tanto del mapa como en la posición prevista, por su parte, $dist(X_{m,i}, X_{o,j})$ representa la distancia Euclidiana entre los puntos de control, d_{min} es el parámetro que regula si los puntos se encuentran relacionados o no. Si no existe una Spline suficientemente cerca en el mapa, entonces el objeto observado es agregado directamente al mapa, una vez que la posición del robot ha sido localizada. En caso contrario, si la Spline se asocia con una característica del mapa, se aplican los pasos uno y dos (Figura 2.3.2).

Dentro de la ejecución del método se pueden observar tres tipos de características **picos**, **líneas rectas** y **esquinas**, siendo los picos y las equinas las características más fáciles de asociar, ya que solo se buscan características almacenadas en el sistema donde la distancia Euclidiana entre la característica observada y la que se encuentra almacenada sea menor que un umbral d_{min}. Para las líneas rectas es necesario transformarlas a un punto fijo, para ello, se toma la posición dada por el robot y se calcula el punto ortogonal a la línea recta, el proceso se realiza tanto para la línea observada como a la contenida en el mapa (Figura 3.2.7).

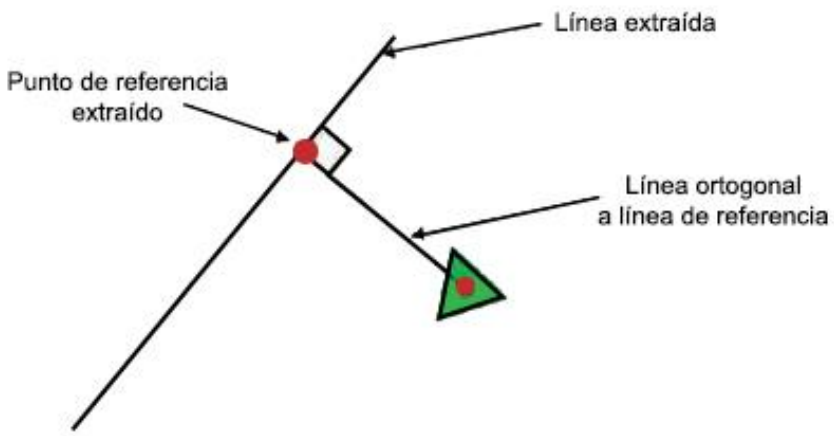

Figura 3.2.7 Característica de línea recta a punto.

De esta manera, las características obtenidas en cada observación se incorporan al mapa correspondiente a la exploración del robot. Hasta este punto el robot capta información de su entorno la cual pasa al proceso de segmentación de datos y posteriormente a la asociación generando información precisa del entorno del robot que tiene que ser llevada al proceso de localización.

3.2.4 Método Sensor-based Random Tree (SRT)

Un árbol aleatorio de exploración basado en sensores, SRT (Sensor-based Random Tree), es un método de exploración de ambientes desconocidos presentado por Oriolo, Vendittelli, Freda y Torso en [19], el cual representa el roadmap del área explorada asociando una región segura (Safe Region, SR), para esto, cada nodo del árbol (Tree, T) consiste en una posición adoptada por el robot con su región segura local asociada (Local Safe Region, LSR) construida a partir del sistema de percepción del robot. El proceso de localización se realiza a través de la comparación y asociación de las características presentes en las curvas B-Splines, extraídas del ambiente que explora el sistema robótico.

3.2.4.1 Fundamentos de SRT

El problema de SLAM al cual nos hemos referido a lo largo del presente trabajo de tesis, parte de las técnicas de planificación de movimientos aleatorios (PMA), donde el problema planteado es el aprendizaje activo de orden-libre [2], el cual describe que: lo que observa el robot de su ambiente se encuentra disponible, sin embargo, depende solo de la última acción que este realiza (búsqueda aleatoria). Lo anterior nace a raíz de que el mapa del ambiente se genera de manera progresiva a la par de que el robot se desplaza recopilando información, de esta manera, los planificadores aleatorios se consideran estrategias de exploración que se encuentran orientados a objetivos utilizando la selección aleatoria de acciones. En la actualidad, muchas de las técnicas caen dentro de la clase de exploración basada en fronteras, donde el robot se mueve hacía los límites (de dichas fronteras) establecidos por las SR exploradas y el entorno por explorar, con el objetivo de obtener nuevas observaciones que provean de características nuevas en el ambiente, el problema central de la exploración es cómo seleccionar la siguiente acción a ejecutar.

Una solución que se ha adoptado para atacar las problemáticas presentadas han sido los métodos de exploración basados en sensores, esto fue presentado en [20], este método procede de la construcción de una estructura de datos llamada SRT, cada nodo de esta estructura consiste de una configuración q libre de colisión que ha sido alcanzada por el robot, al igual, se incorpora la descripción de la región segura S circuncidante a q que es percibida por los sensores. La construcción del árbol se realiza en medida de que el robot explora el ambiente, al generarse nuevos nodos el árbol se extiende, el algoritmo de construcción del SRT base que se utilizó se presenta en la Figura 3.2.8.

Algoritmo de construcción de SRT base

Entrada: $q_{init}, K_{max}, I_{max}, \alpha, d_{min}$

1. $q_{curr} = q_{init}$
2. ***for*** $k = 1\ to\ K_{max}\ do$
3. $S \leftarrow PERCEPTION(q_{curr})$
4. $ADD(T, (q_{curr}, S))$
5. $i = 0$
6. ***do***
7. $\theta_{rand} \leftarrow RANDOM\_DIR()$
8. $r \leftarrow RADIO(S, \theta_{rand})$
9. $q_{cand} \leftarrow DISPLACE(q_{curr}, \theta_{rand}, \alpha \cdot r)$
10. $i = i + 1$
11. ***While*** $(VALID(q_{cand}, d_{min}, T)\ OR\ i = I_{max})$
12. ***if*** $(VALID(q_{cand}, d_{min}, T)\ then$
13. $MOVE\_TO(q_{cand})$
14. $q_{curr} \leftarrow q_{cand}$
15. ***else***
16. $MOVE\_TO(q_{curr}.parent)$
17. $q_{curr} \leftarrow q_{curr}.parent$
18. ***end***
19. ***return*** T

Figura 3.2.8 Algoritmo de construcción de SRT base.

En cada iteración k del algoritmo, el ambiente es sensado y se recopilan datos, de esta manera, se obtiene la región S que estima el espacio libre circuncidante al robot en la configuración q_{curr}, a la cual se asocia su LSR, esto contiene un nodo del árbol T. La forma de representar S en la estructura SRT dependerá de la estrategia de percepción.

La función $RANDOM\_DIR()$ genera una dirección aleatoria de exploración θ_{rand} y la función $RADIO$ calcula el radio r de S en la dirección de exploración. A continuación, se genera una configuración candidata q_{cand} la cual se obtiene a partir de la longitud $\alpha \cdot r$ en dirección a θ_{rand}. Mientras tanto la función $VALID$ determina que q_{curr} este a una distancia mayor a d_{min} y la cual no debe situarse en la LSR de otra configuración previa en T. Si la validación es correcta el robot se desplaza de q_{curr} a q_{dest} y así el ciclo se repite. En caso de que la validación no sea exitosa se considera la exploración de otras configuraciones aleatorias partiendo del nodo actual q_{curr}, si ninguno de los nodos satisface la validación o se exceden un número máximo de i intentos q_{dest} se define como $q_{curr}.parent$.

En [3] se presentan tres estrategias de SRT, cuya diferencia radica en la forma de manejarlas:

- **SRT-Ball:** Considerada una técnica conservadora para usar sensores con ruido (Figura 3.2.9 a).

- **SRT-Star:** En comparación con SRT-Ball es menos conservadora, y tiene un mayor índice de confianza en la información reportada por los sensores (Figura 3.2.9 b).

- **SRT-Radial:** En esta técnica la forma de la LSR se deformará, y el radio (distancia desde el robot al borde de la LSR) será diferente dependiendo la dirección (Figura 3.2.9 c).

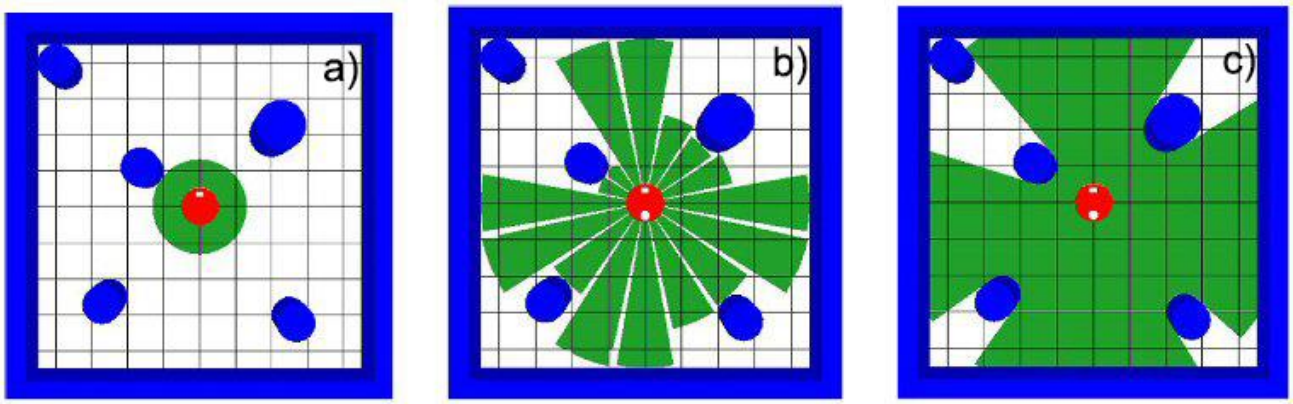

Figura 3.2.9 Estrategias de SRT (Ball, Star y Radial)".

3.2.4.2 Estrategia de SRT aplicada

Dentro del trabajo de tesis se incorporó el algoritmo de exploración integrada basado en SRT, para definir los procesos referentes a la exploración del robot, incluyendo los asociados al tratamiento de las características del entorno utilizando curvas B-Splines. En cada iteración del algoritmo el primer paso consiste en obtener la LRS asociada a la configuración actual del robot, q_{curr}. Una vez que se obtiene se ejecuta la función $EXTEND\_TREE$, la cual tiene la responsabilidad de actualizar el árbol añadiendo un nuevo nodo y su LRS, además el vector que contiene los puntos de control de las curvas que constituyen el entorno será actualizado con los puntos de control de las curvas contenidas en la LSR que todavía no forma parte de él. A continuación, se procesa la frontera local F, donde se identifican los obstáculos y áreas libres, F es una colección de arcos discretos, después de haber definido los límites y en caso de existir zonas libres se ejecuta el proceso $RANDOM\_DIR$ el cual genera direcciones aleatorias que puede seguir el sistema robótico dentro de un arco libre, a partir de esto, una configuración q_{cand} se genera, para esto el robot debe ajustarse en dirección θ_{rand}, que será la orientación que tomará el robot para dirigirse a la dirección aleatoria elegida, dado que, el punto se encuentra en un arco libre de obstáculos q_{cand} estará libre de colisión. En caso de no encontrar un arco de frontera libre, el robot volverá a la posición del nodo padre, se generará un nuevo q_{curr} y el ciclo de exploración comenzará de nuevo.

Una vez obtenida la configuración q_{cand} se pasa al proceso $VALID\_CONF$ el cuál evaluará la configuración y determinará si es válida o no, al igual, establecerá si esta fuera de la LSR de otros nodos del árbol. Al comprobarse que la configuración es válida, se establecerá

como la configuración destino para el robot q_{dest}, por otra parte, si la validación no es posible después de un número de i intentos, el nodo padre q_{cand} será la nueva configuración q_{dest}, volviendo de esta forma a la configuración padre. Para el movimiento del robot se aplicará la función $MOVE\_TO(q_{curr}, q_{dest})$ (Figura 3.2.10), en primer lugar, se buscará en una lista previa de entradas de control $List\_U$, una entrada que aproxime la posición del robot a la posición q_{dest} desde la posición q_{curr}, llamada $u_{control}$. Una vez definido $u_{control}$ se aplicará al robot, lo que tendrá como resultado el desplazamiento del sistema robótico cerca de la meta. Para este punto, se realiza una actualización de la posición odométrica del robot que tenía almacenada en memoria añadiendo el aumento en (x, y, θ) que se generó en su desplazamiento, esta información reportada por el sistema robótico es esencial para obtener la posición detectada por el método de localización basado en características con B-Splines, la importancia de la posición recae también en la forma en que se asocian los elementos del entorno a cada nodo de la exploración. El algoritmo se repetirá hasta que las configuraciones q_{curr} y q_{dest} sean las mismas, cuando ese caso suceda el robot habrá conseguido llegar a la meta, marcando el fin del proceso de exploración, el algoritmo se presenta en la Figura 3.2.11.

$MOVE\_TO(q_{curr}, q_{dest}, ENV\_BS)$

1. **while** $q_{curr} \neq q_{dest}$
2. $\quad u_{control} \leftarrow \boldsymbol{BEST\_U}(List\_U, q_{curr}, q_{dest})$
3. $\quad ROBOT \leftarrow u_{control}$
4. $\quad \hat{q} \leftarrow \boldsymbol{ODOMETRY}$
5. $\quad q_{curr} \leftarrow \boldsymbol{LOCALIZATION}(\hat{q}, q_{curr}, ENV\_BS)$
6. **end**
7. **return** q_{curr}

Figura 3.2.10 Función "MOVE_TO".

Algoritmo de Exploración Integrada basado en SRT (q_{init}, K_{max})

1. $q_{curr} \leftarrow q_{init}$
2. $\boldsymbol{for}\ k = 1\ \boldsymbol{to}\ K_{max}$
3. $S \leftarrow LSR(q_{curr})$
4. $T \leftarrow EXTEND\_TREE(q_{curr}, S, T)$
5. $ENV\_BS \leftarrow UPDATE\_BSPLINES(S)$
6. $F \leftarrow FRONTIER(q_{curr}, S)$
7. $\boldsymbol{if}\ F \neq 0$
8. $i = 0$
9. $VALID \leftarrow FALSE$
10. $\boldsymbol{While}\big((i < MAX_{ITER})\ \&\&\ (¡VALID)\big)$
11. $\theta_{rand} \leftarrow RANDOM\_DIR$
12. $q_{cand} \leftarrow DISPLACE(q_{curr}, \theta_{rand})$
13. $VALID \leftarrow VALID\_CONF(q_{cand})$
14. $i++;$
15. $\boldsymbol{end}$
16. $\boldsymbol{if}(VALID)$
17. $q_{dest} \leftarrow q_{cand}$
18. $\boldsymbol{else}$
19. $q_{dest} \leftarrow q_{curr}.parent$
20. $\boldsymbol{if}\ q_{dest} = NULL$
21. $\boldsymbol{return}\ [T, ENV_{BS}]$
22. $\boldsymbol{end}$
23. $\boldsymbol{end}$
24. $\boldsymbol{else}$
25. $q_{dest} \leftarrow q_{curr}.parent$
26. $\boldsymbol{if}\ q_{dest} \leftarrow NULL$
27. $\boldsymbol{return}\ [T, ENV_{BS}]$
28. $\boldsymbol{end}$
29. $\boldsymbol{end}$
30. $MOVE\_TO(q_{dest}, q_{curr}, ENV\_BS)$
31. $q_{curr} \leftarrow q_{dest}$
32. $\boldsymbol{end}$
33. $\boldsymbol{return}\ [T, ENV\_BS]$

Figura 3.2.11 Algoritmo de Exploración Integrada basado en SRT.

Como se mencionó el proceso de localización se apoya de las curvas B-Splines que definen los obstáculos presentes en el ambiente, permitiendo obtener y representar las fronteras existentes en las regiones libres del robot. De esta manera, se obtiene una posición estimada del sistema robótico, añadiendo a la última posición los incrementos de posición de Δx, Δy y $\Delta\theta$ realizados por el robot a la hora de desplazarse, de este modo, se realiza la construcción del mapa del ambiente explorado, al finalizar el proceso el robot ha recabado las características necesarias para definir el entorno explorado y los obstáculos encontrados en él.

3.3 Detalles de la implementación

Hasta el momento, se han presentado los diferentes procesos que conforman la solución propuesta para atacar el problema de SLAM, incorporando las curvas B-Splines. La estrategia se ha detallado a un nivel teórico, explícitamente de los algoritmos presentes en cada proceso, sin embargo, se presentaron detalles que se trataron fuera de los conceptos algorítmicos presentados y los cuales se presentan en este capítulo.

Principalmente, hablaremos de los paquetes de ROS utilizados para el control del robot Pioneer y el sensor láser Hokuyo:

- **RosAria:** Es un paquete de ROS que provee una interfaz para la mayoría de los robots fabricados por Adept MobileRobots, MobileRobots Inc., ActivMedia, entre los cuales, se encuentra el robot Pioneer P3-DX [21].

- **Urg_node:** Este paquete permite la conexión del sensor láser Hokuyo con ROS, así mismo, la transmisión de los datos de las lecturas mediante su publicación en un tema [22].

- **tf:** Paquete que permite dar seguimiento a diferentes marcos de coordenadas a lo largo del tiempo. Mantiene la relación entre los marcos de coordenadas en una estructura de árbol con un búfer de tiempo, lo que permite a los usuarios realizar transformaciones de puntos, vectores, etc., entre dos marcos de coordenadas en cualquier momento.

3.3.1 Odometría

Dentro de los comportamientos que presenta el robot dentro de SLAM uno de los más complejos es el de navegación, en el cual, se le permite al robot navegar desde una posición actual q_{curr} a una posición deseada q_{dest}, en la cual después de i iteraciones q_{dest} deberá ser igual a la configuración meta. Dentro de [23], se hace referencia que para que este proceso se lleve a cabo se utilizan tres estados en el robot:

- **Estado de corrección del ángulo del robot:** Su objetivo es disminuir el error en la orientación con respecto a q_{dest}. Para ello, se realiza el calculo del ángulo de referencia existente entre el robot y la posición deseada (Ecuación 3.16).

$$\theta_{referencia} = atan2(y_2 - y_1, x_2 - x_1) \tag{3.16}$$

  Donde $[x_1, y_1]$ representan el valor de las coordenadas x, y referentes a la posición q_{curr} del robot, mientras que, $[x_2, y_2]$ son la representación de las coordenadas x, y de la posición q_{dest}. En cada calculo se presenta un margen de error derivado de los cálculos, para obtener el valor de dicho margen en la orientación del robot se realiza la diferencia entre el valor del ángulo de referencia y el ángulo actual del robot (Ecuación 3.17).

$$\theta_{error} = \theta_{referencia} - \theta_{robot} \tag{3.17}$$

  Cuando θ_{error} tiende más hacía cero el robot puede cambiar al siguiente estado para avanzar en línea recta.

- **Estado de desplazamiento en línea recta (avanzar):** El objetivo de este estado es conducir al robot lo más cerca posible de la configuración deseada q_{dest}. Para realizar las comprobaciones de la distancia entre q_{curr} y q_{dest} se calcula la distancia Euclidiana (Ecuación 3.18).

$$d = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2} \tag{3.18}$$

Se establece un parámetro de tolerancia el cual se evalúa con el valor de d y si este último se encuentra dentro del rango aceptado, entonces se procede al siguiente estado del robot. En caso de que θ_{error} es mayor, entonces el estado se define en la corrección del ángulo del robot nuevamente. La Figura 3.3.1 el diagrama referente al comportamiento de desplazarse en línea recta.

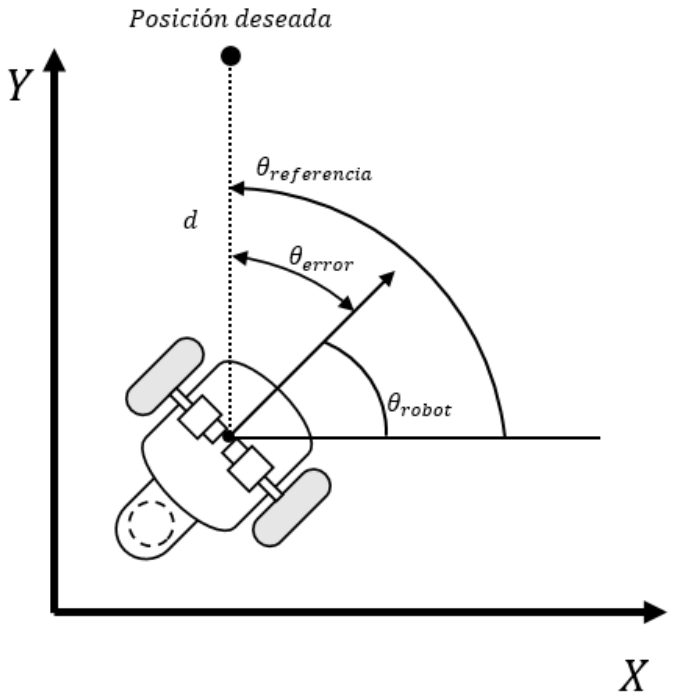

Figura 3.3.1 Diagrama de comportamiento del estado "Avanzar en línea recta".

- **Estado Completado:** Se establece cuando las dos acciones anteriores son completadas, puede darse al llegar a un nodo q_{dest} intermedio de la exploración o al alcanzar la configuración meta.

De igual manera, una parte importante es la referente a la ubicación del robot. Para esto, ROS permite obtener esta información si se realiza una suscripción al tema que se encuentre publicando mensajes referentes a la odometría del robot. En este punto se tiene que realizar una aclaración, el trabajo se probó bajo un entorno de simulación, por lo cual el tema al que se suscribe es llamado *"/odom"*, ya que la odometría está siendo obtenida desde el simulador Gazebo, el cual publica mensajes de tipo nav_msgs::Odometry. En caso de que el presente trabajo se aplicara con el robot real, el tema al que debería realizarse la suscripción sería a *"/RosAria/pose"*, que igualmente, realiza la publicación de mensajes de tipo nav_msgs::Odometry.

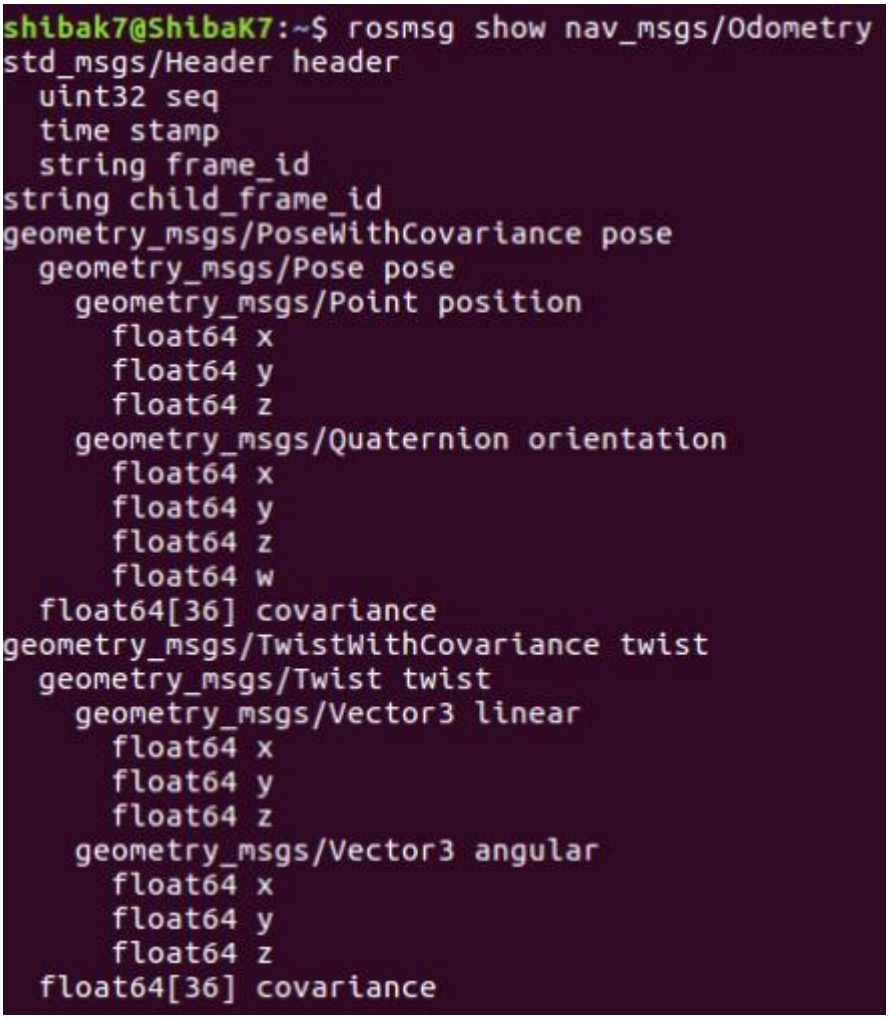

Figura 3.3.2 Estructura del mensaje nav_msgs::Odometry.

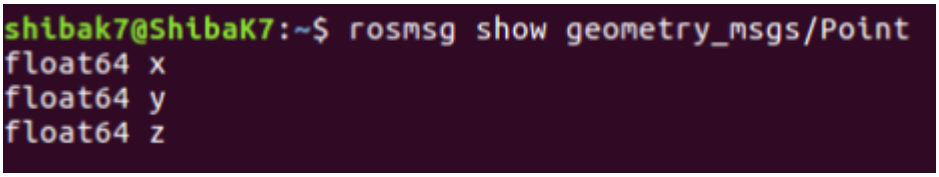

Figura 3.3.3 Estructura del mensaje geometry_msgs::Point.

nav_msgs::Odometry es un tipo de mensaje compuesto (Figura 3.3.2), dentro de los mensajes que contiene se utiliza el elemento *"posittion"* de tipo geometry_msgs/Point (Figura 3.3.3), donde se encuentran las coordenadas en x, y, z de la posición del robot. En este momento se tendrían las coordenadas de la ubicación del robot, sin embargo, faltaría contemplar la orientación en la que se encuentra el sistema robótico, ya que, sin esto, la localización del punto estaría en diferencia con la ubicación real. Para obtener la orientación se necesita acceder al elemento *"orientation"* el cual esta expresado en cuarterniones,

utilizados en procesos referentes a rotaciones 3D, y una vez con los valores del elemento, se puede elegir entre dos procesos para definir la orientación del robot:

- **Aplicación de librerías de transformaciones:** ROS ofrece librerías que proveen de funciones para realizar las trasformaciones requeridas para obtener la orientación del robot.

- **Aplicación de la Ecuación 3.19**: La ecuación describe la relación entre el eje Z en el contexto de un plano 2D, dado por el ángulo θ y el elemento Q_Z en cuarterniones [24].

$$\theta_{robot} = 2 \cdot \arcsin(Q_Z) \tag{3.19}$$

3.3.2 Definición de la nube de puntos (cloud points).

De igual forma, uno de los procesos importantes que debe realizar el sistema robótico es el sensado del ambiente, ya se ha definido en apartados anteriores, como la información obtenida dará pauta para desencadenar las acciones correspondientes a la exploración. Para lograr acceder a los mensajes correspondientes a sensor láser Hokuyo se tiene que realizar una suscripción al tema *"p3dx/laser/scan"*, esto en la parte de simulación, la cual permitirá obtener mensajes de tipo sensor_msgs::LaserScan (Figura 3.3.4). Si se deseará utilizar el robot real se tendría que realizar la suscripción al tema *"/scan"*, el cual también devolvería mensajes de tipo sensor_msgs::LaserScan.

```
shibak7@ShibaK7:~$ rosmsg show sensor_msgs/LaserScan
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
float32 angle_min
float32 angle_max
float32 angle_increment
float32 time_increment
float32 scan_time
float32 range_min
float32 range_max
float32[] ranges
float32[] intensities
```

Figura 3.3.4 Estructura del mensaje sensor_msgs::LaserScan.

Los mensajes de LaserScan obtienen las relaciones espaciales y las formas del entorno midiendo el tiempo que tardan las señales, en rebotar con los objetos y regresar al laser, de esta manera se realiza la definición del elemento *"ranges"*, junto con el resto de elementos presentes en el mensaje se puede definir un valor asociado a los ángulos que comprende el sensor para determinar la proximidad de algún obstáculo para el robot. Sin embargo, este tipo de mensaje regresa un arreglo de valores de tipo flotante (referentes a los rangos), y para la incorporación del proceso que trata las características del ambiente a través de curvas B-Splines es necesario manejar la información del entorno a través de puntos, correspondientes a mensajes de tipo sensor_msgs::PointCloud (Figura 3.3.5.

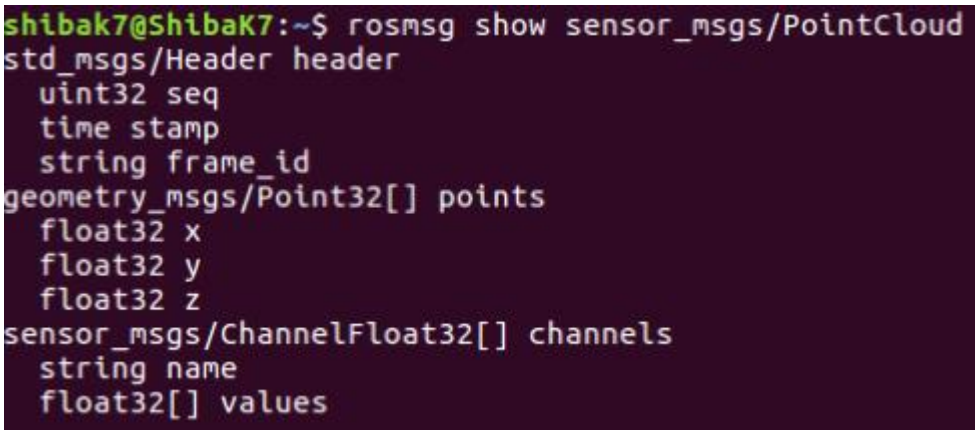

Figura 3.3.5 Estructura del mensaje sensor_msgs::PointCloud.

Para dar solución a esta problemática un proceso de transformación de mensajes es realizado, convirtiendo mensajes de tipo sensor_msgs::LaserScan a sensor_msgs::PointCloud, la forma de realizar esta acción puede variar de acuerdo al lenguaje en el que se realice la implementación, dado que existen alternativas para C++ y Phyton, para el lenguaje en que se ha desarrollado el trabajo (C++), existen dos formas de realizar el proceso de transformación según [25]:

- **Proyección simple:** Como su nombre lo indica se realiza una proyección simple del láser. Los mensajes sensor_msgs::PointCloud se generan en el mismo marco que los mensajes sensor_msgs::LaserScan originales (Figura 3.3.6).

```
laser_geometry::LaserProjection projector_;

void scanCallback (const sensor_msgs::LaserScan::ConstPtr& scan_in)
{
    sensor_msgs::PointCloud cloud;
    projector_.projectLaser(*scan_in, cloud);

    // Trabajar con la nube de puntos cloud
}
```

Figura 3.3.6 Código de Proyección simple (C++).

En este proceso no es necesario utilizar una instancia del paquete tf, sin embargo, este tipo de transformación no es recomendada para situaciones en la cual el sistema robótico se tenga que desplazar

- **Proyección de alta fidelidad:** En este caso se realiza una proyección más avanzada, para eso, es necesario que se establezca un oyente de transformación tf. Los datos son recopilados a lo largo del tiempo de escaneo, para este proceso es importante seleccionar un target_frame de marco fijo, los mensajes sensor_msgs::LaserScan son procesados desde la primera medición realizada por el láser, pasan a un proceso de transformación hasta la última medición obtenida (Figura 3.3.7).

```
laser_geometry::LaserProjection projector_;
tf::TransformListener listener_;

void scanCallback (const sensor_msgs::LaserScan::ConstPtr& scan_in)
{
    if(!listener_.waitForTransform(
    scan_in->header.frame_id,
    "/base_link",
    scan_in->header.stamp + ros::Duration().fromSec(scan_in->ranges.size()*scan_in->time_increment),
    ros::Duration(1.0))){
    return;
}

sensor_msgs::PointCloud cloud;
projector_.transformLaserScanToPointCloud("/base_link",*scan_in,cloud,listener_);

// Trabajar con la nube de puntos cloud

}
```

Figura 3.3.7 Código de Proyección de alta fidelidad (C++).

La proyección de alta fidelidad es recomendada para situaciones en las cuales el robot se encuentre en movimiento constante.

En el caso de nuestra problemática se seleccionó la implementación de la transformación de alta fidelidad, dado que, el robot se desplazará dentro del ambiente y tendrá que realizar observaciones mientras se mueve en él. De igual forma se realizaron pruebas con ambos métodos, teniendo un mejor rendimiento y precisión la opción elegida.

3.3.3 Conceptos de movimiento del robot Pioneer P3-DX

Para cumplir los pasos movimiento y detección de los obstáculos con el robot Pioneer P3-DX se delimitaron regiones válidas a las cuales podría dirigirse, recordemos que el robot equipado con el sensor Hokuyo cuenta con un rango de amplitud de 240°, para la definición de mensajes sensor_msgs::LaserScan se generan 727 rayos que parten del robot en diferentes ángulos de amplitud. En total se establecieron 5 regiones válidas, cada una con una amplitud de 48° (Figura 3.3.8). De igual manera, tomando como consideración las dimensiones del robot Pioneer P3-DX y su punto de rotación, presente en el centro del eje que une sus ruedas, se determinó que la distancia mínima permitida para efectuar una rotación libre de los obstáculos.

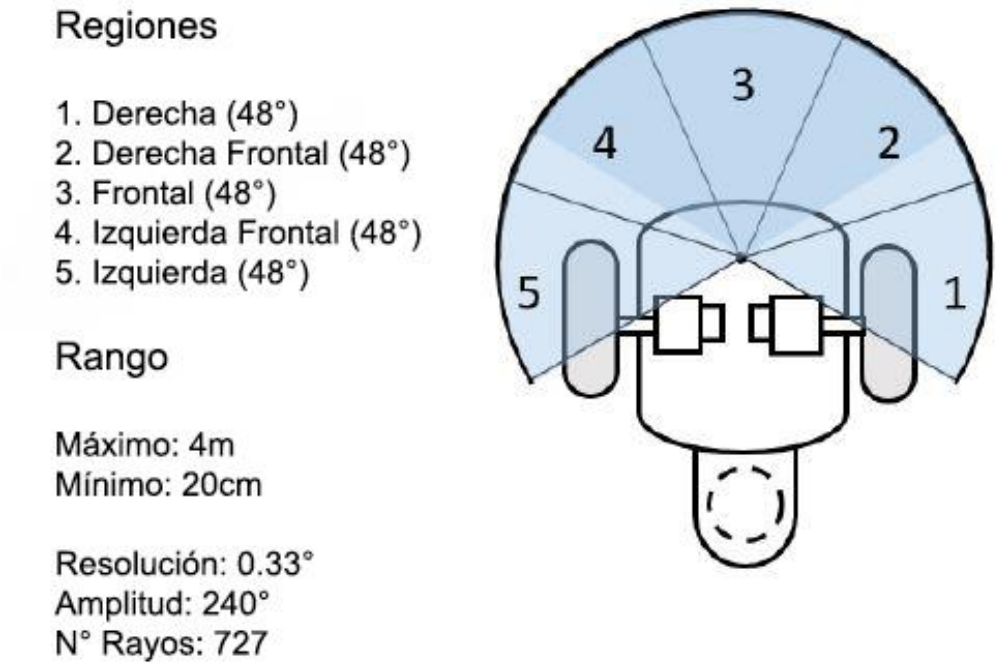

Figura 3.3.8 Definición de las regiones del láser.

A través de la definición de las regiones el robot es capaz de establecer un nuevo nodo para avanzar, manteniendo una distancia segura con los obstáculos que se pueda encontrar en su trayectoria hacia la meta mientras sigue las fronteras establecidas en el SRT presentado. De igual forma, la definición de las regiones del láser forma un papel importante, dado que, si en alguna región es localizado un obstáculo que no permita el desplazamiento del robot, entonces esta región no pertenecerá a la región segura local del sistema robótico, en el nodo en cuestión, la forma de la SRL y los arcos válidos que contenga determinará cuantos nuevos nodo candidato serán generados. Los nodos se definirán en el punto más alejado del robot perteneciente a cada arco en cuestión, definiendo de esta forma las fronteras de la SRL, tal y como se detalla en el tema 3.2.4.2. En la figura 3.3.9 se presenta un ejemplo de la definición de la SRL, donde las regiones 1 (Derecha) y 5 (Izquierda) contienen obstáculos que impiden el movimiento del robot hacía esas áreas. De tal forma, la SRL queda definida por las regiones 2 (Derecha Frontal), 3 (Frontal) y 4 (Izquierda Frontal), en las cuales se definen los nodos correspondientes a las fronteras de la SRL en los arcos de cada región.

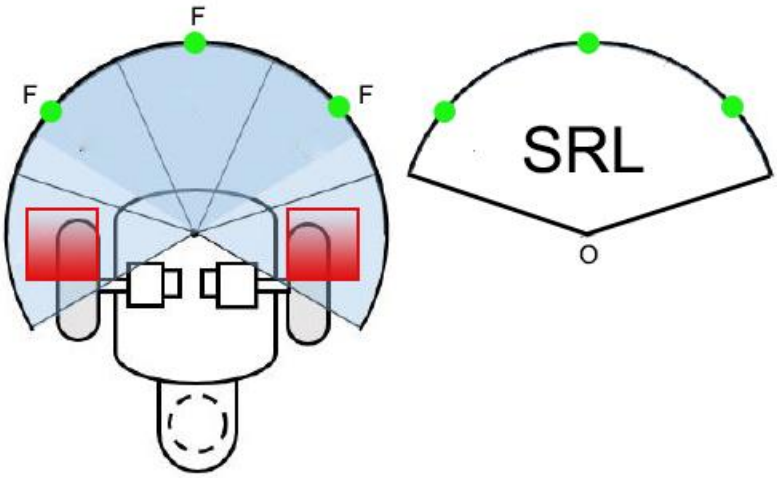

Figura 3.3.9 Ejemplo definición de la SRL del robot.

De esta manera, cada vez que se realiza una publicación en el tema correspondiente al sensor láser, se realiza la evaluación del nodo más adecuado para realizar la transición. Como se menciona en la especificación del SRT, el robot seguirá generando candidatos destinos delimitando fronteras dentro de los arcos pertenecientes entre el robot y las características observadas pertenecientes a la SRL, de tal manera, el proceso se repetirá hasta que el robot logre llegar a la configuración meta.

Capítulo 4

Resultados

En este capítulo se presentan los resultados obtenidos en la implementación del trabajo de tesis en un entorno simulado, los ambientes que se presentan son desconocidos para el robot Pioneer P3-DX a la hora de realizar el proceso de exploración. En cada uno de los mapas presentados se establecen las coordenadas con las que inicia el robot y las coordenadas del destino del robot (meta), de igual forma, se establece el tiempo de ejecución, los nodos generados en la exploración y el número de vectores de los segmentos pertenecientes a los puntos de control que definen el mapa.

Las pruebas se realizaron con el simulador Gazebo y para ello se utilizaron mapas diferentes, de igual forma, se utilizó el modelo del robot Pioneer P3-DX, creado inicialmente por Rafael Berkvens [26] adaptado a nuestro robot.

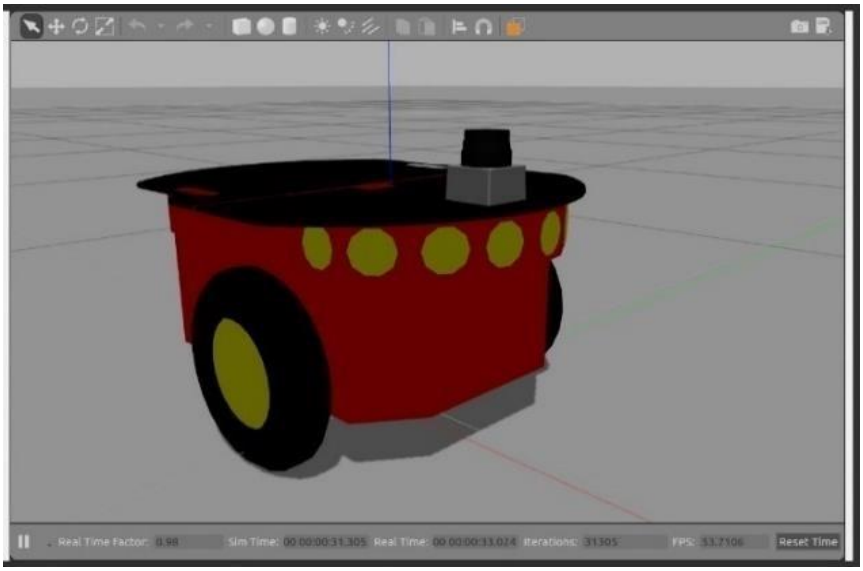

Figura 4.1.1. Modelo del Robot Pioneer P3-DX en Gazebo.

Las simulaciones se realizaron dentro del sistema operativo Ubuntu con apoyo del framework Qt Creator con el que es compatible el desarrollo de proyectos ROS, cada simulación presenta imágenes del inicio en rviz y gazebo, el mapa obtenido a partir de la implementación de la solución propuesta y el mapa que genera rviz a partir de los escáneres del láser. Es importante realizar la comparación entre el mapa generado por rviz y el mapa obtenido de la aplicación del algoritmo de exploración integrada basado en SRT, de esta manera podemos comprobar la exactitud de los resultados obtenidos a nivel de simulación.

De igual forma es preciso establecer que el mapa que se genera por defaul en rviz solo está presente durante la ejecución del programa, al finalizar el proceso de rviz los datos correspondientes al mapa se eliminan, por otro lado, los segmentos que almacenan las características del mapa en la solución de SRT propuesta se pueden almacenar y usar incluso después de finalizar el proceso de prueba. De esta forma, se podrá utilizar la información recabada en un tipo de exploración offline en un robot similar esto como un complemento.

4.1 Mapa 1: Pasillo

Para este ambiente se presenta un pasillo que se encuentra conectado por una sección sesgada, en este ambiente el robot inicia en la parte inferior del pasillo y navegará hasta la parte superior, aunque no se presentan obstáculos dentro de los pasillos los límites del

mapa deberán ser bien definidos por el robot para realizar correctamente el proceso de exploración. La figura 4.1.2 presenta la configuración de la posición del robot y los resultados obtenidos.

Inicio (X, Y)	Destino (X, Y)	Nodos Generados	Segmentos de Puntos	Tiempo en segundos
(0, 0)	(-2, 8)	59	69	48.9612

Figura 4.1.2. Descripción Mapa 1: Pasillo.

Nodos (mínimos)	43
Nodos (máximos)	63
Tiempo (mínimo)	42.8547
Tiempo (máximo)	98.1254

Figura 4.1.3. Comparativa Resultados Mapa 1: Pasillo.

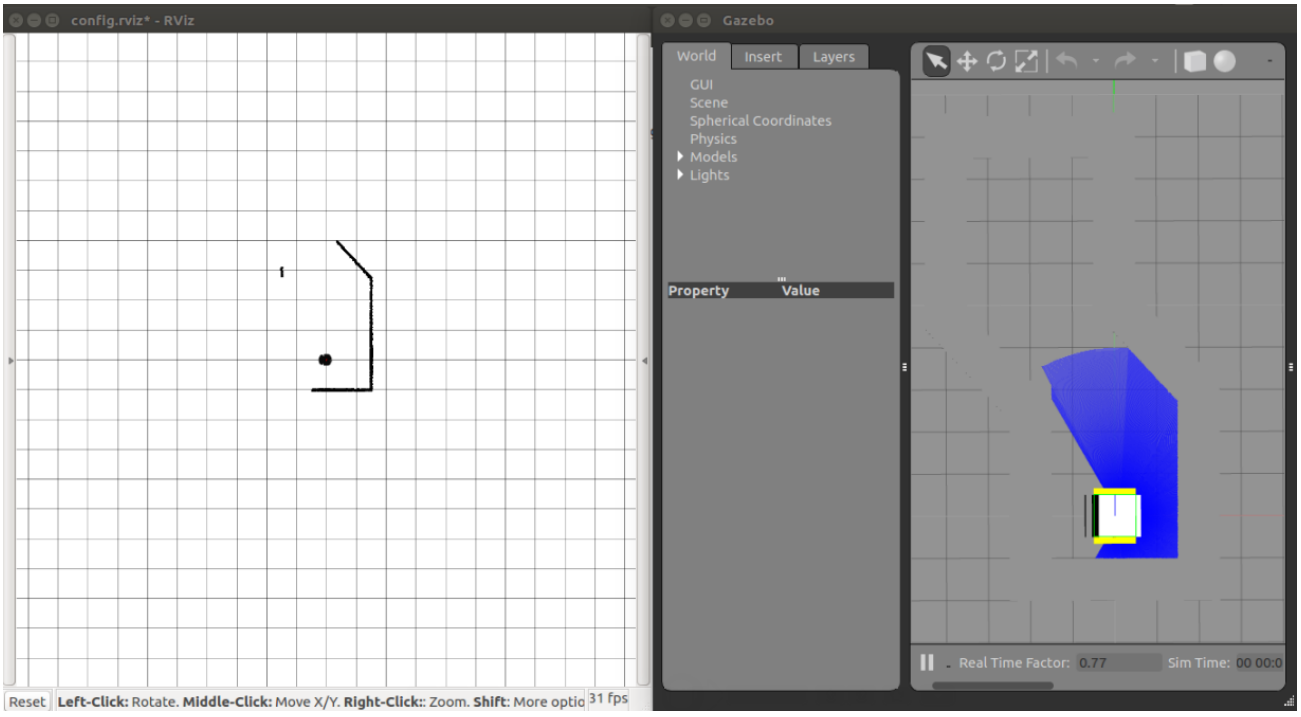

Figura 4.1.4. Configuración Inicial Mapa 1: Pasillo.

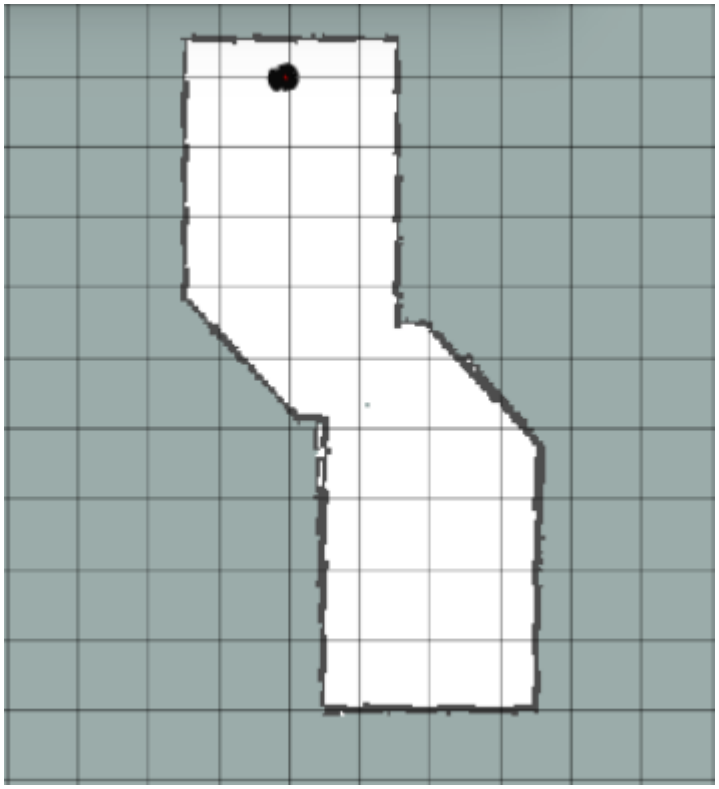

Figura 4.1.5. Resultado Mapa 1 generado por rviz.

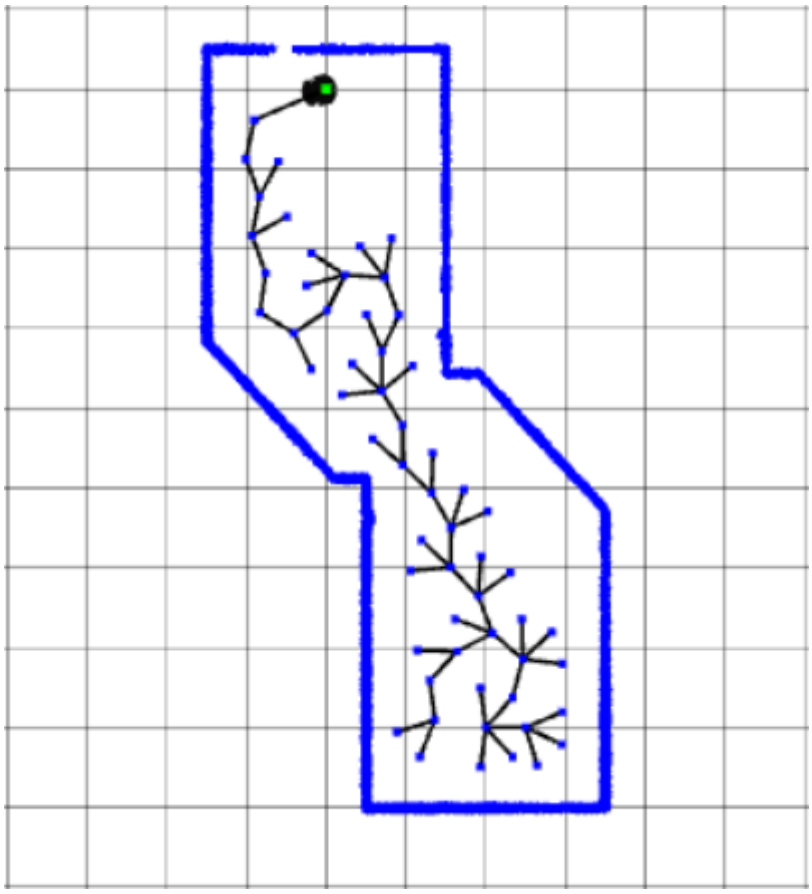

Figura 4.1.6. Resultado Mapa 1 generado por el algoritmo SRT propuesto.

Como resultados tenemos que el proceso de auto localización se realizó con éxito al llegar correctamente a la meta, además el mapa que genera rviz es igual al obtenido por la implementación del algoritmo de SRT propuesto, con esto se establece que la definición de las características de los límites del mapa se realiza de manera efectiva, de igual forma, tenemos que dichas características que conforman el mapa están contenidas en un total de 68 segmentos que a su vez contienen los puntos de control obtenidos. Finalmente, para completar la exploración, detallada en la Figura 4.1.2, se establecieron un total de 59 nodos en un tiempo de 49 segundos aproximadamente.

4.2 Mapa 2: Cuarto con obstáculo en L

Para el segundo ambiente se tiene un cuarto con forma cuadrada que contine un obstáculo en forma de L, el robot inicia en la parte inferior izquierda y se define la meta en la parte superior derecha, existen dos caminos para llegar a la meta, tomando el pasillo superior o tomar el pasillo de la derecha, rodeando el obstáculo, los detalles de la implementación y resultados obtenidos se presentan a continuación.

Inicio (X, Y)	Destino (X, Y)	Nodos Generados	Segmentos de Puntos	Tiempo en segundos
(0, 0)	(3, 4)	53	48	60.1271

Figura 4.2.1. Descripción Mapa 2: Cuarto con obstáculo en L.

Nodos (mínimos)	48
Nodos (máximos)	64
Tiempo (mínimo)	50.4116
Tiempo (máximo)	86.4671

Figura 4.2.2. Comparativa Resultados Mapa 2: Cuarto con obstáculo en L.

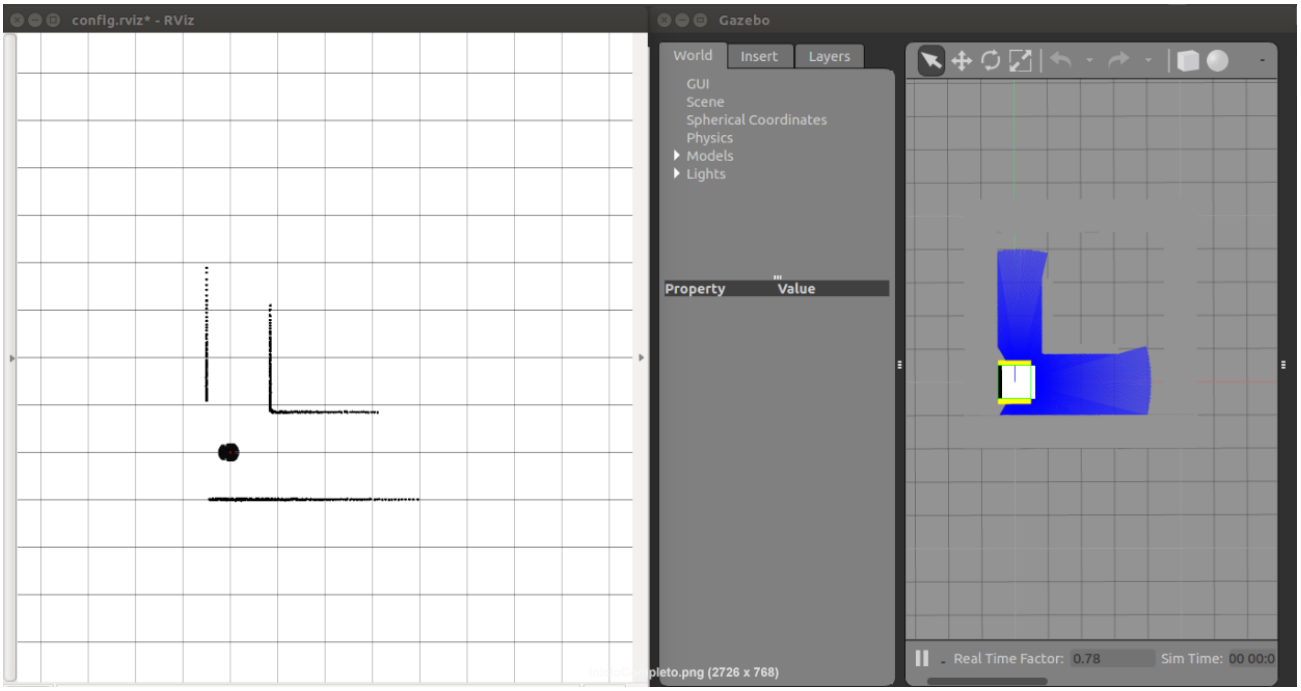

Figura 4.2.3. Configuración Inicial Mapa 2: Cuarto con obstáculo en L.

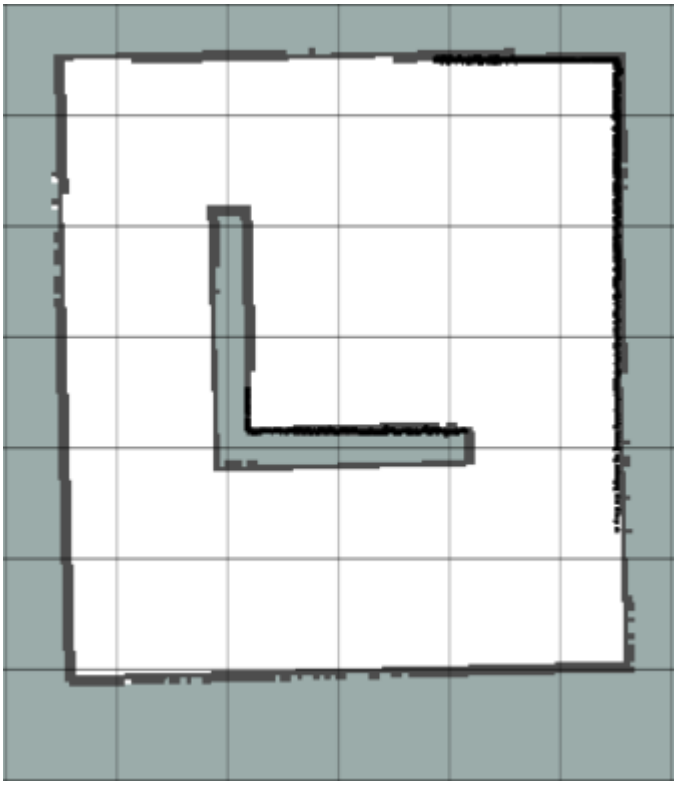

Figura 4.2.4. Resultado Mapa 2 generado por rviz.

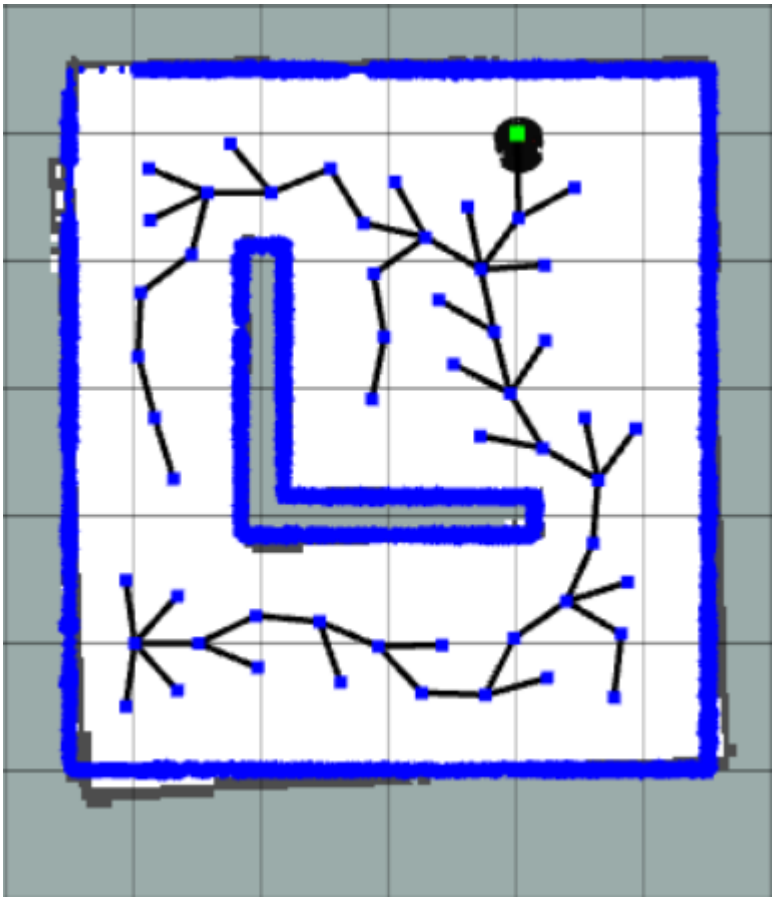

Figura 4.2.5. Resultado Mapa 2 generado por el algoritmo SRT propuesto.

Respecto a los resultados obtenidos en el segundo mapa tenemos que el proceso de auto localización tuvo éxito al llegar correctamente a la meta, se observa que el robot opto por navegar por el pasillo de la derecha, tanto el mapa que se genera en rviz como el que se obtiene como resultado de la aplicación del algoritmo de SRT coinciden, sin embargo, los límites establecidos por el SRT propuesto presenta una delimitación más precisa que el generado por defecto en rviz. La forma de delimitar el obstáculo se visualiza correctamente en la Figura 4.2.5, se obtuvieron un total de 53 nodos con la definición de 48 segmentos de puntos, que establecen los puntos de control con las características del mapa, en un tiempo total de 60 segundos aproximadamente.

4.3 Mapa 3: Laboratorio Movis

Para el tercer entorno se realizó una aproximación a un mapa construido en base al laboratorio de Movis, dentro de los aspectos importantes el mapa cuenta con limites curvos que presentarán un buen reto para el algoritmo de SRT propuesto, estos límites están en la

primera sala del mapa que tiene una estructura no geométrica la cual conecta con otra sala con una figura que aproxima un rectángulo, como el mapa es una representación a escala del laboratorio el ambiente es más pequeño en comparación de los escenarios anteriores.

Inicio (X, Y)	Destino (X, Y)	Nodos Generados	Segmentos de Puntos	Tiempo en segundos
(0, 0)	(2.5, 2.5)	14	55	19.0963

Figura 4.3.1. Descripción Mapa 2: Cuarto con obstáculo en L.

Nodos (mínimos)	14
Nodos (máximos)	16
Tiempo (mínimo)	19.0963
Tiempo (máximo)	22.8283

Figura 4.3.2. Comparativa Resultados Mapa 3: Laboratorio Movis.

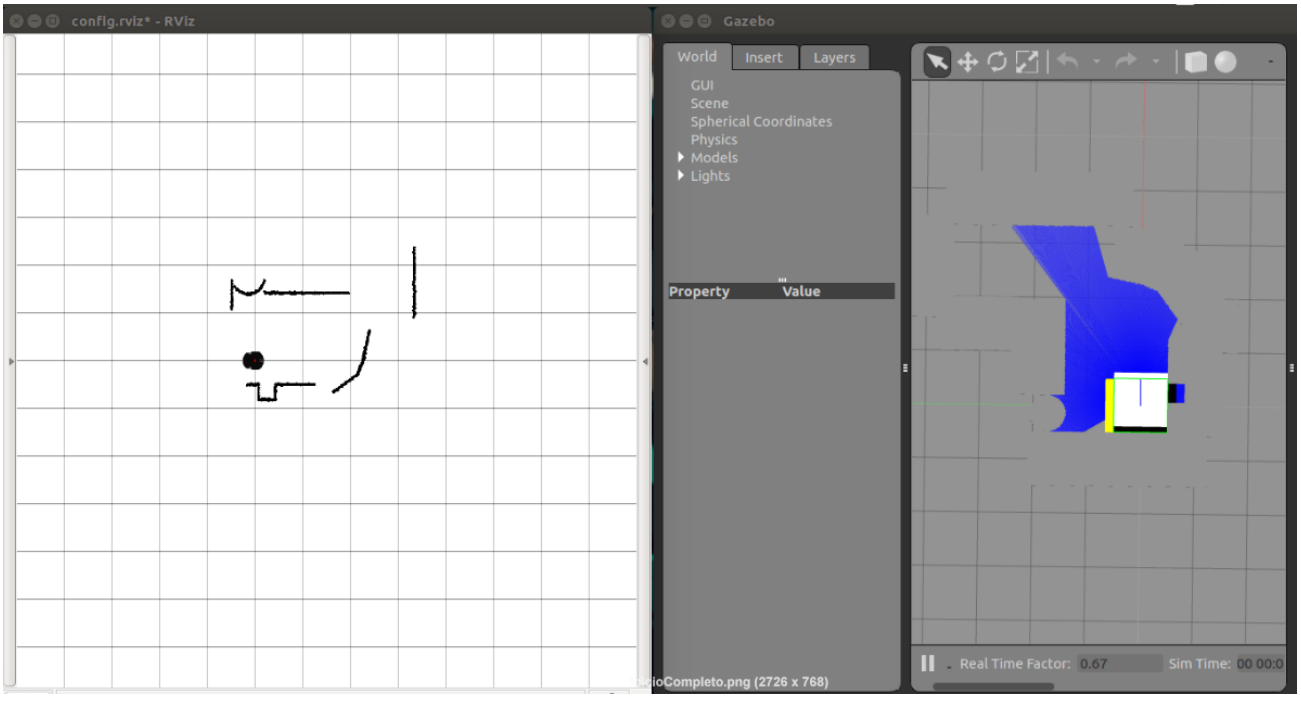

Figura 4.2.3. Configuración Inicial Mapa 3: Laboratorio Movis.

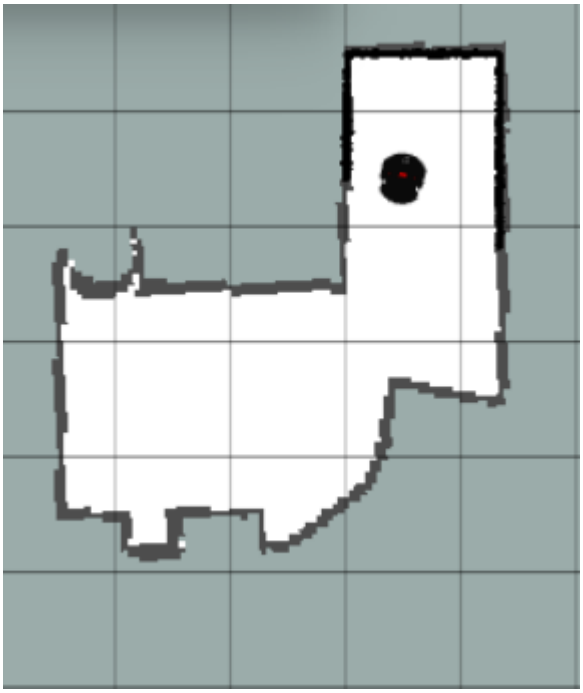

Figura 4.3.4. Resultado Mapa 3 generado por rviz.

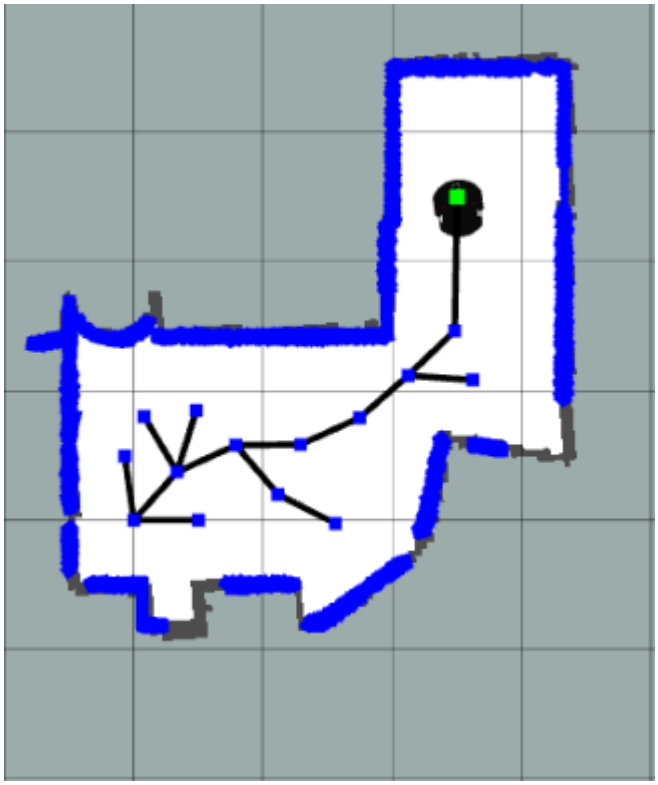

Figura 4.3.5. Resultado Mapa 3 generado por el algoritmo SRT propuesto.

Dentro de los resultados obtenidos en el mapa 3 tenemos que la autolocalización como en los casos anteriores cumplió correctamente con su función al llevar al robot a la meta planteada, respecto a la construcción del mapa tanto rviz como el algoritmo SRT propuesto

establecen similitudes, no obstante, en este caso al ser un mapa relativamente pequeño, donde a lo máximo se generaron 16 nodos, el procesamiento del algoritmo de SRT no tuvo las iteraciones suficientes para agregar más puntos de control en la delimitación de las características del mapa, en esta situación rviz genera un mapa tan solo un poco más completo, de igual forma, los puntos obtenidos por el algoritmo de SRT define correctamente las características que logró procesar. En la navegación se generaron 14 nodos, obteniendo un total de 55 segmentos de puntos en un tiempo total de 19 segundos aproximadamente, siendo este ejemplo el que tiene el menor número de nodos generados y tiempo en todas las pruebas que se hicieron sobre este mapa.

4.4 Mapa 4: Unión de pasillos

Para este cuarto ambiente se presenta un escenario donde se unen dos pasillos en forma de T, el robot inicia en la parte inferior del pasillo principal y a mitad de dicho pasillo se presenta la unión al pasillo secundario, este mapa contrario al anterior es el más grande de los presentados, la meta para este caso se encuentra en la parte superior del pasillo principal, sin embargo, se busca que el robot explore también el pasillo secundario. El detalle de la implementación y los resultados obtenidos se presentan a continuación.

Inicio (X, Y)	Destino (X, Y)	Nodos Generados	Segmentos de Puntos	Tiempo en segundos
(0, 0)	(0, 7)	74	84	97.2624

Figura 4.4.1. Descripción Mapa 4: Unión de pasillos.

Nodos (mínimos)	68
Nodos (máximos)	74
Tiempo (mínimo)	19.0963
Tiempo (máximo)	106.4410

Figura 4.4.2. Comparativa Resultados Mapa 4: Unión de pasillos.

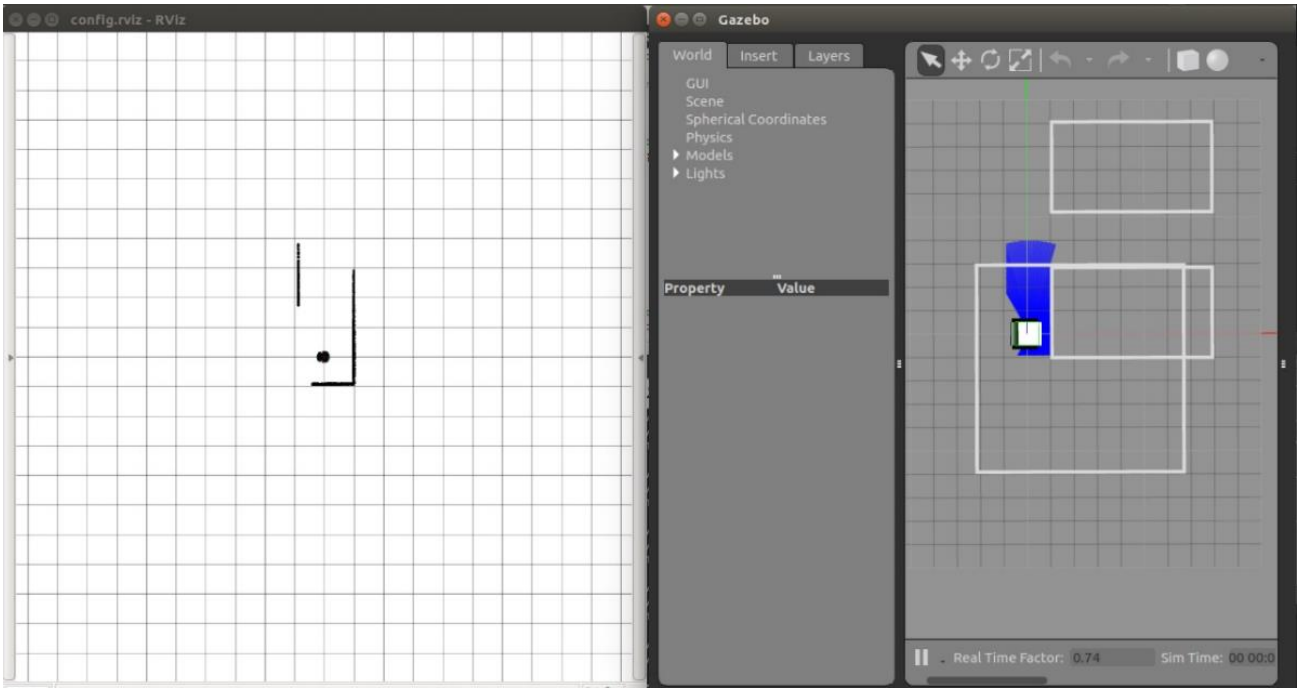

Figura 4.4.3. Configuración Inicial Mapa 4: Unión de pasillos.

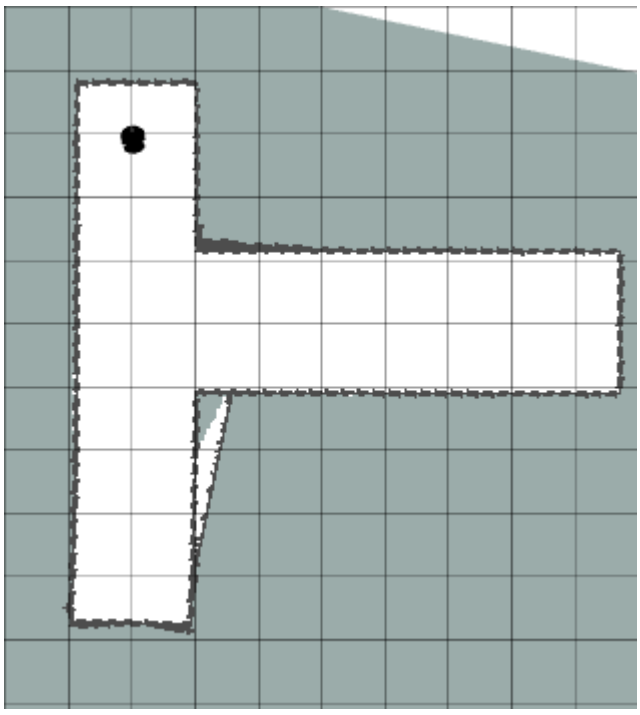

Figura 4.4.4. Resultado Mapa 4 generado por rviz.

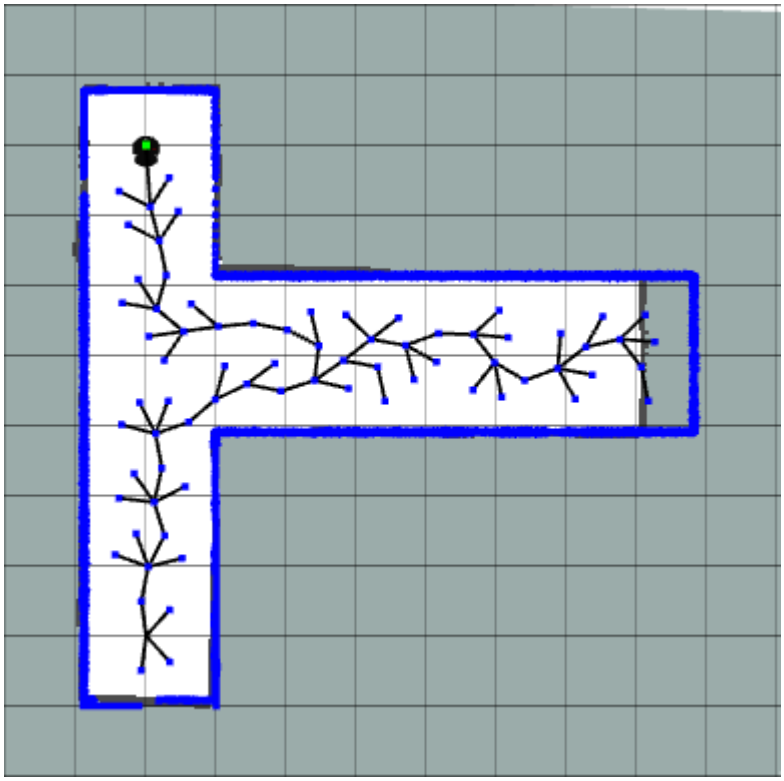

Figura 4.4.5. Resultado Mapa 4 generado por el algoritmo SRT propuesto.

Los resultados de este mapa son interesantes, para comenzar podemos observar que la Figura 4.4.4 y la Figura 4.4.5 no corresponden a la misma navegación, la razón de esto es que como se observa en la segunda figura, el mapa generado por rviz recorto el largo del pasillo secundario, para esta prueba del escenario el algoritmo de SRT construye correctamente el mapa con las lecturas del sensor láser, mientras que el mapa construido en rviz presenta una inconsistencia muy importante, dado que, si se toma ese mapa el robot podría colisionar. La Figura 4.4.4 corresponde al mapa construido en otra prueba donde los límites del mapa dados en rviz se generan de forma correcta, sin embargo, podemos seguir observando inconsistencias dado que presenta un límite transversal en la parte inferior del pasillo principal que une con el secundario. Para este mapa en cuestión el algoritmo de SRT propuesto genero un mejor desempeño en todos los casos de prueba que se ejecutaron, en este ejemplo se puede visualizar la importancia del remuestreo en el filtro de partículas, dado que, a partir de esto se puede realizar los ajustes en la construcción del mapa. Volviendo a la prueba de la Figura 4.4.5 se obtuvieron un total de 74 nodos, 84 segmentos de puntos en un tiempo total de 97 segundos aproximadamente, el proceso de auto localización se ejecutó correctamente ya que en todos los casos el robot logró llegar exitosamente a la meta definida.

4.5 Mapa 5: Cuarto con múltiples obstáculos

Para el quinto escenario se presenta un cuarto en el cual están definidos diversos obstáculos, algunos con formas definidas como un rectángulo, cuadrados y otros más complejos. En este mapa la intención, a diferencia de los anteriores, es mostrar un ejemplo donde al robot no le sea posible llegar a las coordenadas destino, recordemos que en la solución de SRT propuesta cuando el robot no le es posible generar más nodos candidatos que lo acerquen al destino este regresará al nodo padre hasta llegar al nodo inicial y ahí finalizará la ejecución.

Inicio (X, Y)	Destino (X, Y)	Nodos Generados	Segmentos de Puntos	Tiempo en segundos
(0, 0)	(5, 3)	No definidos	No definidos	No definido

Figura 4.5.1. Descripción Mapa 5: Cuarto con múltiples obstáculos.

Con la información de la Figura 4.5.1 se establece que el robot no logró llegar a la meta establecida, por lo cual, no genero información correspondiente al total de nodos generados, los segmentos obtenidos y menos el total del tiempo necesario para llegar a la meta. En la Figura 4.5.2 se logra observar la información correspondiente a la configuración inicial en el ambiente de rviz y en Gazebo.

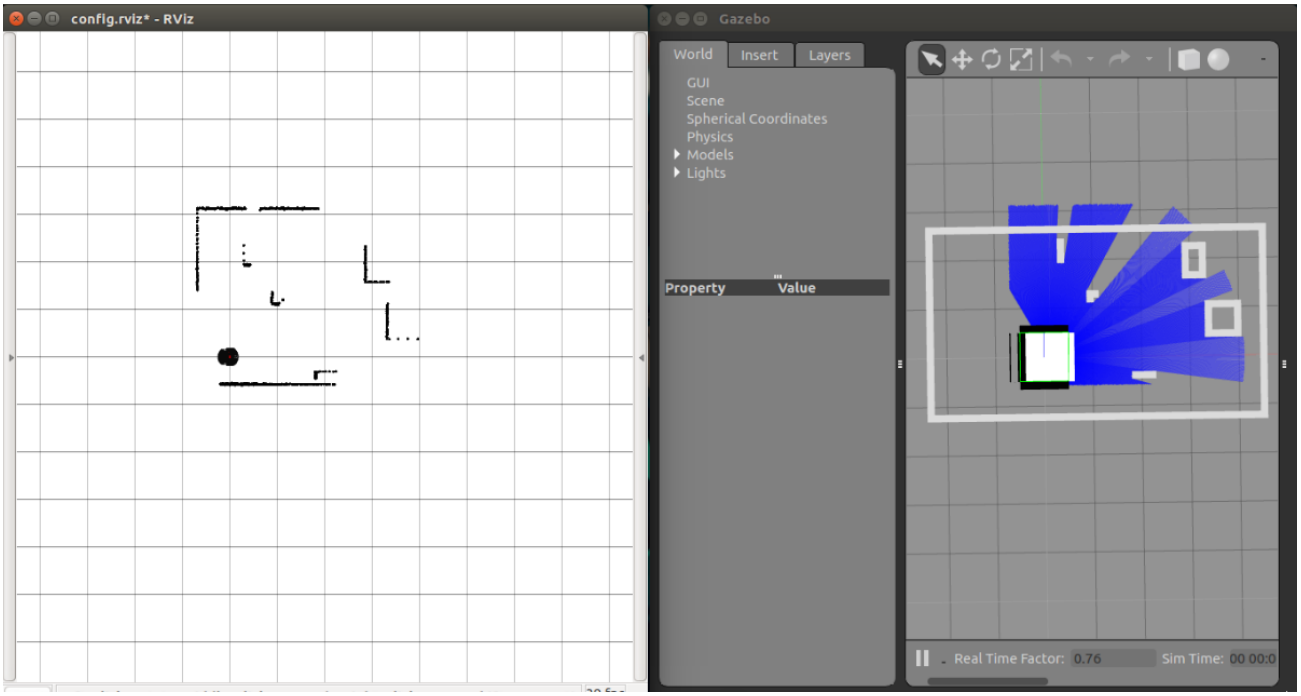

Figura 4.5.2. Configuración Inicial Mapa 5: Cuarto con múltiples obstáculos.

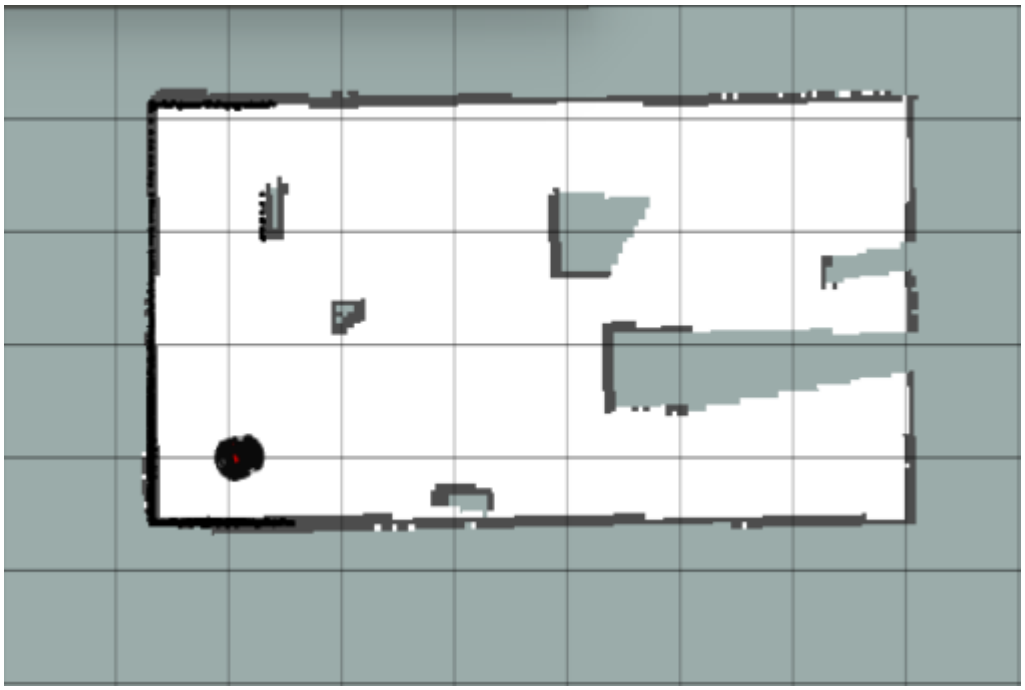

Figura 4.5.3. Resultado Mapa 4 generado por rviz.

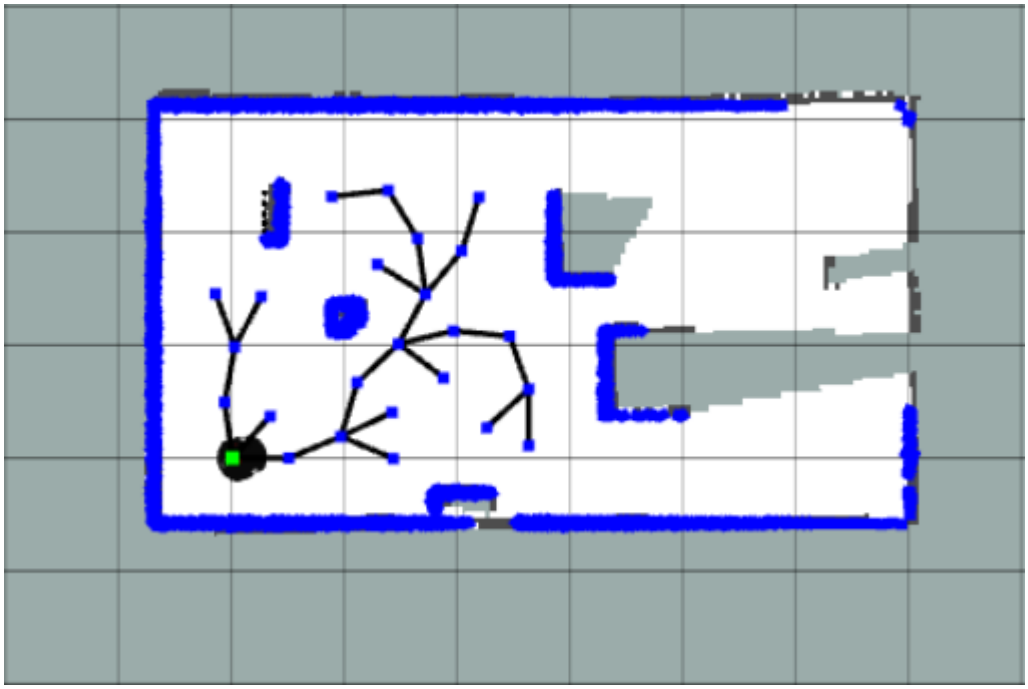

Figura 4.5.4. Resultado Mapa 5: generado por el algoritmo SRT propuesto.

Los resultados para este mapa se evalúan de forma diferente a los anteriores, debido a que no se obtienen valores para los nodos generados, los segmentos finales y el tiempo de ejecución. Sin embargo, las Figuras 4.5.3 y 4.5.4 nos muestran que rviz y el algoritmo de SRT propuesto si obtuvieron características del ambiente, las suficientes para realizar una construcción parcial del mapa, se puede observar que en rviz se tiene un mayor porcentaje del mapa, no obstante, la definición de las características del mapa esta mejor representada por el algoritmo implementado. En comparativa los mapas construidos no se encuentran completamente construidos, la definición de los obstáculos presentes se establece correctamente hasta donde fue posible la exploración, al finalizar se observa como el robot regresa al nodo inicial.

Capítulo 5

Conclusiones y trabajo a futuro

Durante el presente trabajo de tesis se utilizó el algoritmo de exploración integrada basado en SRT para el proceso de auto localización, además como incorporación al trabajo se agregó el proceso de recopilación de características del mapa explorado utilizando curvas B-Splines, el objetivo que se planteo es utilizar dichas curvas para recabar los límites y obstáculos que componen el entorno, y de esta manera, dar una solución al problema de SLAM agregando la construcción del mapa mientras el robot se desplaza de una configuración inicial A(X, Y) a una meta B(X, Y).

De igual modo, la forma de exploración que adopta el robot es definir una serie de nodos candidatos que se encuentran en la frontera de la zona segura local, del nodo actual, pertenecientes a los arcos de las regiones dadas por el sensor láser, en este trabajo es importante que el robot llegue a la meta, pero también es importante que el robot pueda recorrer la mayor parte del mapa para que se genere una construcción total del ambiente. Como se observó en el Mapa 5, de los resultados, si el robot no puede llegar a la configuración meta definida entonces la construcción del mapa se interrumpe hasta donde el robot deje de generar nuevos nodos candidatos, el mapa queda construido parcialmente y regresa a su configuración inicial, obteniendo un mapa parcialmente construido.

Un aspecto importante a resaltar es sobre el marco de trabajo ROS, la utilización de este framework permitió manejar correctamente la complejidad del desarrollo y la implementación de los algoritmos presentados, dado que, al utilizar sus nodos las tareas y comportamientos se distribuían aminorando la carga de trabajo para el nodo principal. Por ejemplo, al iniciar el proceso de rviz se recibían los parámetros de la configuración del mundo a explorar, también llegaba información correspondientes a la odometría y escaneo por parte del sensor láser Hokuyo, todo esto se conectaba al programa principal para realizar la actualización tanto de la navegación como de las características que se observaban, el manejo de esta información fue posible gracias a la integración de proyectos como RosAria y urg_node, los cuales ROS permitió integrar y manejar con facilidad.

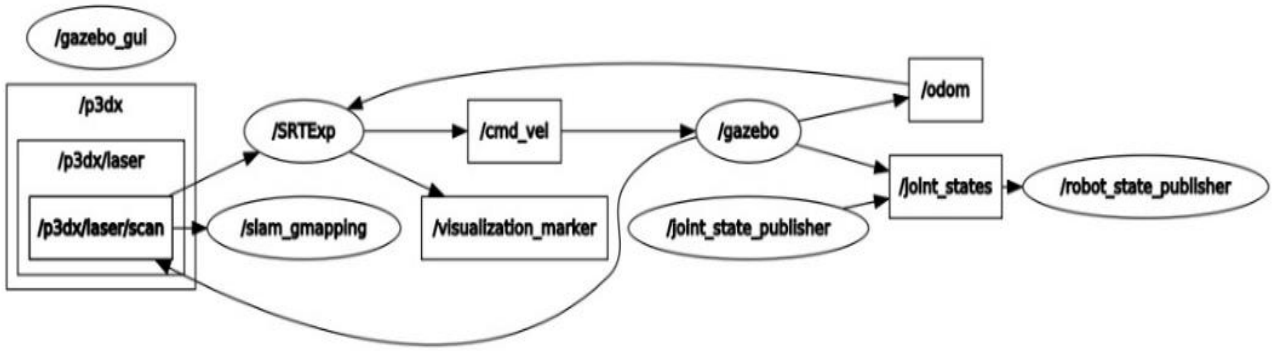

Figura 5.1.1. Ejemplo de comunicación de los nodos de ROS.

Con este trabajo se continua el trabajo realizado en el laboratorio de Movis utilizando el framework ROS, aunque este marco de trabajo este restringido para ciertos sistemas operativos las ventajas que ofrece son un factor muy importante a considerar, por lo cual, sería de gran utilidad su uso en trabajos futuros.

Finalmente, como consideraciones a futuro que puedan enriquecer y complementar el presente trabajo se sugiere la incorporación de ambientes de pruebas reales, donde se pueda probar la eficiencia de los algoritmos bajo entorno real, con sus respectivas adecuaciones, y evaluar su rendimiento, esto a primera mano, sería un complemento importante para el trabajo. De igual manera, se podrían estructurar las funcionalidades presentes en el trabajo dentro de un módulo para que la solución propuesta se logrará reutilizar utilizando las herramientas que ofrece ROS, a partir de esto, diferentes proyectos podrían integrar el módulo de exploración de ambientes complejos utilizando curvas B-Splines, en medida que lo requieran.

Bibliografía

[1] R. Siegwart y I. R. Nourbakhsh, Introduction to Autonomous Mobile Robots, Massachusetts: The MIT Press, 2004.

[2] V. Galiote Teutle, Estrategias para la auto localización de robots móviles, Puebla, Puebla, 2019, p. 82.

[3] A. Toriz Palacios y A. Sánchez López, Probabilistic integrated Exploration for Mobile Robots in Complex Enviroments, 2014.

[4] H. Durrant-Whyte y T. Bailey, «Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms,» *IEEE Robotics and Automation Magazine,* 2006.

[5] F. Andrade y M. Llofriu, «SLAM».

[6] R. Smith, M. Self y P. Cheeseman, «A stochastic map for uncertain spatial relationships,» MIT Press, 1987.

[7] S. J. Julier y J. K. Uhlmann, «A Counter Example to the Theory of Simultaneous Localization and Map Building,» de *IEEE Int. Conf. on Robotics and Automation*, Seoul, Korea, 2001.

[8] E. B. Raigal, «Aplicación del Filtro de Particulas al seguimiento de objetos en secuencias de imágenes,» Madrid, España, 2003.

[9] N. J. Gordon, D. J. Samond y A. F. M. Smith, «Joint Audio-Visual Tracking Using Particle Filters,» *Eurasip Journal on Applied Signal Processing,* 2002.

[10] L. Pedraza, D. Rodríguez-Losada, F. Matía, G. Dissanayake y J. Miro, «Extending the Limits of Feature-Based SLAM with B-Splines,» *IEEE Transactions on Robotics,* vol. 25, nº 2, p. 353–366, 2009.

[11] C. De Boor, New York: Springer-Velarg, 1978.

[12] A. Toriz Palacios y A. Sánchez López , «Método de asociación de datos basado en curvas B-Spline para el problema de SLAM en ambientes complejos,» vol. 21, nº 2, pp. 353-368, 2017.

[13] R. Org, «ROS,» Movable Type, [En línea]. Available: https://www.ros.org/about-ros/. [Último acceso: 31 07 2021].

[14] O. Robotics, «Wiki ROS Org,» [En línea]. Available: http://wiki.ros.org/es/ROS/Introduccion. [Último acceso: 31 07 2021].

[15] UA Cloud Campus Virtual, «Manual de ROS,» [En línea]. Available: https://moodle2018-19.ua.es/moodle/mod/book/view.php?id=8465&chapterid=181. [Último acceso: 2021 01 2021].

[16] Gazebo, «Gazebo Sim,» Open Source Robotics Foundation, [En línea]. Available: http://gazebosim.org/tutorials?tut=build_model. [Último acceso: 31 07 2021].

[17] I. Anvari, Non-holonomic Differential Drive Mobile Robot Control & Design : Critical Dynamics and Coupling Constraints, Arizona, 2013.

[18] K. Dempski, Focus on Curves and Surfaces, Premier Press, 2003.

[19] G. Oriolo, M. Vendittelli, L. Freda y G. Troso, «The srt method: Randomized strategies for exploration,» *IEEE,* 2004.

[20] V. Galiote T., A. Sánchez L., J. F. Texcucano D., A. Toriz P., R. González V. y E. Pérez G., «Planificación de movimientos basada en sensores para robots móviles en ambientes desconocidos,» *Research in Computing Science,* p. 147–158, 2019.

[21] S. Jurić-Kavelj, «ROSARIA,» 2018. [En línea]. Available: http://wiki.ros.org/ROSARIA. [Último acceso: 2021].

[22] C. Rockey y M. O'Driscoll, «urg_node,» 2017. [En línea]. Available: http://wiki.ros.org/urg_node. [Último acceso: 2021].

[23] M. Serna Hernández, Navegación de robots móviles utilizando los algoritmos Bug extendidos, Puebla, 2018.

[24] «D3DXQUATERNION structure Microsoft Developer Network,» 2018. [En línea]. Available: https://docs.microsoft.com/es-es/windows/desktop/direct3d9/d3dxquaternion. [Último acceso: 08 2021].

[25] T. Foote y R. Bogdan Rusu, «ROS.org,» [En línea]. Available: http://wiki.ros.org/laser_geometry. [Último acceso: 02 2021].

[26] R. Berkvens, «ua_ros_p3dx,» [En línea]. Available: https://github.com/RafBerkvens/ua_ros_p3dx.