UNIVERSIDAD
POLITÉCNICA
DE MADRID

Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Industriales

Máster Universitario en Automática y Robótica

Trabajo Fin de Máster

NAVEGACIÓN TOPOLÓGICA PARA EXPLORACIÓN DE MINAS ABANDONADAS

Autor: **Adrián Caro Zapata**

Director : **Claudio Rossi**

Co-director: **Ramón Suarez Fernández**

octubre, 2020

Condemnant quo non intellegunt.

Agradecimientos

Con este trabajo, culmino este maravilloso año, rodeado de grandes profesionales y compañeros, en el que he tenido el placer de profundizar en una de mis mayores pasiones en la vida, la Robótica y la Automatización.

Mil gracias a mi familia, por brindarme todo y más en esta vida, por apoyarme incondicionalmente y acompañarme siempre.

A mis directores, por guiarme en este proyecto y proyectar su profesionalidad en mi. Por sacar siempre lo mejor de los ellos.

A dos maravillosas personas, que se convirtieron en grandes amigos desde aquel primer día, Hugo y Pablo, con los que he compartido innumerables momentos y alegrías, así como nos hemos apoyado durante largas jornadas trabajando juntos.

A mis amigos, los que nunca fallan, los que me hacen recordar lo maravillosa que es la vida día a día y con los que comparto mis alegrías y penas, mis chavs.

Me gustaría agradecer también y hacer mención de toda la gente que está haciendo todo lo posible para superar estos tiempos tan desafortunados, en medio de esta pandemia mundial del COVID19. Fuerza.

Resumen

Este Trabajo de Fin de Máster ha sido inspirado por el proyecto europeo UNEXMIN. Dicho proyecto hace uso de robots sumergibles autónomos para explorar, cartografiar y caracterizar de forma no invasiva minas abandonadas inundadas en Europa. Esto permite obtener datos actualizados para evaluar su reapertura y volver a Europa más independiente de la importación de estas materias primas, a la vez que genera un registro actualizado de las minas.

Dicho proyecto se presenta muy ambicioso y complejo y está compuesto por una multitud de módulos y equipos de trabajo. Este Trabajo de Fin de Máster se inspira en una parte de la navegación autónoma mediante mapas topológicos y a la toma de decisiones del robot de dicho proyecto, la cual resulta bastante atractiva y presenta un reto realmente interesante.

El principal objetivo que se plantea es conseguir que el robot navegue (simuladamente) de forma eficiente por el laberinto de una mina inundada, dado su mapa topológico, alcanzando correctamente los puntos de destino fijados por el usuario y haciendo frente a errores de navegación. Al tratarse de una simulación, el usuario ha de poder monitorizar las decisiones y posición del robot mediante una representación 3D del grafo.

Para ello, se decidió partir de un sistema informado, donde el robot es conocedor del mapa, dado en forma de mapa topológico.

Se ha programado un algoritmo capaz de recoger la información de dicho mapa topológico y generar una estructura de grafo a partir de él.

El grafo generado no sólo está compuesto por nodos y aristas. Se han añadido más estructuras como segmentos, que componen las aristas, lo que permite una mejor caracterización del entorno. Además se ha implementado una identificación de objetos de interés durante la navegación para realizar confirmación de posiciones.

La información del grafo se puede proporcionar de dos formas: a partir de un archivo de texto o generando un grafo aleatorio mediante un generador de grafos dados el número de nodos deseados.

Mediante este algoritmo, el usuario puede seleccionar un nodo objetivo para que el robot navegue hasta él. La generación de rutas se ha programado siguiendo un algoritmo Dijkstra de búsqueda en grafos. Este permite seleccionar la ruta más corta entre dos nodos de forma rápida y óptima.

Una vez obtenida la ruta, se simula la navegación. El robot puede partir desde una posición desconocida e intentar localizarse, o partir desde una posición conocida. Una vez estimada su posición, va navegando por el grafo y va recibiendo información simulada de los sensores. Esta información es comparada con la información que dispone del mapa, conocido inicialmente. A raíz de la comparación, se lleva a cabo la toma de decisiones de la navegación sobre qué camino seguir para alcanzar el objetivo. A medida que el robot avanza, se va generando una incertidumbre de posición, la cual se restablece al confirmar su posición mediante sus sensores u objetos de interés.

El robot es capaz de detectar errores de posición respecto a la ruta, a veces forzados por el propio algoritmo para proporcionar dinamismo a la navegación. Cuando detecta un error de posición, es capaz de relocalizarse y de replanificar una nueva ruta hacia su objetivo.

Todo este proceso es monitorizado por parte del usuario mediante la herramienta Rviz de ROS, que representa en 3D el grafo informado y la ruta y posición del robot en todo momento. Adicionalmente, cada uno de los pasos y toma de decisiones del robot durante su navegación es monitorizado a través de una ventana de comandos.

Palabras clave: Robot autónomo, mina inundada, grafos, navegación topológica, planificación, localización.

Códigos UNESCO: 331101, 331105, 250508

Abstract

This Master's Thesis has been inspired by the European project UNEXMIN. The project uses autonomous submersible robots to non-invasively explore, map and characterize flooded abandoned mines in Europe. This allows obtaining up-to-date data to evaluate their reopening and making Europe more independent from importing these materials, while generating an updated record of the mines.

This project is very ambitious and complex and is made up of a multitude of modules and work teams. This Master's Thesis is inspired by a part of the autonomous navigation through topology and the decision-making of the robot of this project, which is quite attractive and presents a really interesting challenge.

The main objective that arises is to get the robot to navigate (simulated) efficiently through the labyrinth of a flooded mine, given its topological map, correctly reaching the destination points set by the user and making navigation errors. As it is a simulation, the user must be able to monitor the decisions and the position of the robot in a 3D representation of the graph.

For this, it was decided to start from an informed system, where the robot is aware of the map, given in the form of a topological map.

An algorithm capable of collecting the information from said topological map has been programmed and generating a graph structure from it.

The generated graph is not only composed of nodes and edges. More structures have been added as segments, which make up the edges, which allows a better characterization of the environment. In addition, an identification of objects of interest has been implemented during navigation to carry out the confirmation of positions.

The information of the graph can be provided in two ways: from a text file or generating a random graph using a graph generator given the desired number of nodes.

Using this algorithm, the user can select a target node for the robot to navigate to. The generation of routes has been programmed following a Dijkstra search algorithm in graphs. This allows to select the shortest path between two nodes quickly and optimally.

Once the route is obtained, navigation is simulated. The robot can start from an unknown position and try to locate itself, or start from a known position. Once its position is estimated, it navigates through the graph and receives simulated information from the sensors. This information is compared with the information available on the map, initially known. Following the comparison, the navigation decision-making is carried out on which way to go to reach the goal. As the robot advances, a position uncertainty is generated, which is restored by confirming its position through its sensors or objects of interest.

The robot is capable of detecting position errors with respect to the route, sometimes forced by the algorithm itself to provide dynamism to navigation. When it detects a positional error, it is able to relocate and re-plan a new route to its goal.

This entire process is monitored by the user using the ROS Rviz tool, which represents the reported graph in 3D and the path and position of the robot at all times. Additionally, each of the robot's steps and decision-making during its navigation is monitored through a command window.

Keywords: Autonomous robot, flooded mine, graphs, topological navigation, planning, location.

UNESCO Codes: 331101, 331105, 250508

Índice

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Marco y motivación

Desde sus comienzos hasta el día de hoy, la robótica no ha dejado de evolucionar, mejorando en todos sus ámbitos y con ello, expandiéndose a una gran infinidad de sectores. Esta evolución abre nuevas oportunidades en sectores donde antes la robótica no tenia una aplicación competitiva o que ofreciese una ventaja real, y donde ahora su implementación desarrolla un papel indispensable.

Numerosas evoluciones en hardware como la implementación de novedosos materiales de mejores prestaciones, y en software, como el desarrollo y aplicación de *Machine Learning* están permitiendo a los robots ser más versátiles y desempeñar nuevas tareas antes solo realizables por humanos.

Uno de los retos que se presenta actualmente es el desarrollo de robots completamente autónomos, sobre todo en condiciones adversas. Existen diversos proyectos que tratan con robots teleoperados o con comunicación directa con este, pero dando un paso más allá, existen condiciones donde esto no es viable o presenta numerosos problemas. Es ahí donde los robots autónomos están tomando parte.

Estas condiciones adversas se presentan por ejemplo en minas inundadas, aún repletas de recursos sin explotar, pero que han quedado inutilizadas por los problemas que presenta su explotación en dichas condiciones. Estas suelen encontrarse a gran profundidad, y completa o parcialmente inundadas. Usualmente suelen estar formadas por un entramado de túneles, con densas paredes de roca, donde las telecomunicaciones no pueden operar. La posibilidad de tener una conexión directa cableada entre el centro de mando y el robot queda descartada al tratarse de laberintos de túneles donde el robot ha de navegar, y a la resistencia que esto genera al avance de este. Se plantea entonces el uso de robots

autónomos para la tarea de explorar dichas minas inundadas, eliminando los problemas anteriormente mencionados.

Un robot autónomo es capaz de explorar las minas gracias a sus sensores de posición, y de lectura e identificación del entorno, pudiendo navegar sobre lugares previamente informados (mapa del entorno conocido por el robot), o por lugares desconocidos, en cuyo caso el robot iría generando el mapa del entorno.

Se abre entonces una nueva posibilidad, la de acceder a minas inundadas abandonadas, pudiendo explorarlas e incluso explotarlas, con todas la ventajas que eso supone para la industria europea. Así surge el proyecto UNEXMIN, proyecto subvencionado por la Comisión Europea.

UNEXMIN es la inspiración de este Trabajo de Fin de Máster, mediante el cual se trata de diseñar una parte de este complejo proyecto, concretamente el diseño y programación de la navegación topológica por parte del robot submarino. Esto incluye la lectura del mapa, toma de decisiones y navegación del robot en la mina inundada.

1.1.1. Proyecto UNEXMIN

Cada vez es mayor la necesidad de obtener materias primas para producir dispositivos electrónicos y baterías, elementos enormemente reclamados por las últimas tecnologías. La Unión Europea busca volverse más independiente de la importación de estas materias primas y mejorar la actividad de investigación en minas, ya que su producción anual de minerales metálicos (cobalto, litio, níquel y grafito) supone apenas un 3 % de la producción mundial [3] y ha de comprar una gran cantidad de estos. Europa busca ser un líder mundial en la producción y el uso sostenibles de baterías, en el contexto de la economía circular.

Mediante la creación del proyecto UNEXMIN se busca desarrollar robots sumergibles autónomos para explorar, cartografiar y caracterizar minas abandonadas inundadas en Europa. Muchas de estas minas pueden contener todavía cantidades rentables de materias primas. El principal motivo del cierre o abandono de las minas fue económico: la no muy avanzada tecnología y los costosos métodos de extracción de minerales hacían más conveniente importar la materia prima necesaria de otros países (ej. Chile, Congo, China...).

Hoy en día, con el desarrollo de nuevas tecnologías de minería y refinación, las minas que ya no se consideraban explotables por razones económicas y técnicas en el momento del cierre, pueden ser reconsideradas y reevaluadas para reducir la dependencia de Europa

en la importación de materias primas.

El robot sumergible UX-1 se puede utilizar en el estudio de sitios potenciales. Puede explorar y mapear minas subterráneas inundadas de forma no invasiva y proporcionar datos útiles para la industria minera con su instrumentación científica, siendo capaz de operar sin control remoto. El equipo científico incluye un muestreador de agua, una unidad de medición de conductividad y pH, un perfilador de subfondo, una unidad de medición de campo magnético, imágenes de fluorescencia UV y unidades de imágenes multiespectrales [10].

Figura 1.1 Robot UX-1 explorando la cueva Molnár János. Budapest, Hungría. (https://www.unexmin.eu/)

Este proyecto, junto con el desarrollo de robots, ha creado el primer inventario de minas inundadas de Europa, contando ya con más de 8000 en 24 países. Se espera que la base de datos UNEXMIN y el robot contribuyan a la expansión de la industria de las baterías en Europa.

Información adicional:

- **Proyecto:** UNEXMIN
- **Presupuesto Global:** 4.862.865 €.
- **Coordinado por:** Miskolci Egyetem, Hungría
- **Fecha de inicio:** 1 de febrero de 2016
- **Fecha final:** 31 de octubre de 2019
- **Programa:** H2020-EU.3.5.3. - Garantizar el suministro sostenible de materias primas no energéticas y no agrícolas.
- **Página del proyecto europeo:** https://cordis.europa.eu/project/id/690008
- **Página del proyecto:** https://www.unexmin.eu/

Como se ha mencionado anteriormente, este proyecto al ser tan ambicioso y complejo, se ha dividido en módulos para su desarrollo. Una visión general de las partes que lo componen se puede observar en la Figura 1.2.

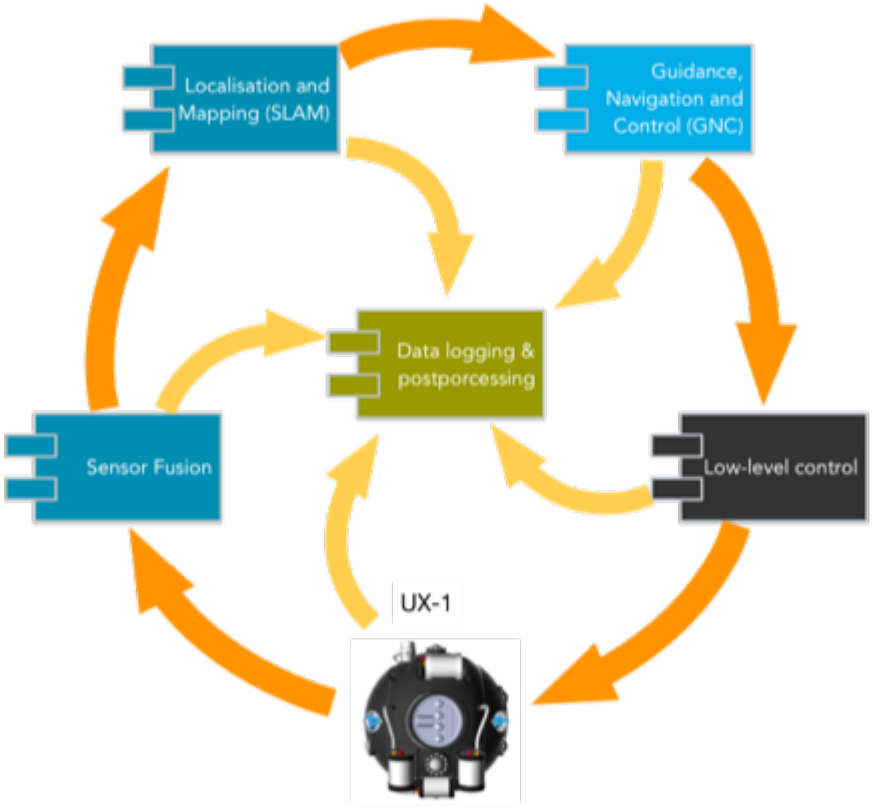


Figura 1.2 Esquema general de los módulos que componen el proyecto UNEXMIN y las conexiones entre estos.

Este Trabajo de Fin de Máster se centra en el desarrollo de la lógica de navegación, que forma parte del módulo de Guiado, Navegación y Control del robot.

Este módulo está directamente relacionado con el módulo de localización y mapeado mediante SLAM, con el control de nivel bajo y con el tráfico de datos y postprocesado.

Estas interacciones entre módulos se muestran de forma más visual en la Figura 1.3.

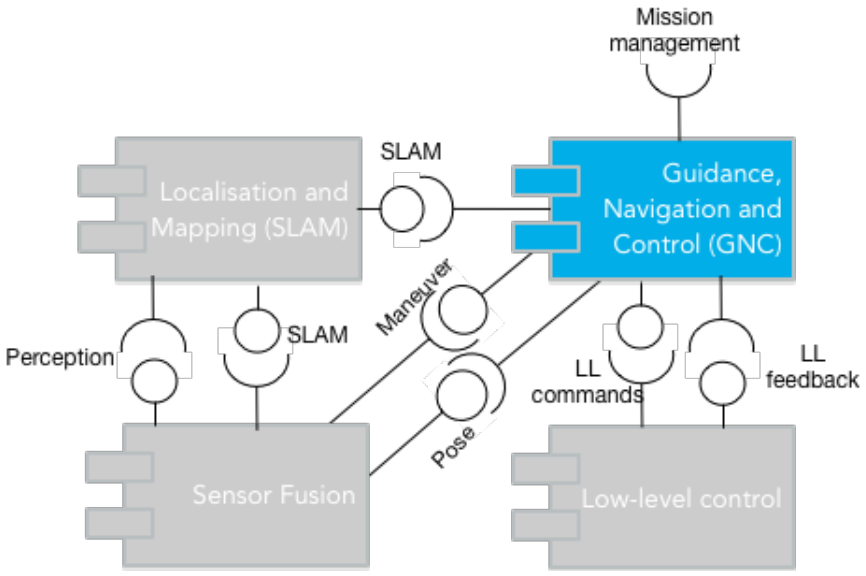


Figura 1.3 Esquema de interacción del módulo de Guiado, Navegación y Control con otros módulos del proyecto UNEXMIN.

Donde el módulo de Guiado, Navegación y Control del robot se compone de la siguiente forma:

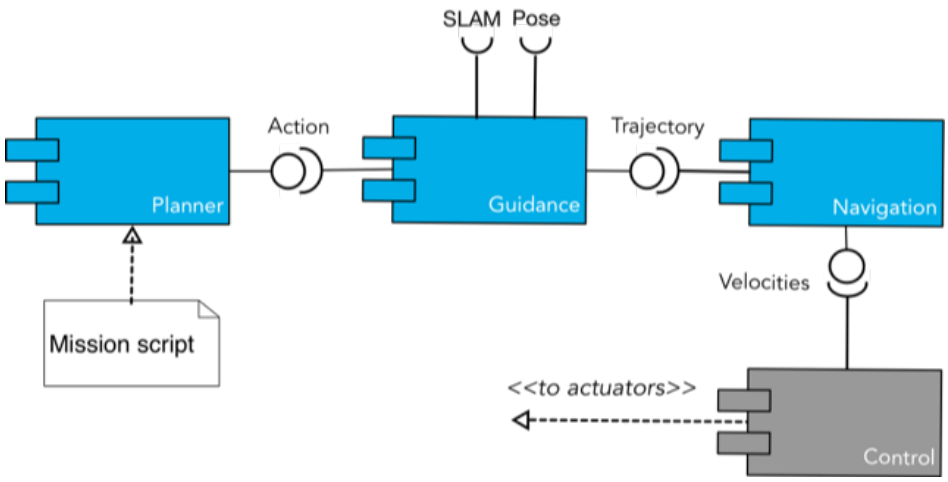


Figura 1.4 Esquema del módulo de Guiado, Navegación y Control del robot en el proyecto UNEXMIN.

En cuanto a la comunicación entre módulos, esta se realiza mediante nodos de ROS. En este trabajo se desarrolla un nodo ROS, el cual interaccionaría con los demás nodos mediante la suscripción y publicación en *topics*, obteniendo información de estos, aplicando la arquitectura de toma de decisiones en la navegación y volviendo a publicar los resultados obtenidos para que puedan ser utilizadas por otras partes del programa. Esta interacción mediante nodos de ROS se puede observar esquemáticamente en la Figura 1.5.

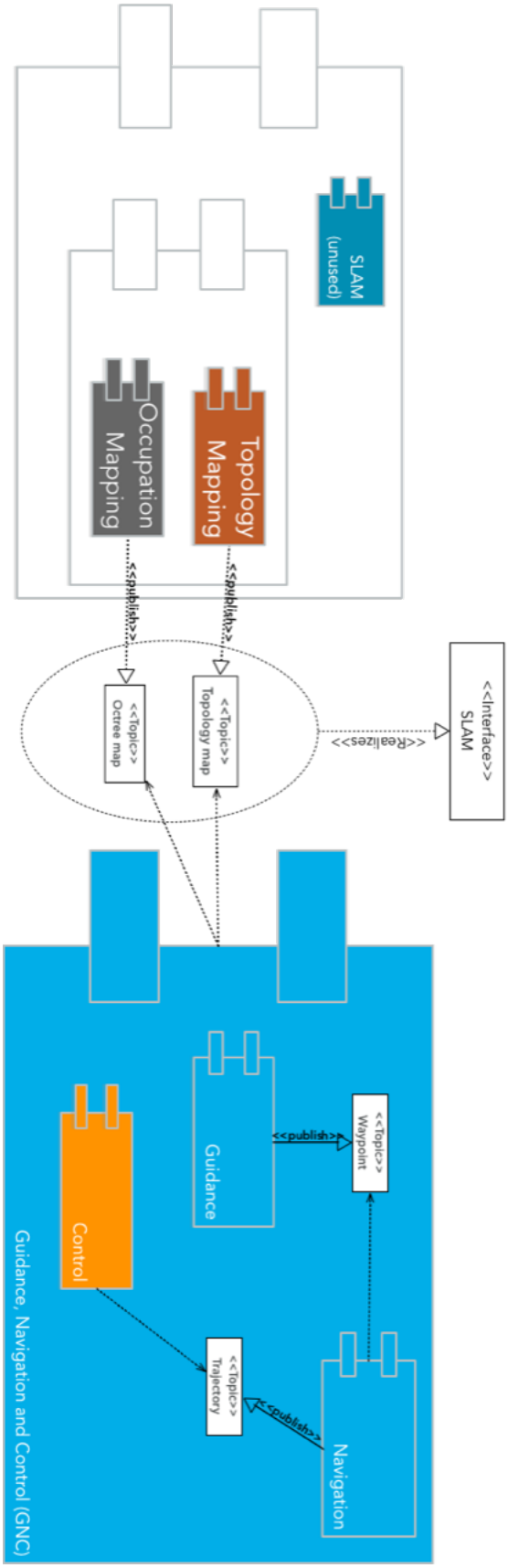


Figura 1.5 Esquema de interacción entre nodos de ROS en el esquema del proyecto UNEXMIN.

1.2. Objetivos

Los objetivos del presente Trabajo de Fin de Máster están orientados a conseguir que el robot navegue (simuladamente) de forma eficiente por el laberinto de una mina inundada, dado su mapa topológico, alcanzando correctamente los puntos de destino fijados por el usuario. Para ello el robot ha de realizar una navegación deliberativa, haciendo frente a errores de navegación y lectura de sensores, siendo capaz de detectar cuando se ha perdido y rectificando su itinerario para alcanzar satisfactoriamente el punto de destino. Adicionalmente se ha de mostrar información al usuario para seguir la simulación y se debe representar en un entorno 3D la navegación del robot.

Los objetivos más concretos que motivan esta tesis son:

- Programación de grafos. Se ha de generar la estructura de grafo con los siguientes elementos: nodos, túneles, segmentos de túneles y posibles objetos de interés presentes en nodos y túneles, atendiendo a las características de cada uno. Los grafos contendrán información especifica de cada elemento y se han de poder generar de forma aleatoria o bien crear a partir de la información proporcionada por un archivo de texto.

- Programación de un algoritmo de búsqueda en grafos. Este ha de ser rápido y eficiente, ofreciendo siempre la ruta más corta entre el punto origen y el punto destino seleccionado por el usuario.

- Programar un modo de exploración, mediante el cual el robot navegue por el mapa topológico simulado en forma de grafo, añadiendo ruido a las lecturas de los sensores. Se forzará al robot a equivocarse de túneles y este tendrá que detectar la equivocación y corregir su situación estimada, así como su itinerario hacia el destino.

- Capacitar al robot simulado de reconocer objetos de interés en el trayecto de su exploración. Esto servirá para contrastar la posición conocida del objeto reconocido con la posición actual del robot según su programa de navegación.

- Brindar al robot la capacidad de relocalizarse, bien tras haberse perdido o desde un inicio en una zona desconocida. Para ello hará uso de las lecturas de sus sensores y buscará la localización más probable según el mapa topológico memorizado.

- Proporcionar información al usuario de la posición y las decisiones del robot, para así poder monitorizar la ejecución del programa. Adicionalmente se deberá mostrar en un entorno 3D el mapa en forma de grafo, la ruta planificada y la posición del robot para poder monitorizar su navegación de forma visual.

1.3. Estructura del documento

El contenido de este trabajo se encuentra distribuido en varios capítulos, los cuales se describirán a continuación.

En el capítulo 2 se habla de la navegación de robots, de las últimas técnicas para el desarrollo de mapas topológicos, y de los mejores algoritmos de búsqueda en grafos. Expone el estado del arte en las distintas metodologías que se van a aplicar en este proyecto, ofreciendo al lector una idea generalizada de las metodologías más usadas y sus características.

En el capítulo 3 se realiza la descripción de la arquitectura desarrollada. Se presentará cada uno de los niveles del programa, explicando detalladamente los procedimientos, así como la interacción entre las partes.

En el capítulo 4 se muestran los resultados obtenidos de las simulaciones tras poner a prueba la arquitectura presentada en este trabajo ante diversas situaciones. Se realiza a su discusión y análisis.

En el capítulo 5 se discuten las conclusiones del trabajo, se comenta desde una vista general todo el proyecto así como sus posibles impactos, responsabilidades y líneas futuras.

Por último, en el capítulo 6, se trata el estudio económico del proyecto, así como la organización y la distribución temporal del proyecto.

Capítulo 2

Estado del arte

En este capítulo se van a estudiar las diversas líneas que tienen influencia hoy en día para resolver los principales problemas que este trabajo plantea. Primeramente se estudiará el contexto de por qué se usan los mapas topológicos, distintas formas de generarlos y su uso.

Adicionalmente, se estudiarán conceptos interesantes de navegación en mapas topológicos.

Finalmente se exponen diversos algoritmos de búsqueda en grafos, comparándolos entre si y exponiendo sus aplicaciones.

2.1. Desarrollo de mapas topológicos

El módulo del proyecto el cual proporciona la información del entorno al módulo de navegación que se desarrolla en este trabajo obtiene los datos del entorno mediante la técnica de SLAM (*Simultaneous localization and mapping*). Gran parte de los sistemas SLAM de hoy en día generan mapas métricos o mapas basados en cuadrículas de ocupación, los cuales son altamente ineficientes para espacios grandes pues presentan dificultades a la hora de planificar y tienden a generar problemas de diversos niveles de complejidad. Cuando la representación es 3D, desemboca en espacios aun más grandes a tener en cuenta y que dificultan el cómputo en mayor medida.

Generalmente, a la hora de la navegación en espacios grandes, los mapas topológicos ganan en prestaciones frente a los mapas métricos ya que son más eficientes planificando rutas. También reducen el nivel de información a transmitir en caso de colaborar varios robots juntos. Como desventaja este tipo de representación de mapa presenta que a la

hora de localizar el robot, este se limita simplemente a los nodos más cercanos, lo que elimina casi toda la precisión de posicionamiento.

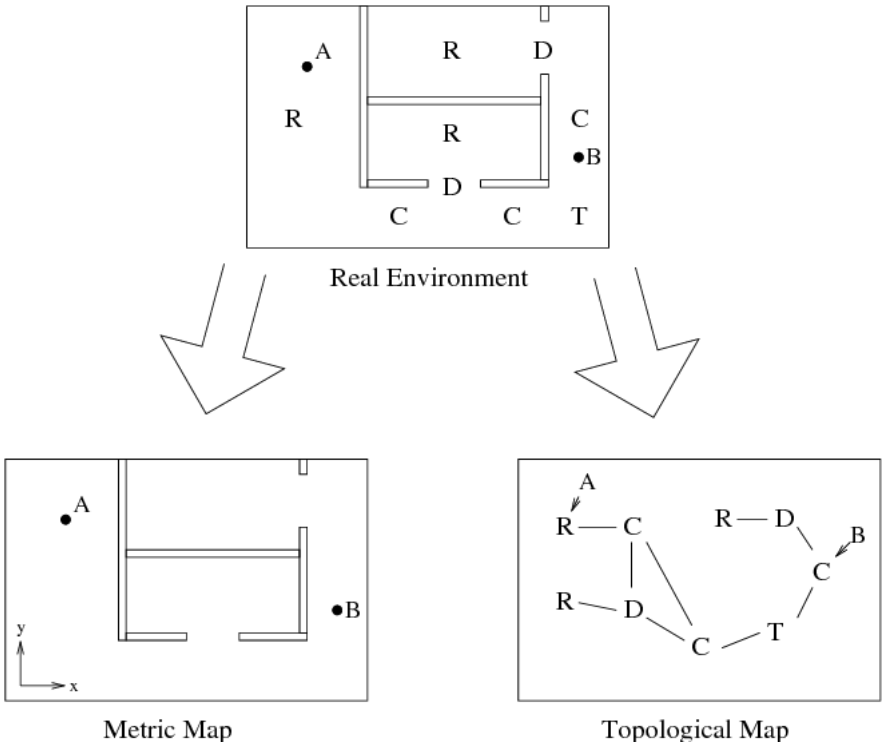


Figura 2.1 Diferencia de detalle entre mapa métrico (izquierda) y mapa topológico (derecha). Autor: J. Meyer.

Dado a sus ventajas y desventajas, algunos investigadores han tratado de crear sistemas híbridos para obtener la mejor parte de ambos métodos. El trabajo de Konolige y otros [8] usa una navegación híbrida combinando información métrica y topológica para mapas de gran escala, usando una red de nodos creada a base de poses en SLAM. De otra forma, Thrun [24] trata de lograr una planificación eficiente obteniendo un mapa topológico de un mapa métrico previo; construye un mapa de ocupación y luego una red de ruta usando Diagramas de Voronoi. En el capítulo 2 del trabajo escrito por Szabó [22] se compara la navegación métrica y topológica en detalle.

Existen varias formas de construir mapas topológicos, cada uno con sus ventajas e inconvenientes. Wallgrun [26] utiliza los famosos Diagramas de Voronoi Generalizados, el cual genera un camino en forma de red que se mantiene lo más lejos de los obstáculos posible, usado para interiores sobre todo. Otra técnica común es la construcción de diagramas mediante la triangulación de Delaunay, donde se crean redes mediante triángulos cuyos vértices pertenecen a circunferencias circunscritas.

Otro método de construcción de mapas topológicos es mediante grafos de visibilidad los cuales usa Lozano-Pérez [11], donde se forma un diagrama uniendo todos los vértices de los objetos que forman el mapa con aristas siempre y cuando no se interrumpan con ningún obstáculo.

Para imágenes, existen otros dos métodos, el *thinning* y el *skeletonization*, que básicamente tratan de obtener un diagrama mediante la eliminación capa a capa de los píxeles exteriores de imágenes binarizadas.

Dado a su gran ventaja en la navegación de robots, diversos proyectos se están centrando en la creación de mapas topológicos, como es el de Negre y otros [2]. En su trabajo, crean un mapa topológico mediante la navegación del robot submarino SPARUS II. Cada nodo se genera según el avance de este, mediante el conocimiento de la pose del robot y con ayuda de un sistema de visión estéreo usando Graph-SLAM.

2.2. Navegación en robots

El objetivo de la navegación en robots es mover el robot de tal forma que alcance una posición objetivo. Para ello, ha de localizarse en el mapa y tomar decisiones a medida que avanza para poder llegar correctamente a su objetivo. Toman parte el problema de localización, comprobando la posición del robot con la posición del mapa conocido, la generación de rutas entre la posición del robot y la posición de destino y la exploración mediante el movimiento del robot.

En esta sección se exponen distintas metodologías que son usadas en las distintas partes de la navegación del robot, las cuales puestas en conjunto logran que el robot llegue a su destino.

En cuanto a la navegación en entornos acuáticos se refiere, en los pasados años casi todo ese tipo de tareas la han desarrollado robots teleoperados, por lo que no existe mucha información al respecto en este campo específico. Sin embargo, la implementación de la autonomía en robots exploradores en condiciones submarinas se encuentra actualmente en auge.

Relacionando la navegación con las características del mapa topológico, Portugal y otros [15] destacan la importancia de la extracción de características del entorno y su asociación al grafo para realizar la navegación. En su trabajo, su algoritmo asocia un ID a cada nodo, así como sus coordenadas, su número de salidas y la orientación y forma de

cada una de sus salidas. Esto facilita la identificación de los nodos así como la posterior navegación.

En cuanto a la localización, Praczyk [16] en su trabajo habla de cuales son actualmente los tres métodos que utilizan los robots submarinos para localizarse.

- El primero de ellos es mediante posicionamiento acústico; pero para este caso se necesita de un punto de referencia para comparar, como puede ser una baliza submarina. El sistema más preciso de este tipo actualmente es el Long BaseLine (LBL), disponiéndose de la misma tecnología en su versión corta (SBL) y ultracorta (USBL).
- El el caso de no poder instalar balizas, el autor comenta la posibilidad de usar la navegación geofísica, la cual usa SLAM (como se mencionó anteriormente) para identificar características físicas del medio y poder situarse. Este método tiene las desventajas de que hace al robot más pesado, grande y costoso, lo que en muchas ocasiones dificulta el desarrollo de la navegación
- Como tercera técnica de posicionamiento bajo el agua, el autor expone la navegación inercial. Este método usa el conocimiento de la orientación y la velocidad del robot para conocer su posición. Para conocer la velocidad del robot se suelen usar distintos elementos que usan distintas técnicas, que pueden ser mecánicos, electromagnéticos, de presión y Doppler. Se destaca el Doppler por su precisión incluso a bajas velocidades. Una alternativa al uso de estos elementos es el uso de la odometría lo que reduce bastante los costes de implementación.

Una vez localizado, Praczyk usa una red neuro-evolutiva llamada Assembler Encoding with Evolvable Operations (AEEO) mediante la cual procesa los datos de posición, usándolos para la navegación del robot.

Ya que el robot se orienta respecto a un mapa topológico, se requiere asociar su posición a un nodo de este. Para lograr dicha asociación existen diversas técnicas, como la que plantea Sören y otros [20], quienes establecen en su trabajo que para comparar el entorno con un nodo del mapa topológico se han de comparar los vértices adyacentes. Advierte de que comparar la posición de los nodos y el robot puede no ser una buena idea debido a que puede haber grandes errores en la estimación de posición. En su defecto, propone comparar orientaciones relativas de las salidas de los nodos o características puramente topológicas como el grado de cada nodo (número de salidas) usando para ello sensores en el robot. Propone crear un índice de similitud para evaluar las comparaciones entre la posición del robot y los nodos y usar valores normalizados.

Habla también de la propagación de corrección de errores, basándose en que si un nodo resulta con un índice de similitud alto, los nodos adyacentes deberán tener también

un índice de similitud alto.

Otro planteamiento que propone el autor es la creación de un grafo por parte del robot a medida que explora. Este se compara con con el grafo del mapa conocido buscando similitudes entre grafos creando un problema de isomorfismo. Expone también la posibilidad más heurística de crear un grafo que vaya aumentando nodos vecinos de forma recursiva, el cual usa también la similitud para guiar su búsqueda.

Para cuando el robot se pierde, Jan y otros [6] hablan del modo de *kidnapped robot* que viene a ser un modo secuestro, donde el robot no sabe donde se encuentra y ha de encontrarse dentro del mapa. Para encontrarse, en este caso el robot hace uso de la odometría en las cercanías de su posición inicial, computando posibles nodos cercanos con reconocimiento del entorno mediante sensores. Entonces se interpola datos para buscar la solución más probable.

En su trabajo, hace uso de cuatro módulos, cada uno siendo un nodo de ROS y comunicándose entre ellos para lograr la localización mientras el robot navega:

- 1. Extracción de características.
- 2. Reconocimiento del lugar.
- 3. Estimación de conexiones.
- 4. Optimización del grafo.

Para la planificación de la ruta en la navegación existen distintos planteamientos. Por ejemplo Kurt en su trabajo [8] expone cómo a cada arista del grafo se le ha asociado un peso dependiendo de la distancia de esta, lo cual ayuda a la hora de la planificación. En su caso, realiza la planificación con un algoritmo Dijkstra. Primero selecciona una posición inicial para el robot, una posición objetivo y posteriormente se realiza la planificación. Una vez se tiene la ruta, usa un nodo de ROS que permite la navegación mediante el uso de sensores.

Adicionalmente, muestra como es posible aumentar nuevos nodos al grafo existente mediante el uso de la pose del robot y el enlace con nodos adyacentes.

Kostavelis y otros [9] realiza la navegación usando un LSTM (*Labeled Sparse Topological Map*) junto a un ANG (*Augmented Navigation Graph*), el cual contiene información a más alto nivel y sirve de intermediario entre las directrices del usuario mediante una interfaz y el mapa LSTM. Mediante un algoritmo Dijkstra obtiene la secuencia de nodos que el robot ha de atravesar. En este caso, cada vértice tiene una nube de puntos clave asociada, obtenida mediante (*iterative closest point*) ICP; desde la posición de cada

nodo, el robot observa determinados puntos característicos que identifican la posición. Esta nube de puntos es usada para la navegación, ya que el robot se localiza en base a esta información entre todos los nodos del grafo. Así, el robot va pasando por cada nodo planeado en la ruta a seguir, hasta llegar a su objetivo.

Nasiruddin [12] expone las distintas arquitecturas de control de navegación en el caso de robots submarinos, evaluando las ventajas y desventajas de cada una de ellas, su flexibilidad, la facilidad de integración entre sensores y acciones y su coste computacional. Expone la existencia de cuatro tipos de control: jerárquico, heterárquico, híbrido y basado en el comportamiento.

- El control jerárquico enfatiza el razonamiento y la realización de predicciones sobre el entorno mediante elementos deliberativos.
- El control heterárquico se compone de elementos reactivos que se basan en el principio de acción-reacción.
- El control basado en comportamientos evalúa algunos estados cercanos pero solo se desarrolla cuando avanza. Combina elementos reactivos y deliberativos.
- El control híbrido combina las tres clases de control mencionadas anteriormente. Usa un planificador de estados y realiza predicciones, reaccionando a corto y largo plazo ante las situaciones que se presentan. Algunos ejemplos de arquitecturas que usan este tipo de control son *Distributed Architecture for Mobile Navigation* (DAMN) y *A Three-Layer Architecture for Navigating Through Intricate Situations* (ATLANTIS).

Junto a toda esta mención de trabajos y metodologías, resulta imprescindible también destacar los avances que realiza el grupo del CAR en la Universidad Politécnica de Madrid en el cual se encuentran entre otros, Fernando Matía y Paloma de la Puente, quienes han publicado trabajos excepcionales sobre la localización y navegación en robots.

2.3. Algoritmos de búsqueda en grafos

Existen una gran multitud de algoritmos que realizan búsquedas en grafos. Todos ellos se focalizan en encontrar una ruta que una dos nodos del grafo y cada uno usa métodos distintos según sus prioridades.

Nótese que se considera que el grafo se ha obtenido de forma que el robot siempre vaya por el centro del entorno, lejos de los obstáculos, por lo que únicamente lo que se busca

mediante la búsqueda de ruta en el grafo es la secuencia de nodos a seguir, considerando que el robot será capaz de navegar por dicha ruta sin ningún tipo de problema.

A continuación se describen algunos de las metodologías más usadas:

Algoritmo Depth-First Search

Este algoritmo trata de explorar el grafo desde un nodo inicial. Elige un camino determinado, y va avanzando por este hasta llegar a que ya no quedan nodos por explorar por ese camino, es entonces cuando retrocede por el camino recorrido hasta que encuentra un nodo al que poder ir y que no está explorado, y vuelve a repetir la misma operación.

El algoritmo explora todo el grafo hasta que encuentra el nodo objetivo. Para su ejecución, hace uso de almacenamiento de la información en forma de pila, usando el método *Last in first out.*

Algoritmo Breadth-First Search

A diferencia del *Depth-First Search*, este tipo de algoritmo explora el grafo a lo ancho. Parte de un nodo inicial y evalúa todos nos nodos adyacentes hasta haberlos evaluado a todos. Esto se repite para cada uno de los nodos adyacentes una vez se han evaluado todos, evaluando los nodos adyacentes a estos. Este algoritmo busca encontrar el nodo objetivo alejándose lo menos posible del nodo inicial, suponiendo que la distancia entre nodos es igual siempre. El algoritmo busca en todo el grafo hasta dar con el nodo objetivo.

Hace uso de almacenamiento de la información en forma de pila, usando el método *First in first out.*

Algoritmo Dijkstra

Este algoritmo hace uso de pesos en las aristas entre nodos del grafo. Conociendo los pesos, trata de buscar el camino más corto entre dos nodos. Para ello va explorando todos los caminos más cortos que van desde el nodo origen hasta todos los demás nodos. El algoritmo va explorando todo el grafo y a medida que lo va haciendo puede actualizar las rutas más cortas para llegar a determinados nodos.

Este algoritmo va expandiéndose en todas direcciones hasta dar con el nodo objetivo, para el cual tendrá la ruta más corta teniendo en cuenta los pesos.

Algoritmo Greedy-Best-First

El algoritmo *Greedy-Best-First* es un algoritmo voraz, que se basa en partir del nodo origen y explorar el grafo según el coste de una función heurística que informa al algoritmo en qué dirección explorar. Avanza por el grafo explorando siempre el nodo de menor coste hasta dar con el nodo objetivo.

Algoritmo A*

Este algoritmo es como el Dijkstra, pero se apoya en una función que le informa de qué camino es mejor para explorar, la función heurística. Teniendo en cuenta el peso de las aristas, se explora de tal forma que se minimice el valor de la función coste:

$$f(n) = g(n) + h(n) \tag{2.1}$$

Donde $f(n)$ expresa el coste de la ruta hacia el nodo n, $g(n)$ es el coste hasta dicho nodo y $h(n)$ es la función heurística.

La función heurística puede tratarse desde simplemente la distancia euclidiana entre nodos hasta estar compuesta por un conglomerado de funciones más complejas.

Tabla 2.1 Comparación entre algoritmos de búsqueda en grafos.

Algoritmos sin pesos	
Depth-First Search	*Last in first out*, no óptimo y serpenteante
Breadth-First Search	*First in first out*, óptimo pero lento
Greedy-Best-First	Voraz pero tendencia a atascarse
Algoritmos con pesos	
Dijkstra	Explora en orden creciente de coste, óptimo pero un poco lento
A*	Óptimo y rápido

Capítulo 3

Metodología

En este capítulo, inicialmente se describe la estructura de grafo programada, así como todos sus elementos y las características asociadas a estos.

A continuación se expone detalladamente los algoritmos que interactúan con la estructura de grafo. Primero se habla del algoritmo de búsqueda en grafos, para luego continuar con el algoritmo de navegación, con todas las partes de las que está compuesto.

Por último, se habla de cómo se realiza la monitorización de los datos por parte del usuario final.

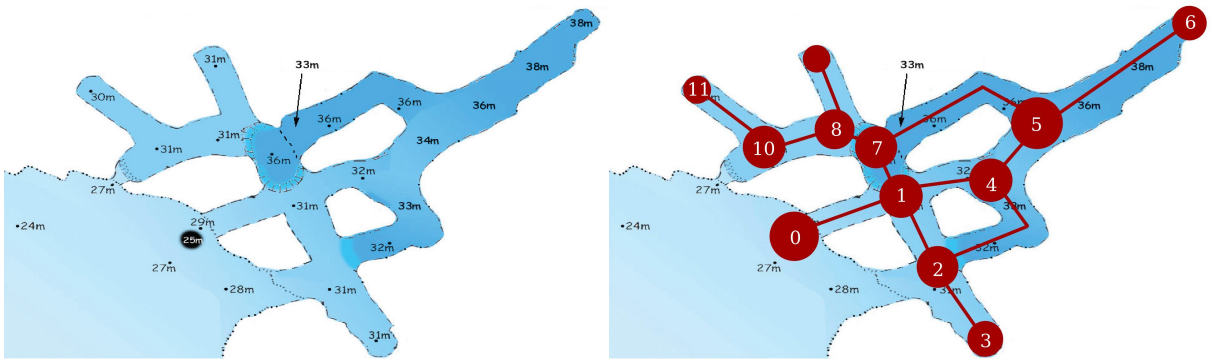
En cuanto al desarrollo del proyecto, todo el código se ha programado en el lenguaje C++, en la plataforma *QtCreator*. El trabajo se ha construido sobre un nodo en ROS, mediante el cual se comunica con otros nodos como el de la herramienta Rviz, encargada de hacer la representación 3D del grafo.

3.1. Mapeado topológico

Como se comentó en el estado del arte, existen numerosas formas de transformar un espacio en un mapa topológico. Este proceso facilita enormemente la tarea de navegación del robot.

En el caso de este trabajo, se parte ya de un mapa topológico conocido, formado por nodos predefinidos, enlazados con aristas. Cada nodo posee información única.

Se crea entonces del mapa topológico conocido, un grafo. El grafo, en este caso, se



(a) **(b)**

Figura 3.1 El mapa de la cueva Kaatiala el cual se corresponde a la imagen (a), es interpretado en forma de grafo (b), compuesto por nodos y aristas que representan el entorno del mapa.

trata de un grafo simple, conexo (para toda pareja de vértices (u, v) existe un camino de u a v) y no dirigido.

El grafo se expresa tal que:

$$G = (V, E) \tag{3.1}$$

donde $E \subseteq VxV$, siendo V los vértices y E las aristas.

Las características del entorno se proyectan entonces en el grafo, compuesto por vértices y aristas que los interconectan según la configuración del mapa. Para el propósito de la exploración de minas, se ha decidido que los nodos representen convergencias entre dos o más túneles, y que las aristas representen los túneles.

De forma general, en una convergencia de túneles, suele crearse un espacio más grande, con unas características más destacadas. Adicionalmente, al ser un espacio más amplio, el robot dispone de más espacio para realizar maniobras de giros y detección de túneles en las intersecciones (nodos).

Además de por nodos y túneles, al grafo se le ha brindado de más elementos, como segmentos que componen los túneles. Esto es porque se ha detectado la existencia de túneles más extensos que otros, y dada su extensión se encuentran distintas dimensiones de sección a lo largo de su recorrido. La división de un túnel en segmentos permite caracterizar este de forma más cercana a la realidad.

En el caso de que un túnel tenga un estrechamiento en su punto medio, o un saliente en cualquier otro punto es fácilmente caracterizable e interpretable por el robot si ese tramo se considera como un segmento distinto al resto de tramo que compone el túnel.

Con estos elementos se dispondría de un grafo que transmite bastante bien las carac-

terísticas del entorno al robot y permite su completa navegabilidad. Para añadir algún elemento adicional que permita al robot confirmar su posición, más allá de la detección de sensores, se ha añadido el elemento de objetos de interés al grafo. Estos objetos de interés permitirán al robot obtener su posición exacta al ser detectados e identificados mediante un algoritmo de visión por computador, contrastándolo con su posición estimada y así pudiendo rectificar la ruta en caso necesario.

Para la creación de la estructura de grafo, el algoritmo se apoya en una lista de adyacencia. Esta está compuesta por un vector de vectores. Un vector principal, alberga todos los nodos existentes en el grafo, y para cada nodo, existe un vector que almacena los nodos a los cuales se encuentra conectado dicho nodo, además de la distancia a la que se encuentran. Esto se muestra en la siguiente imagen, que corresponde a una parte de la monitorización de la creación del grafo mediante la ventana de comandos, donde se observa la lista de adyacencia de un grafo almacenado en el algoritmo.

En ella se interpreta las conexiones, por ejemplo, el nodo 2 está conectado con el nodo 3 a una distancia de 6 metros, al nodo 6 a una distancia de 17 metros y al nodo 4 a una distancia de 10 metros.

Terminal:

```
------Node adjacency list------

Node --> (LinkedNode, Distance)

0 --> (1,1)
1 --> (0,1) (6,19) (5,20)
2 --> (3,6) (6,17) (4,10)
3 --> (2,6) (7,15) (6,20)
4 --> (2,10) (6,7) (7,2) (8,6)
5 --> (1,20) (9,10) (8,7)
6 --> (1,19) (2,17) (3,20) (4,7)
7 --> (3,15) (4,2) (8,4) (9,4)
8 --> (5,7) (7,4) (4,6) (9,6)
9 --> (5,10) (8,6) (7,4)
```

A continuación se procede a hablar más en detalle de cada uno de los elementos que componen el mapa topológico en forma de grafo.

3.1.1. Nodos

Como se enunció anteriormente, los puntos de convergencia entre túneles (aristas del grafo) se han considerado como nodos.

Los nodos son puntos del entorno de dimensiones mucho más amplias que las de los túneles. En un mismo nodo pueden acabar varios túneles, y cada desembocadura de túnel

en el nodo tiene unas características concretas, que permiten la identificación del túnel y el nodo al que conducen.

Con el fin de identificar los nodos, a cada nodo se le ha asociado un número de nodo. El orden de numeración sin embargo no tiene relevancia, simplemente atiende al orden de registro de los nodos en la estructura de grafo. Así como un número, se le ha asociado a cada nodo un nombre. El nombre por defecto es "N-X", siendo X el número de nodo correspondiente. La decisión de añadir nombres a los nodos viene de la monitorización que realiza el usuario, pues nodos característicos como el nodo de la entrada a la mina, o nodos destacados como por ejemplo puede ser un espacio de almacenamiento pueden ser así fácilmente identificados en vez de atender a su numeración.

Las dimensiones de cada nodo (altura, anchura y profundidad) están también registradas como características, pues se considera un dato de vital importancia y relevancia, ya sea para considerar las dimensiones del robot a la hora de navegar por él, o con fin informativo para los datos de exploración.

Una de las características principales de los nodos, es entonces el número de salidas que posee. Una vez conocido el número de salidas, a cada salida se le asocia unas características.

Estas salidas (desembocaduras de túnel) son analizadas por los sensores del robot, el cual hace un barrido completo del nodo. Así se obtienen la orientación de la salida de cada túnel estando compuesta por dos coordenadas: orientación horizontal y orientación vertical.

La orientación siempre es consistente, ya que se tiene el mismo sistema de referencia, proporcionado por los giróscopos, brújulas y acelerómetros del robot. El sistema de referencia se considera el Norte, con inclinación del robot de 0 grados.

Aparte de la orientación de cada desembocadura de túnel, el robot registra las dimensiones de estas (altura y anchura), pues al ser una mina excavada de forma manual, cada una tiene una dimensión distinta, lo que permite su fácil identificación gracias a los sensores del robot.

Para cada salida también se registra el nodo al cual conduce ese túnel. Este dato es de vital importancia para llevar a cabo una correcta navegación.

Cada nodo ocupa una localización en el plano tridimensional, que toma como origen la entrada de la mina. Las tres coordenadas (x,y,z), que hacen referencia a cada una de las dimensiones del espacio, permiten la situación de cada nodo en el espacio y la representación del nodo en un entorno gráfico.

Los nodos pueden poseer (o no) los llamados objetos de interés, que permite al robot, en el caso de detectarlos, confirmar la identidad del nodo en el que se encuentra. Estos serán detallados más adelante en este mismo capítulo.

Existen nodos denominados desconocidos. Los nodos desconocidos son finales de túneles que no se ha seguido explorando, o puntos en los que se conoce que la mina sigue, pero el mapa informado no contempla. El robot conoce inicialmente los nodos desconocidos.

Concretando, las características que poseen los nodos en el grafo son:

- **Número de nodo**.
- **Nombre de nodo**.
- **Altura**.
- **Anchura**.
- **Profundidad**.
- **Posición tridimensional**. Coordenadas (x,y,z), que hacen referencia a cada una de las dimensiones del espacio. Permiten situar el nodo en el espacio.
- **Número de salidas**. Para cada salida:
  - **Dimensiones de la salida**. Altura y anchura de la desembocadura del túnel.
  - **Orientación vertical**. Orientación en grados respecto al eje vertical del sistema de referencia.
  - **Orientación horizontal**. Orientación en grados respecto al eje horizontal del sistema de referencia.
  - **Nodo conectado**. Nodo al cual se llega tras seguir el túnel de esa salida.
- **Objeto de interés**. Pueden existir en el nodo o no. Permiten la identificación del nodo.

3.1.2. Túneles

Las aristas que forman el grafo están directamente asociadas a los túneles de la mina, que interconectan los nodos. Estos túneles varían en forma y longitud, pues son túneles generalmente excavados de forma manual, y en numerosas ocasiones dependen de la densidad del material, por lo que pueden tener diversas desviaciones.

Los túneles, al igual que los nodos, han de numerarse para su identificación. El orden de numeración tampoco es relevante y depende del orden de introducción de los datos.

El conocimiento de las características del túnel resulta crítico para la navegación del robot. En situaciones en las que el túnel tenga alguna dimensión inferior que la envergadura del robot explorador, este no podrá pasar por él. En dicho supuesto se debería documentar la existencia de ese túnel pero no podrá ser explorado. Es por eso por lo que una de las características principales que se le atribuyen al elemento túnel dentro del grafo son sus dimensiones, tanto el alto como el ancho.

Otro dato imprescindible que ha de acompañar a los elementos de túnel son los nodos que comunican, es decir, el nodo adyacente a cada extremo del túnel. Esto permite, conociendo los nodos, realizar todo el entramado de conexiones entre estos y dar así forma al grafo que compone el mapa topológico del entorno.

La longitud del túnel forma un papel importante a la hora de la navegación, es por eso por lo que también queda registrada esa característica. Si bien los túneles pueden no formar una línea recta, esta característica no se verá afectada.

Al igual que en el caso de los nodos, los túneles pueden poseer elementos destacados, reconocibles por el sistema de visión por computador del robot, los denominados objetos de interés.

De acuerdo a lo mencionado en apartados anteriores, un túnel está formado a su vez por un conjunto de segmentos. El número de segmentos que compone cada túnel también queda registrado como característica. Además, cada segmento queda directamente enlazado con el túnel del cual forma parte. Dicho elemento se describe a continuación.

Concretando, las características que poseen los túneles en el grafo son:

- **Número de túnel**.
- **Nodos que conecta**. Nodos que se encuentran a cada extremo del túnel.
- **Altura**.
- **Anchura**.
- **Longitud**.
- **Número de segmentos**.
    - Información de cada segmento.

- **Objeto de interés**. Pueden existir en el nodo o no. Permiten la identificación del nodo.

Segmentos de túnel

El elemento túnel presenta visibles carencias a la hora de representar un terreno de trayecto irregular que conecta dos nodos. Un túnel con un estrechamiento en alguna parte o con desviaciones o cambios de forma sería realmente difícil de reportar con un solo elemento. Para solventar dichas situaciones se decidió crear el elemento segmento, el cual compondría a los túneles.

Básicamente, un segmento se podría definir en este caso y para este fin, como un túnel de menor longitud, que concatenado con otros segmentos o simplemente él mismo, es capaz de recoger de forma más detallada las características de un túnel.

Además de las características dimensionales que registra un túnel, el segmento almacena el valor de la pendiente que tiene el tramo sobre la referencia de la componente horizontal. Este dato permite al robot ajustar los parámetros de desplazamiento, además de servir para una representación más detallada en tres dimensiones del entorno.

Cada segmento es numerado, para poder ser identificado. La nomenclatura atiende a "T-X", siendo T el número identificativo del túnel al que pertenece, y X el número asociado al segmento respecto al número de segmentos que componen el túnel.

Por ejemplo, en el túnel 4, el segmento 2 de todos los 3 segmentos que lo componen quedaría denominado como: "4-2".

Concretando, las características que poseen los segmentos en el grafo son:

- **Número de segmento**.
- **Túnel al que pertenece**.
- **Altura**.
- **Anchura**.
- **Longitud**.
- **Pendiente**.

3.1.3. Objetos de interés

Los objetos de interés (OI), son elementos destacados del entorno, fácilmente reconocibles y que tienen asociada, en este contexto, una ubicación precisa en el mapa conocido inicialmente por el robot.

Se introdujeron para dar dinamismo a la navegación, a la vez que sirven para confirmar la posición del robot en el mapa al ser detectados. El robot compara su posición estimada con la posición del objeto de interés detectado, si coinciden entonces el robot va por el camino correcto, por el contrario si son distintas, significa que el robot ha errado y se ha de rectificar. Esto afecta directamente al valor de incertidumbre de posición asociado al robot según su avance, de tal forma que al confirmar su posición mediante el OI, este valor se reinicia.

Un ejemplo de OI podría ser una carretilla abandonada en un túnel de la mina, o un pico, objetos fácilmente detectables por su forma tan característica que destaca sobre las formas de las paredes del túnel o nodo.

La detección de los objetos de interés se realiza por un sistema de visión por computador incorporado ya por el robot, el cual analiza formas y características de los objetos que encuentra por su camino y los coteja con una base de datos que posee desde el inicio, pues es cargada junto con el mapa del entorno. En esta base de datos, cada OI tiene asociado un número de identificación (de ahora en adelante referido como ID), y su posición. Los OI solo se pueden encontrar en nodos o en túneles, así que cada ID tiene asociado el número de nodo o túnel en el que se encuentra.

Con fin de diferenciar la numeración de nodo o túnel, se ha decidido simplificar el problema de una sencilla forma. Los ID de los OI presentes en nodos se encuentran en el rango [0, número de nodos·2], y el ID de los nodos presentes en los túneles en el rango [númerodenodos·2, ∞] , siendo el númerodenodos el número total de nodos presentes en el grafo (dato conocido). Esto permite identificar si el OI pertenece a un nodo o a un túnel simplemente observando el rango en el que se encuentra su número de ID.

La Tabla 3.1, ejemplifica cómo sería una tabla de objetos de interés mediante la cual el robot compara su posición. Se observa lo anteriormente descrito, para un total de diez nodos en un grafo, los ID quedan claramente identificados perteneciendo los de menor valor a nodos y los de mayor valor a túneles. Por ejemplo en OI con ID: 2 se encuentra en el nodo 8, así como el OI con ID: 22 se encuentra en el túnel 7.

Tabla 3.1 Tabla ejemplo de objetos de interés

ID	Posición
0	1
1	5
2	8
21	3
22	7
23	11

Para añadir realismo a la simulación, se ha establecido una probabilidad de detección de los OI (por defecto, del 90 %), de tal forma que el robot puede pasar por delante de un objeto reconocible pero no detectarlo debido a cualquier tipo de anomalía.

Facilitando la monitorización del usuario, se ha añadido la posibilidad de asociar nombres a los OI para no tener que consultar la tabla de OI en busca de la correspondencia objeto-ID.

Con fin de añadir más referencias, se puede registrar la distancia del comienzo del túnel hasta el OI, siempre teniendo en cuenta a un sentido del túnel, definido por el orden de los nodos conectados a ambos extremos de este. También es posible registrar la orientación para el caso de los OI situados en los nodos.

Concretando, las características que poseen los objetos de interés en el grafo son:

- **ID**. Número identificativo.
- **Posición**. Referencia a la numeración del nodo o túnel asociado.
- **Nombre**. Alias para facilitar la identificación
- **Distancia**. Distancia respecto al inicio del túnel.
- **Orientación**. Orientación respecto al sistema de referencia del nodo.

3.1.4. Generación de mapas informados

A la hora de obtener los datos del entorno para generar el grafo, se optó por permitir al usuario configurar estos datos en un archivo de texto, y que el algoritmo se encargase de leer dichos datos e identificar a que elemento pertenecen para asociarlos a este.

Esto permite aplicar el algoritmo a situaciones reales, con datos precisos sacados de mapas de minas. Como ejemplo, se ha usado la mina Kaatiala, situada en Finlandia, la

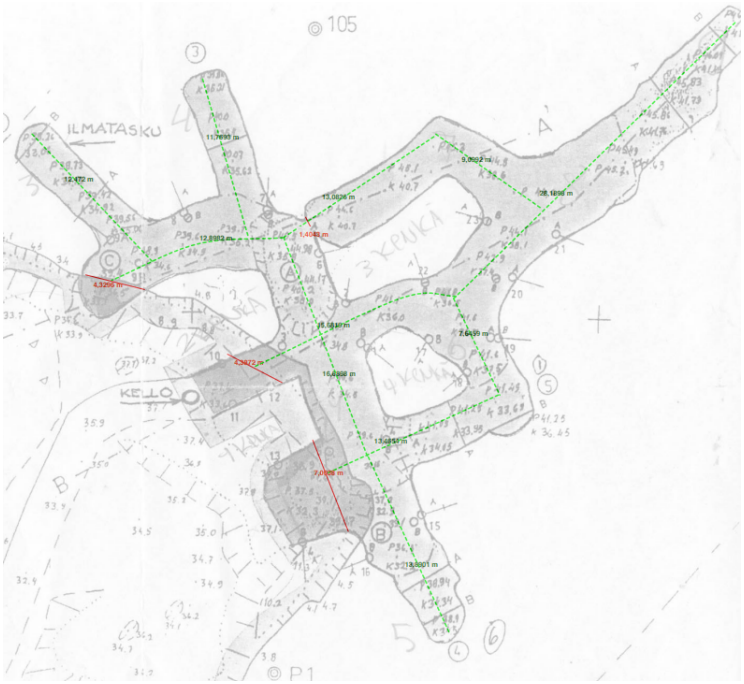


Figura 3.2 Mapa de la mina Kaatiala, Finlandia.

cual se encuentra actualmente inundada y fue mencionada en capítulos anteriores. En la Figura 3.2 se muestra el mapa de esta, acotado, de donde se pueden extraer todos los datos y dimensiones de los túneles y nodos, así como la configuración de conexiones.

Al llevar la ejecución del algoritmo programado en este trabajo en entornos reales, se pueden realizar mediciones de recorridos, encontrar rápidamente la ruta más corta entre dos puntos del mapa real, así como simular la navegación del robot en este.

En cuanto a la configuración del archivo de texto, este se ha diseñado de tal forma que los datos han de ser escritos en un orden específico. Primero se definen todos los nodos del mapa, posteriormente se definen todos los túneles con sus segmentos y finalmente los OI. El algoritmo es capaz de detectar cuando se termina de definir los nodos y se comienza a definir los túneles, sin importar el número de nodos definidos, lo que ofrece comodidad en la introducción de datos.

La secuencia de introducción de los datos se muestra en lineas que actúan de cabecera, mostrando el significado del orden de los datos.

A continuación, se muestra un ejemplo del archivo de texto de introducción de datos, al cual se le ha reducido el número de nodos y de túneles para acortar su longitud, ya

que el objetivo es ilustrar al lector la forma de la secuencia de introducción de datos, y entender las líneas de cabecera.

Conocida la estructura de introducción de datos, se puede hacer uso de programas que, conteniendo la información en bases de datos, la escriban en archivos de texto siguiendo la estructura requerida para poder ser leídos por el algoritmo de este trabajo.

El objetivo es disponer de numerosos archivos de texto que describan el entorno de distintas minas, para así poder hacer uso del potencial de este trabajo.

Los nodos se definirán mediante el siguiente bloque secuencia:

Nombre del nodo - Número de nodo - Número de salidas del nodo - Altura - Anchura - Profundidad - Coordenada X - Coordenada Y - Coordenada Z.

Para cada salida:

Nodo al que conecta - Altura - Anchura - Orientación horizontal - Orientación Vertical.

Los túneles se definirán mediante el siguiente bloque secuencia:

Número de túnel - Nodo de origen - Nodo de destino - Número de segmentos

Para cada segmento:

Longitud - Anchura - Altura - Pendiente

Para los OOI:

ID - Número de nodo o túnel - Identificador de si está en un nodo (0) o si está en un túnel (1).

```
//Node nodenumber numberofexits height width depth x y z //ConnectedNode height width HorizontalOrientation VerticalOrientation

Entry 0 1 4 3 1 0.5 1 4
1 11 22 45 55

Main 1 4 3 4 5 -1 -3 1
0 33 44 63 66
5 55 66 56 66
2 77 88 24 62
9 22 33 46 75

N2 2 2 1 1 3 0 4 0.3
1 34 55 63 52.6
3 55 66 56 62

NoMoreNodesFLAG 0 0 0 0 0 0 0 0
//Tunnel startnode finalnode numberofsegments //EachSegmentlenght width height slope

Tunnel1 0 1 3
2 1 1 1
2 1 1 2
1 1 1 3

Tunnel2 1 2 3
1 1 1 1
1 1 1 2
1 1 1 3

NoMoreTunnelsFlag 0 0 0
//ID OOIposition 0node1tunel
1 2 0
21 4 1
3 4 0
```

3.1.5. Generador de mapas aleatorios

La necesidad de lanzar varias simulaciones con diversos mapas de distintas características promovió la creación de un generador de grafos aleatorio. Esto permite disponer de un grafo aleatorio sin necesidad de partir de un archivo de texto para cargar los datos.

Gracias al generador aleatorio de mapas en formas de grafo, es posible ejecutar múltiples simulaciones en distintos escenarios para evaluar las capacidades del algoritmo programado. Esta generación aleatoria proporciona situaciones únicas en las que el robot se pone a prueba, y ha ayudado muchísimo al desarrollo y a la corrección de errores del algoritmo de navegación.

Se dispone de un amplio muestreo de ejecuciones del programa gracias al generador de grafos aleatorios. Esto permite realizar el análisis de resultados y conclusiones con una mayor población muestral, lo que aporta resultados más concluyentes.

El algoritmo generador de grafos básicamente hace uso de la estructura de grafos programada, con todos sus elementos, y partiendo de un número de nodos, este genera aleatoriamente el resto de elementos.

Esta generación aleatoria atiende a unos parámetros, a los cuales el usuario tiene fácil acceso en la cabecera de los archivos de ejecución. Mediante estos parámetros, el usuario es capaz de caracterizar y modificar los grafos generados aleatoriamente.

Para una mejor comprensión del procedimiento de generación, se exponen a continuación los parámetros que el usuario puede cambiar, y sus valores por defecto:

- **Altura nodo**. Mínimo: 1.5 metro — Máximo: 4 metros.
- **Anchura nodo**. Mínimo: 1.5 metro — Máximo: 2 metros.
- **Profundidad nodo**. Mínimo: 1 metro — Máximo: 4 metros.

- **Altura salida de nodo**. Mínimo: 0.5 metro — Máximo: 1.5 metros.
- **Anchura salida de nodo**. Mínimo: 0.5 metro — Máximo: 1.5 metros.
- **Orientación vertical salida de nodo**. Mínimo: 0º— Máximo: 360º.
- **Orientación horizontal salida de nodo**. Mínimo: 0º— Máximo: 360º.

- **Número de salidas en cada nodo**. Mínimo: 2 salidas — Máximo: 4 salidas.

- **Número de segmentos por túnel**. Mínimo: 1 segmentos — Máximo: 4 segmentos.
- **Altura segmento**. Mínimo: 1 metro — Máximo: 2 metros.
- **Anchura segmento**. Mínimo: 0.5 metro — Máximo: 1.5 metros.
- **Longitud segmento**. Mínimo: 1 metro — Máximo: 8 metros.
- **Pendiente segmento**. Mínimo: 0 % — Máximo: 20 %.

- **Distancia de generación entre nodos eje X**. ±10 metros.
- **Distancia de generación entre nodos eje Y**. ±10 metros.
- **Distancia de generación entre nodos eje Z**. ±2 metros.

- **Probabilidad de generación de un OI**. 30 %.
- **Probabilidad de detección de un OI**. 90 %.

- **Probabilidad de generación de nodo desconocido**. 10 %.

Generación del grafo de forma resumida

Dado a que la explicación de la generación del grafo puede considerarse densa se procede a resumir los pasos antes de comentar la generación en detalle:

- Se pide al usuario que determine un número de nodos totales.
- Se generan los nodos dimensionados y numerados, cada uno cercano al anterior en el espacio y con probabilidad de poseer un OI o de ser un nodo desconocido. En caso de generarse un OI, este se añade a la tabla de OI.
- Se conexiona el nodo 0 y el nodo 1 mediante un túnel.
- Para el resto de nodos se genera un número de salidas aleatorio, y se conexiona con nodos adyacentes (5 nodos anteriores o 5 posteriores al actual). Para cada conexión se genera un túnel, un posible OI en él y sus propiedades . Para cada túnel se genera un número de segmentos aleatorio y sus propiedades.

- Durante la generación de cada túnel, el enlace se registra en una lista de adyacencia para posteriormente generar el grafo enlazando todos los elementos.

Generación del grafo de forma detallada

Inicialmente al elegir el modo de generación de grafo aleatorio, se le pide al usuario que introduzca el número total de nodos que desea que posea el grafo.

Partiendo de este número, se generan todos los nodos, recibiendo una numeración identificativa y un nombre por defecto. Las dimensiones del nodo (anchura, altura y profundidad) se generan de forma aleatoria dentro del rango definido por los parámetros *Altura nodo, Anchura nodo y Profundidad nodo*. Además, cada nodo recibe una posición en el espacio tridimensional, la cual es generada de la siguiente forma:

El nodo inicial se crea en torno al origen de coordenadas tridimensionales; cada coordenada se genera aleatoriamente dentro de un rango definido por el usuario en el parámetro *Distancia de generación entre nodos eje X*, en el caso del eje x por ejemplo. Así se generan las tres coordenadas de posición del nodo inicial. Por ejemplo, con centro en [0,0,0] (origen), y según los parámetros por defecto enunciados anteriormente un posible punto de generación para el primer nodo sería [5,-5,-0,11].

Para los nodos sucesivos, se efectúa el mismo procedimiento, sin embargo, los rangos se generan en torno al anterior nodo creado, no respecto al origen de coordenadas. Esto se ha diseñado así para que cada nodo nuevo se genere dentro de una elipsoide definida por los parámetros, siendo su centro el nodo actual.

Con este procedimiento se consigue un efecto de ordenación de nodos respecto a su numero identificativo, estando próximos los nodos con números cercanos. Esto facilita la comprensión de la ruta a la hora de realizar una búsqueda de ruta entre dos nodos, pues aparecen con números próximos, en vez de números aleatorios que pueden generar confusión al usuario durante la monitorización.

Durante la generación de cada nodo, cabe la probabilidad de que se genere un OI en dicho nodo. Esta probabilidad viene determinada por el parámetro *Probabilidad de generación de OI*. En el caso de generarse, un ID es asociado al OI, tal y como se explicó en el apartado 3.1.3. Una vez el objeto de interés tiene asociado un ID, se le asocia el número de nodo en el cual se encuentra, y ambos datos se añaden a la tabla contenedora de OI.

Cabe la posibilidad durante la generación del nodo de que se convierta en un nodo desconocido. La probabilidad de que esto ocurra viene definido por el parámetro *Probabilidad de generación de nodo desconocido*. En el caso de que se convierta en nodo desconocido,

este será marcado como tal y no ofrecerá información alguna.

Una vez generados todos los nodos, se procede a realizar las conexiones entre ellos, creando túneles que los unan. Inicialmente se crea el primer túnel, que une el nodo 0 y el nodo 1. Esto se hace porque se busca que el nodo 0 tenga una sola conexión. El túnel recibe una numeración y se registran los nodos de sus extremos (0 y 1). Cabe la posibilidad de que el túnel genere un OI, en caso afirmativo se procederá igual que en el caso de la generación de OI en nodos, pero en este caso con un ID superior al doble del número máximo de nodos.

Se genera entonces un número de segmentos que lo componen, de forma aleatoria entre los parámetros mínimo y máximo de *Número de segmentos por túnel* y se registra dicho dato. Se procede a generar los segmentos que componen el túnel, para cada segmento se generan los parámetros aleatorios dentro de los rangos definidos por el usuario para los segmentos y son enumerados y enlazados con el túnel. La longitud del túnel restante se obtiene de la suma de las longitudes generadas de los segmentos que lo componen. A continuación, se añade el enlace al vector de vectores que hace de lista de adyacencia, registrando los nodos unidos y la distancia entre estos.

Para los nodos del 1 al último, el procedimiento cambia. En cada nodo se genera un número aleatorio de salidas, que viene determinado por el rango del parámetro establecido por el usuario *Número de salidas en cada nodo*. Una vez que se tiene el número de salidas del nodo, este dato se registra y para cada salida se busca un nodo a conectar.

Este procedimiento se realiza buscando un nodo para conectar entre los 5 anteriores y 5 posteriores nodos. El nodo elegido ha de cumplir que no es un nodo desconocido, que dicho nodo no tiene ya el número máximo de salidas permitido y que no sea el mismo nodo (túnel a sí mismo). Esto se realiza para que las conexiones entre nodos se realicen entre nodos cercanos. Esto da sensación de que los nodos ya conectados, se fueron numerando siguiendo su orden de conexión, pues si se conectasen los nodos aleatoriamente independientemente de su cercanía daría demasiada sensación de caos y aleatoriedad al usuario que monitoriza la navegación.

Una vez obtenido el nodo a conectar, se crea un túnel y se registran sus datos, así como son generados también los segmentos que lo componen con el mismo procedimiento que el primer túnel que unía el nodo 0 y el nodo 1.

Al final de este procedimiento, se tiene una lista de adyacencia que contiene todas las conexiones entre nodos, y mediante la cual se genera el grafo, que enlaza todos los elementos.

La lista e información de nodos, túneles, segmentos y OI generados se puede mostrar en pantalla para la inspección por parte del usuario.

3.2. Planificador Dijkstra

Como planificador, se ha optado por utilizar un algoritmo Dikjstra. Se ha programado el algoritmo en C++, siguiendo el pseudocódigo. Su nombre viene dado por, Edsger Dijkstra, científico de computación que programó el algoritmo en 1959.

Ya que el mapa está dado en un grafo no dirigido, con distancia entre nodos conocida, este algoritmo resulta idóneo para encontrar la secuencia entre nodos más corta para ir de un nodo inicial a un nodo final.

Principalmente, lo que hace este algoritmo es generar dos listas. Una lista contiene los nodos que componen la ruta más corta, y otra lista que contiene los nodos que están sin explorar. El algoritmo va visitando nodo por nodo almacenando el camino más corto hasta él, hasta que llega al nodo destino. En el caso de encontrar un camino más corto hacia un nodo desde el nodo origen, el camino más corto se actualiza a al nuevo camino de menor coste. Se caracteriza por guiarse únicamente por el coste de la distancia entre nodos sin usar ningún tipo de función heurística que guíe al algoritmo sobre qué nodos son más propensos a ser solución que otros.

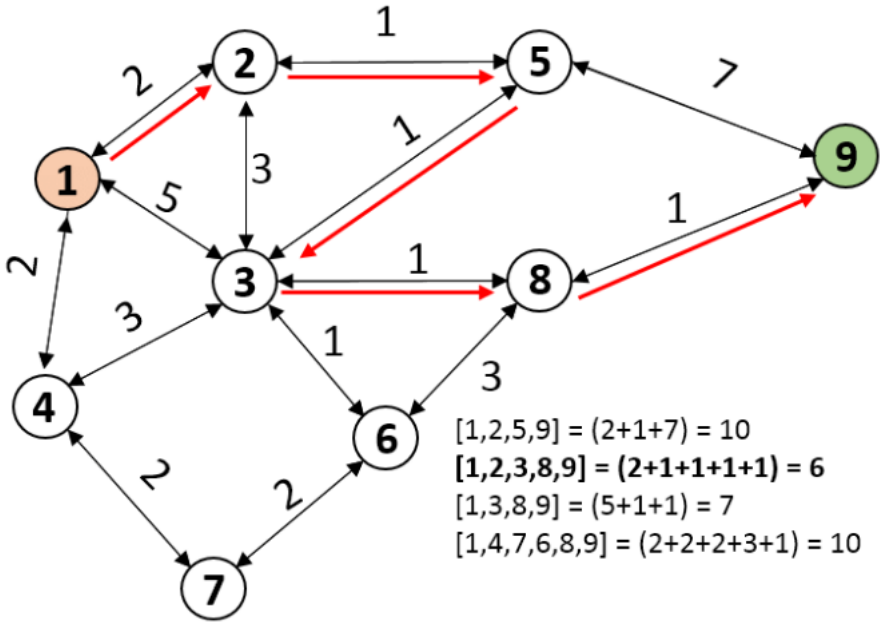


Figura 3.3 Funcionamiento de algoritmo Dijkstra. Fuente: `https://www.researchgate.net/figure/Illustration-of-Dijkstras-algorithm_fig1_331484960`

En la Figura 3.3 se muestra visualmente el funcionamiento del algoritmo. En esta caso el nodo 1 seria el nodo origen y el nodo 9 el nodo destino. Se observa como la ruta solución, pese a atravesar más nodos, resulta más rápida que otras rutas. El algoritmo ha evaluado todas las rutas para llegar hasta ese nodo, quedándose con la ruta más actualizada (la ruta de menor coste).

Este algoritmo suele usarse para tareas sencillas como el cálculo de enrutamiento en redes o rutas entre puntos con varios caminos, como rutas entre ciudades.

Aplicando el algoritmo al problema de este proyecto, dado el grafo, se proporciona un nodo origen y un nodo destino y se realiza una búsqueda del camino más corto entre ellos. Esto proporciona la secuencia de nodos que el robot ha de visitar.

Conocida la secuencia solución de nodos, se obtienen los túneles por los que el robot ha de pasar. Para ello se van seleccionando parejas consecutivas de nodos, y se va buscando en la lista de túneles, el túnel que une la pareja de nodos.

Como cada túnel está formado por segmentos conocidos, ya se puede detallar la ruta solución para el robot, concatenando los nodos solución, con los túneles que los conectan, conociendo las características de los segmentos que componen cada túnel.

3.3. Navegación deliberativa

3.3.1. Simulación sensorial

En cuanto a la obtención de datos del entorno, al tratarse de una simulación, no se dispone de sensores reales que puedan proporcionar información real.

Como se expuso anteriormente, este trabajo está basado en parte de un módulo de guiado, navegación y control del proyecto UNEXMIN. Para tal caso, la información de los sensores vendría dado por el módulo de localización y mapeado mediante SLAM. Dicho módulo obtendría toda la información del entorno mediante los sensores y publicaría los datos en un *topic* de ROS, al cual estaría suscrito el nodo de este módulo.

Dicho módulo de mapeado sería desarrollado por otro equipo, por lo que para el desarrollo de este trabajo, de momento, se simulará la entrada de dichos datos de forma automática.

La simulación de la información proporcionada por los sensores se realiza mediante la agregación de un ruido aleatorio. Este ruido aleatorio se genera atendiendo a unos parámetros que el usuario puede configurar y modificar, estableciendo una lectura de sensores

más precisa o con mucho error.

Antes de introducir los parámetros que pueden ser modificados por el usuario, interesa conocer como funciona la aplicación de estos:

Para la generación de datos provenientes de los sensores, una función se encarga de añadir ruido a los datos reales de la posición del robot. Para ello, la función obtiene como parámetros el mapa del entorno y la posición del robot en este.

El grafo que representa el mapa y sus características es clonado, de tal forma que se tiene un grafo que contiene la información del mapa y otro que va a contener la información generada por los sensores simulados. Esto se hace para poder almacenar toda la información obtenida de los sensores.

Cada vez que el robot llega a un nodo, sus sensores se activan y reciben unos datos de lectura. Estos datos se generan variando un porcentaje el valor de los datos reales. El porcentaje de variación es el dato que el usuario puede modificar, y cada parámetro puede ser ajustado individualmente.

Los parámetros que rigen la generación de ruido en las lecturas para cada salida de túnel y sus valores por defecto son:

- **Variación porcentual del valor de altura:** $\pm 10\,\%$.
- **Variación porcentual del valor de anchura:** $\pm 10\,\%$.
- **Variación porcentual del valor de orientación horizontal:** $\pm 10\,\%$.
- **Variación porcentual del valor de orientación vertical:** $\pm 10\,\%$.

Entonces, para cada valor anteriormente mencionado, la lectura del sensor se obtiene mediante el siguiente cálculo.

$$S = R + V \tag{3.2}$$

S : Valor generado por el sensor.
R : Valor real del mapa.
V : Variación aleatoria respecto al rango porcentual.

Donde la variación aleatoria porcentual puede tomar signo positivo o negativo. Así se obtienen los valores simulados de los sensores para cada parámetro presente en el nodo.

Una vez almacenadas todas las variables de los sensores para dicho nodo en el grafo sensorial clonado, se puede proceder a su lectura por parte de otras funciones del algoritmo.

3.3.2. Incertidumbre de posición

Ya que el robot navega sin saber a ciencia cierta donde se encuentra en cada momento, se ha diseñado un índice de incertidumbre. Este índice de incertidumbre trata de expresar el nivel de seguridad que tiene el robot de que su posición estimada es igual a su posición real.

El valor del índice de incertidumbre diseñado se comprende en el rango [0,1], tomando valor 1 cuando el robot está completamente seguro de que su posición estimada es igual a su posición real, y en caso contrario, valor 0.

La seguridad que tiene el robot de su posición decrece a medida que el robot avanza (el índice de incertidumbre va disminuyendo).

En cuanto a qué características se basa el índice de incertidumbre, principalmente este se construirá en los nodos:

Una vez el robot llega al nodo, se obtiene la lectura de los sensores y estos datos de posición real se comparan con los datos esperados de la posición estimada (nodo estimado). Se obtiene entonces un índice de similitud, el cual representa el nivel de similitud que tienen los datos de la posición estimada con los datos de la posición real y toma valores entre [0,1].

Mediante este índice de similitud el algoritmo puede saber cuanto se parece el nodo en el que está al nodo en el cual espera estar. En el caso de que el índice de similitud sea alto (valor mayor de 0,65), significará que el robot se encuentra efectivamente en el nodo esperado; en caso contrario significa que el nodo en el que se encuentra el robot no tiene similitudes y por lo tanto no coincide con el nodo esperado.

El índice de similitud se extrae de la comparación de las características de las salidas del nodo. Se han usado estos datos porque debido a su posición en orientación vertical y horizontal, y sus dimensiones, son los elementos que más distinguen los nodos entre ellos.

El cálculo del índice de similitud se ha calculado de la siguiente forma:

Siendo:

N : Número de salidas del nodo.

$_{\text{i}}$: Salida del nodo.

$^{\text{s}}$: Nodo real. Percepción de los sensores.

$^{\text{m}}$: Nodo estimado. Datos conocidos mediante el mapa.

H : Altura de una salida del nodo.

W : Anchura de una salida del nodo.

α : Orientación horizontal (respecto al norte, en sentido horario) de una salida del nodo.

β : Orientación vertical (respecto al eje horizontal, hacia arriba) de una salida del nodo.

$$dH_i = \frac{H_i^s - H_i^m}{max(H_i^s, H_i^m)} \tag{3.3}$$

$$dW_i = \frac{W_i^s - W_i^m}{max(W_i^s, W_i^m)} \tag{3.4}$$

$$d\alpha_i = \frac{\alpha_i^s - \alpha_i^m}{max(\alpha_i^s, \alpha_i^m)} \tag{3.5}$$

$$d\beta_i = \frac{\beta_i^s - \beta_i^m}{max(\beta_i^s, \beta_i^m)} \tag{3.6}$$

$$P_i = (1 - dH_i)(1 - dW_i)(1 - d\alpha_i)(1 - d\beta_i) \tag{3.7}$$

Nótese que a la hora de comprobar la similitud entre dos nodos, estos pueden no tener el mismo número de salidas. El módulo de comprobación de similitud programado, comprueba antes el número de salidas de cada nodo a comparar (real y estimado). En caso de tener distinto número de salidas, el programa ajusta automáticamente la comparación y añade una penalización al índice de similitud. Cuanta mayor es la diferencia del número de salidas, mayor es la penalización.

Se continúa entonces:

$$E_{s,m} = \frac{\sum_{i=0}^{N} P_i}{N} \tag{3.8}$$

donde:

E : Índice de similitud entre los nodos s (real) y m (estimado).

De esta forma, se obtiene el índice de similitud entre dos nodos dados (E).

El índice de similitud, nunca toma valor unidad. Esto es porque los datos de los sensores son generados a partir de los datos del mapa pero añadiendo ruido, lo que hace que siempre

exista alguna diferencia entre los nodos. Esto hace que sea imposible confirmar para el robot que se encuentra realmente en dicho nodo.

Esta diferencia existente por el ruido genera en el robot cierta incertidumbre, pues no puede confirmar su posición. El índice de incertidumbre viene dado entonces a partir de la acumulación de los errores de similitud, ya que en este trabajo no se ha introducido la implicación de la odometría, dejándose para trabajos futuros.

Cabe destacar, que la obtención del índice de incertidumbre de posición acumulada podría haberse generado mediante diversos teoremas matemáticos y principios estadísticos mucho más complejos. Sin embargo, para el desarrollo de este trabajo se optó por un método más simple, que igualmente se desenvuelve correctamente. La mejora del cálculo de la incertidumbre de posición se propone como una mejora para líneas futuras.

Cada vez que el robot avanza, va acumulando la incertidumbre de posición. Esta incertidumbre aumenta de la siguiente forma:

$$I_T = I_T E_{s,m} \tag{3.9}$$

donde:

I_T : Índice de incertidumbre de posición acumulado.

El valor del índice de incertidumbre de posición acumulado va disminuyendo hasta que su valor cae por debajo de 0.2, valor en el cual el robot entra en modo perdido automáticamente ya que no puede confiar más en su posición.

Resulta muy interesante mencionar que dada la forma en la que se ha planteado el cálculo de la incertidumbre de posición, es fácil añadir nuevos elementos que contribuyan al cálculo de esta.

Esto pone en el punto de mira a la odometría del robot, que claramente desempeñaría un papel muy importante en el cálculo de la incertidumbre. La pose estimada del robot podría usarse y compararse con la posición tridimensional del nodo. Al igual que la odometría, otros elementos fundamentales podrían añadirse al cálculo. Esto abre la puerta también a una infinidad de mejoras y trabajos futuros asociados a la mejora en el ámbito del cálculo de la incertidumbre de posición del robot.

Volviendo a la estructura del índice planteado, existen elementos que van restaurando el índice de incertidumbre de posición del robot a medida que avanza.

- Que se presente un índice de similitud muy por encima de la media (superior a 0.95). En tal caso se considera que el robot confirma su posición.

- Que se encuentre un OOI en un nodo o en un túnel. Si se identifica correctamente el OOI, puede confirmar que el robot está perdido, o reafirmar su posición.

En ambos casos, el índice de incertidumbre de posición acumulado del robot se restaura, volviendo a tomar valor de la unidad.

Identificación de OI

Como se ha expuesto anteriormente, existen OOIs en túneles y nodos. Durante su navegación, el robot puede toparse con ellos. Su detección dependerá de la probabilidad de detección de OOIs definida por el usuario.

En caso de que se detecte el OOI, este será cotejado con una base de datos que contiene todos los OOI. El algoritmo identificará automáticamente la posición en la cual está registrada dicho OOI, cotejándola con la posición estimada actual.

En el caso de que la posición del OOI encontrado y la posición estimada del robot coincidan, significará que el robot se encuentra donde creía, y se procede a restaurar el valor del índice de incertidumbre de posición acumulado.

Por el contrario, si las posiciones son distintas, se confirma que el robot no se encuentra donde estimaba. Como ahora se conoce la posición real del robot gracias a la identificación del OOI, no hace falta entrar en modo perdido. Se procede entonces a una replanificación.

Todos estos pasos se resumen el diagrama de flujo de la Figura 3.6

3.3.3. Modo robot perdido. Relocalización

En caso de entrar en el modo perdido, el robot realiza una lectura de su entorno mediante los sensores.

Tras recibir la información de su entorno, procede a comparar dicha información con todos los nodos del grafo (conocidos). Esto genera una lista de índice de similitudes.

El robot considera que está en el nodo con mayor índice de similitud de todas las comparaciones realizadas. El método utilizado para generar dicho índice permite crear una gran distancia entre el rango de valores que indican una coincidencia en similitud y los valores que informan de que no existen similitudes, es por esto por lo que no son necesarias más acciones.

En el caso de que los nodos sean muy similares entre ellos, sigue habiendo una diferencia más que detectable por el algoritmo, por lo que no genera ningún problema.

El módulo de robot perdido devuelve como parámetro la posición más probable del robot en el grafo según las lecturas de los sensores.

3.3.4. Arquitectura deliberativa

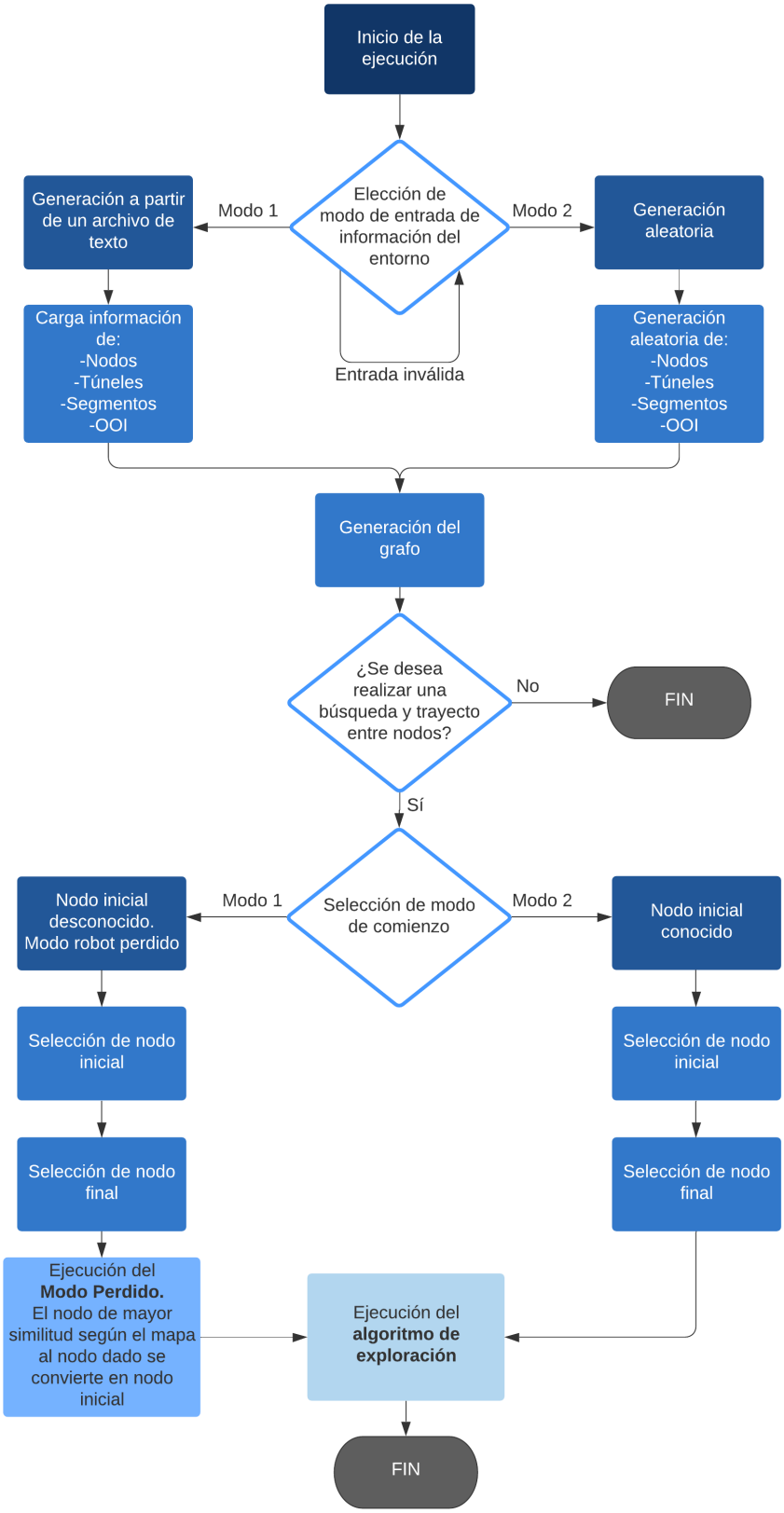


Figura 3.4 Diagrama de flujo de la configuración del programa por parte del usuario.

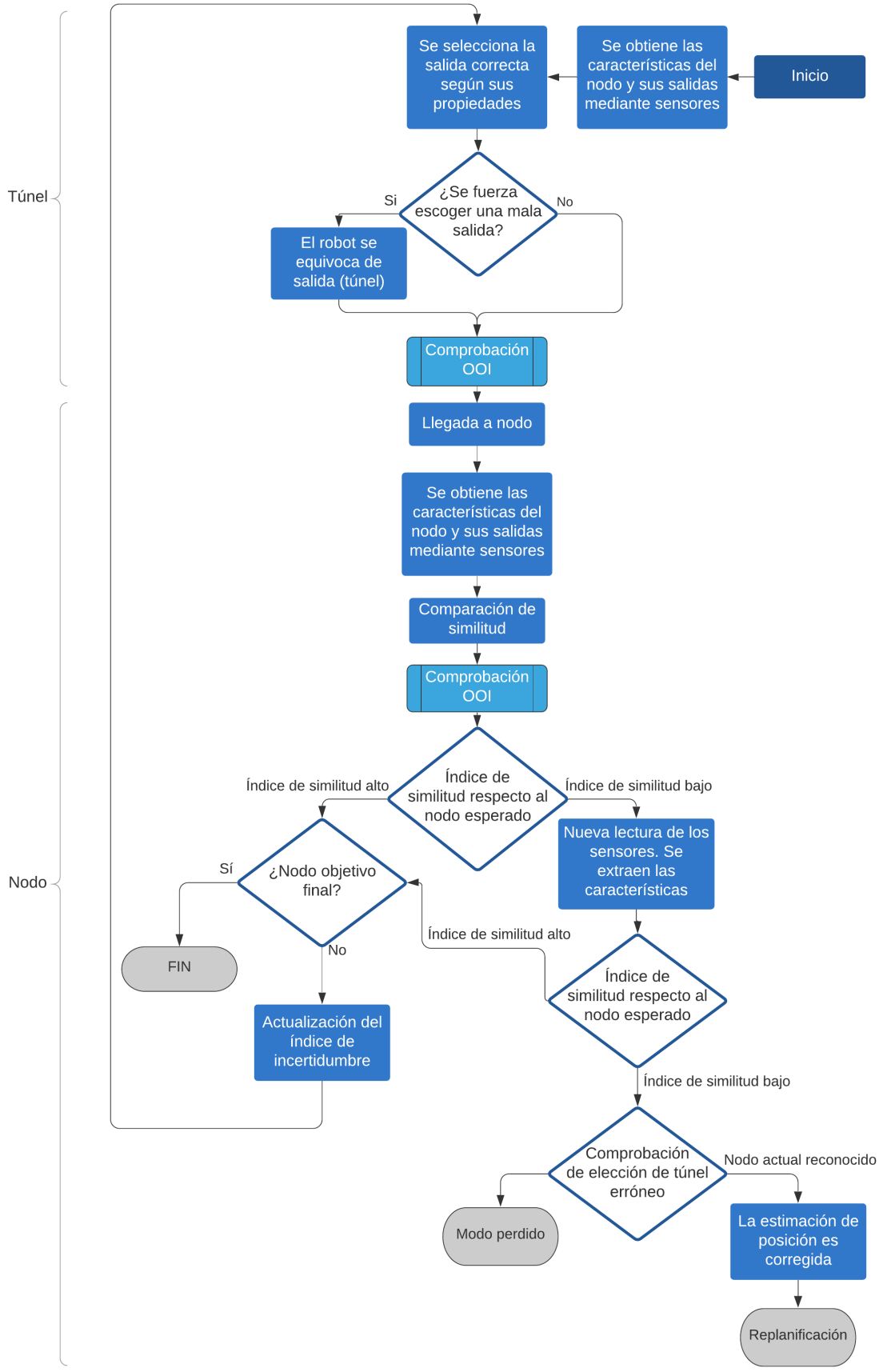


Figura 3.5 Diagrama de flujo de la navegación del robot.

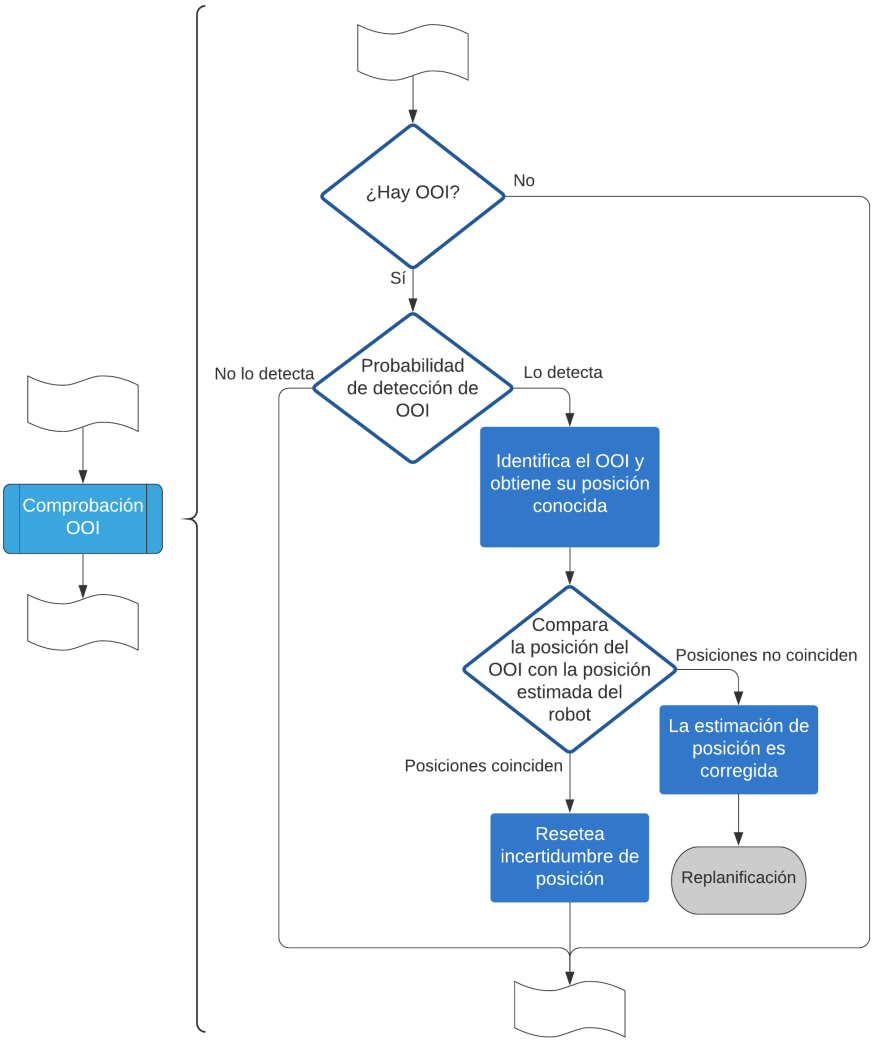


Figura 3.6 Diagrama de flujo de la identificación de OOIs, parte de la navegación en la Figura 3.5.

Al inicio de la ejecución, se han de configurar los parámetros de carga o generación del grafo, así como decidir si se desea realizar una búsqueda entre nodos para la navegación o no. Ya que este proceso requiere la interacción con el usuario, se ha explicado más detalladamente en la sección 3.4.2. La Figura 3.4 resume en un diagrama de flujo la toma de decisiones iniciales, hasta que se ejecuta la búsqueda.

En la Figura 3.5 se muestra el diagrama de flujo simplificado de la toma de decisiones que realiza el algoritmo de navegación. Durante su ejecución, hace uso de varios módulos, como el módulo de programa de modo perdido, el módulo de planificación, el módulo de simulación de sensores o el módulo de comparación para obtener la similitud entre nodos.

Una vez se entra en el módulo de navegación, el módulo de simulación de sensores

extrae los datos de los sensores del nodo actual del robot.

Conocidas las características del nodo, el algoritmo de navegación se encarga de buscar entre todas las salidas del nodo, la salida cuyas características se corresponden con las características de la entrada del túnel planificado por el planificador de la ruta. El robot elige la salida con mayor similitud.

Para añadir dinamismo a la navegación, existe una probabilidad definida por el usuario la cual es evaluada en este punto. Dicha evaluación decide si se engaña al robot mandándolo por un túnel distinto al elegido o si por el contrario el robot sigue el túnel elegido correctamente.

En caso de ser engañado, el robot toma una salida aleatoria dentro del nodo, la cual no puede coincidir con la salida elegida.

Comienza entonces la navegación por el túnel. Durante esta, se muestra por la ventana de comandos el túnel por el que el robot está navegando y sus características. La información túneles son registrados en el grafo en un solo sentido, sin embargo este algoritmo lee el sentido en el que se está dirigiendo el robot y muestra la información en el sentido y orden correcto al usuario para cada transición.

Durante el trayecto por el túnel, el robot puede encontrarse o no un OOI, y en el caso de encontrarlo, puede detectarlo o no. El diagrama de flujo de este procedimiento se puede encontrar en la Figura 3.6. Dicho procedimiento está descrito más en detalle en la sección 3.3.2.

Una vez llega al final del túnel, el robot entra en un nodo. Al hacerlo, el robot realiza una lectura de las características del nodo mediante el módulo de simulación de sensores.

Con esta información se procede a comparar si el nodo al cual se ha llegado coincide con el nodo esperado, esto se hace con el módulo de comparación. Este ofrece un índice de similitud, (procedimiento descrito en la sección 3.3.2).

- Si el índice de similitud es alto, el robot considera que ha llegado correctamente al nodo esperado. Comprueba si se trata del nodo final planificado, y si es así, finaliza la ejecución. En caso de que no sea el nodo final, actualiza el valor del índice de incertidumbre de posición acumulado.

- Si el índice de similitud es bajo, el robot realiza una nueva lectura de los sensores mediante el módulo de simulación de sensores. Esto se realiza para cerciorarse de que la presencia de posible ruido pueda confundir al robot. Se realiza nuevamente

una comparación de los datos de los sensores con los datos del nodo esperado y se evalúa el índice de similitud. Si el índice de similitud se considera alto tras esta segunda lectura, sigue la secuencia descrita en el punto anterior. En el caso de que el índice de similitud siga siendo bajo, se comprueba si se ha tomado una salida errónea.

- Puede darse el caso de que el robot haya tomado una salida errónea a la planificada, aún estando en un nodo correcto. Esto se comprueba fácilmente ya que el mapa es conocido. Se evalúan los nodos conectados con el anterior nodo, y se realizan comparaciones de similitudes con estos. Si algún índice de similitud resulta alto, significa que se ha comprobado que el robot está en ese nodo adyacente al nodo anterior, al cual ha llegado por haberse equivocado de túnel. Esta comprobación corrige la posición estimada del robot.

  Ya que el robot se ha salido de la ruta planificada, se ordena una replanificación para llegar al nodo final de la forma más eficiente posible.

- En el caso de que el robot no estuviese donde esperaba (incertidumbre de posición acumulada muy mala) y no se haya cumplido la comprobación del punto anterior, el robot se considera perdido y entra en modo perdido.

En el caso de que el robot llegase al nodo esperado correctamente y se actualice su índice de incertidumbre de posición acumulada, se volverá al paso inicial, que es elegir la salida que se corresponde con el túnel planificado, cerrando así el ciclo del diagrama de flujo, que continuará hasta llegar al nodo final.

3.4. Monitorización

Teniendo en cuenta que este trabajo trata sobre la operación de un robot autónomo en entornos donde la comunicación no es posible debido a las características del entorno, una monitorización en tiempo real sería imposible. Las comunicaciones robot-usuario se suprimen en el fondo de una mina inundada. Es por eso por lo que la monitorización de la navegación del robot se realizaría *a posteriori*.

Sin embargo, al tratarse de una simulación, se ha diseñado un sistema de monitorización que permite observar las decisiones que toma el robot, así como su posición en tiempo real mediante la representación del mapa topográfico en 3D.

Para la representación 3D del mapa topológico se ha utilizado la herramienta de visualización Rviz para ROS (Robot Operating System). La ejecución de ROS se ha imple-

mentado en el programa QtCreator para facilitar su ejecución.

3.4.1. Implementación de ROS y QtCreator.

QtCreator.

QtCreator es un entorno de desarrollo multi-plataforma el cual permite la programación en C++ entre otros lenguajes. Este programa tiene incorporado un depurador de código visual, lo que ha facilitado enormemente la detección y corrección de errores a lo largo de la programación del proyecto.

Mediante QtCreator se compila y ejecuta todo el código. Al hacerlo, muestra una ventana de comandos que recibe e imprime todos los datos enviados por el algoritmo.

Para este proyecto se ha usado la versión gratuita de QtCreator 5.9.5 y QMake 3.1.

ROS.

ROS (Robot Operating System) o ros es un middleware de robótica que permite el desarrollo de software para robots. Este permite entre otras cosas la administración de paquetes y el intercambio de mensajes entre procesos. Se trata de un software libre y permite la interacción de múltiples lenguajes de programación como C++, Java y Python.

En este proyecto se ha usado la versión *Melodic*, ejecutada en Linux Mint 19.3, aunque existe una versión posterior.

ROS hace principalmente uso de nodos, topics y mensajes:

- Nodos. Son ejecutables que permiten realizar tareas, así como intercambiar mensajes.
- Topics. Los nodos se suscriben o publican sus mensajes en topics.
- Mensajes. Es el tipo de dato usado por ROS y es generado por los nodos.

Con fin de poder ejecutar nodos, previamente se ha de inicializar una colección de nodos y programas que componen los pre-requisitos para que los programas de ROS funcionen. Esto permite que los nodos que se ejecuten puedan comunicarse entre sí. Esto se consigue ejecutando el siguiente comando:

Terminal:

```
user:~$ roscore
```

Para la representación 3D del mapa topológico se ha hecho uso del paquete *rviz_visual_tools*, desarrollado por *PickNik Robotics*. Este paquete es un API (Application Programming Interface) para C++ que permite mostrar formas y mallas en Rviz mediante la publicación de marcadores de distintas características. Para este trabajo únicamente se han usado las formas cilíndricas y esféricas, variando en color y tamaño y de etiquetas de texto para nombrar marcadores.

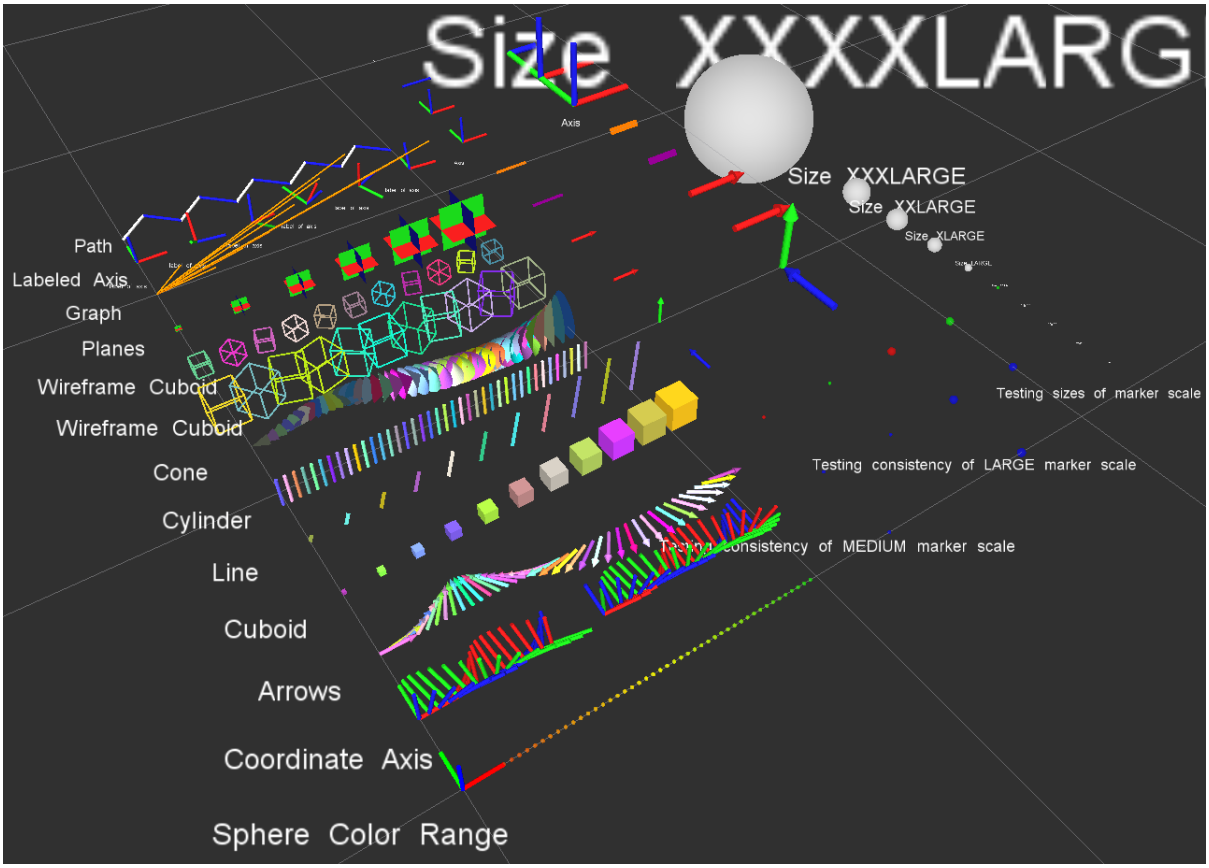


Figura 3.7 Marcadores de distintas formas creados mediante el paquete *rviz_visual_tools* en Rviz mediante ROS.

Mediante este paquete se crean dos nodos suscritos a un mismo *topic*:

- Nodo Rviz. Este nodo es el encargado de recibir la información de los marcadores y representarlos en 3D. Se lanza mediante el comando:

  Terminal:

  ```
  user:~$ roslaunch rviz_visual_tools demo_rviz.launch
  ```

Mediante el cual aparece la interfaz de Rviz:

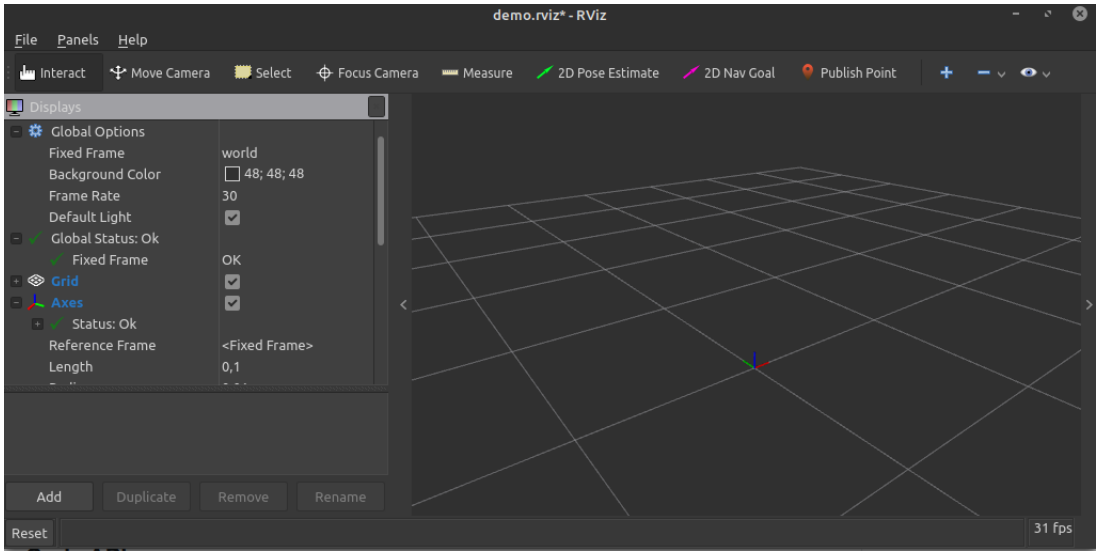


Figura 3.8 Interfaz de Rviz.

- Nodo de ejecución. Este ejecuta todo el programa y se encarga de publicar todos los marcadores para que sean representados, además de imprimir en la ventana de comandos la información necesaria.

Terminal:

```
user:~$ roslaunch rviz_visual_tools demo.launch
```

Aunque pueden funcionar por separado, se planteó el reto de implementar ROS en QtCreator de tal forma que simplemente con un clic se pudiesen ejecutar todos los nodos y funcionase el programa.

Esta fusión viene facilitada por la herramienta *ROS QtCreator Plug-in*, desarrollada por *ROS-Industrial*. Esta herramienta implementa la posibilidad de crear espacios de trabajo de ROS en la plataforma QtCreator. Dentro de dichos espacios de trabajo, permite crear paquetes que contienen nodos, o cargar paquetes ya existentes.

Esta herramienta además permite planificar secuencias de ejecución de nodos de ROS. Por vía de esta característica, se creó la secuencia que permite primero crear el nodo que lanza la interfaz Rviz, seguido del nodo que ejecuta el programa que contiene el algoritmo. Gracias a esto, no es necesario introducir las llamadas de lanzamiento de los nodos en la ventana de comando, realizándose con un simple clic en el icono de ejecutar el programa.

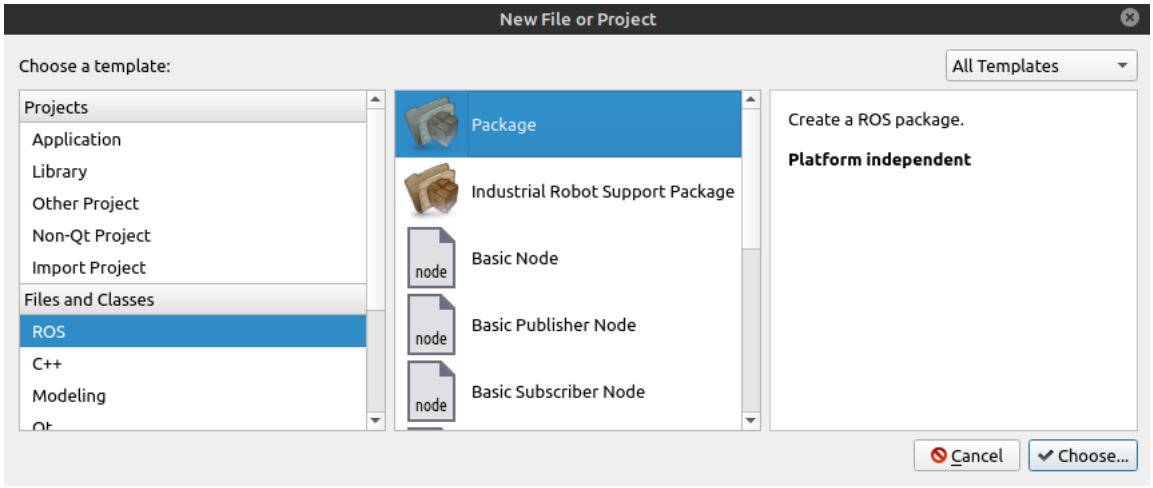


Figura 3.9 Creación de paquetes ROS en la plataforma QtCreator mediante el plug-in diseñado por *ROS-Industrial*.

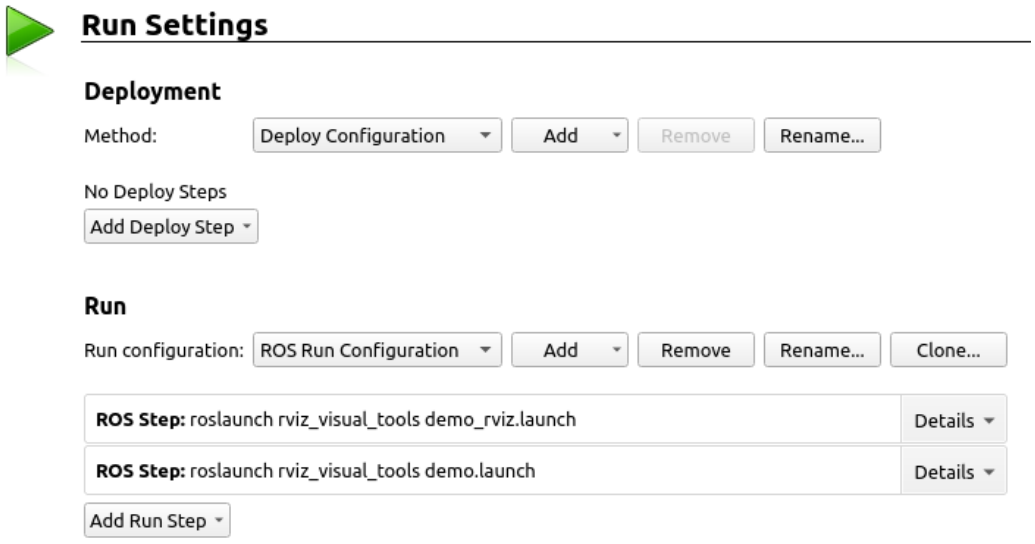


Figura 3.10 Configuración de la secuencia de lanzamiento de nodos ROS implementada en QtCreator.

3.4.2. Construcción del grafo

.

Una vez ejecutada la secuencia de nodos de ROS, el usuario ha de configurar los parámetros de lanzamiento, que son requeridos por el nodo que está ejecutando el algoritmo. Para ello, se hace uso de la ventana de comandos, debido a su simplicidad y rapidez. Una interfaz podría haberse diseñado para la introducción de parámetros, pero dado a la

simplicidad de configuración se consideró conservar el formato de introducción en ventana de comandos.

Inicialmente al ejecutar el programa, el usuario se encuentra lo siguiente en la ventana de comandos:

Terminal:

```
Choose a graph input mode:
1. Load graph from file (.txt).
2. Random graph.
_____

Mode:
```

En ella, se muestra al usuario dos opciones:

- Cargar el mapa desde un archivo de texto. [Opción 1].
- Generar un mapa aleatorio. [Opción 2].

El usuario ha de introducir un número que corresponde con el modo a elegir. En caso de introducir un carácter que no se contemple entre las opciones, se mostrará un mensaje de "Invalid option", que alertará al usuario de que ha introducido una opción inválida y esperará de nuevo la introducción de un número que se corresponda con una opción válida.

1. En el caso de elegir la opción 1 (Cargar el mapa desde un archivo de texto), el programa accederá al archivo de texto cargado en una ruta predefinida, leerá y cargará su contenido según el procedimiento explicado en la sección 3.1.4. En caso de no encontrar dicho archivo de texto, el programa mostrará un error.

   Una vez cargado el mapa, los datos del grafo generado se imprimirán en la ventana de comandos para hacer posible su comprobación.

2. En el caso de elegir la opción 2 (Generar un mapa aleatorio), se le pedirá al usuario que introduzca un número de nodos para generar el grafo.

   Terminal:

   ```
   Generating a random graph
   Introduce the number of nodes of the desired graph:
   ```

   Una vez introducido el número de nodos, se generará el grafo atendiendo al procedimiento explicado en la sección 3.1.5, mostrando por pantalla los datos del entorno generado.

La información del entorno generado se muestra en la ventana de comandos de la siguiente forma:

Nodos:

Terminal:

```
------Nodes input------

Node:0   Name:N0        Number of exits:1
Height:1.631   width:1.951   Depth:3.956
Coordinates [x,y,z]: [-8.292,-4.653,-0.386]

                Exit to node 1
                Height:1.355 Width:1.455   H. ori.245.78   V. ori.:320.70


Node:1   Name:N1        Number of exits:3
Height:2.170   width:1.818   Depth:1.965
Coordinates [x,y,z]: [-12.885,1.711,-2.313]

                Exit to node 0
                Height:1.355   Width:1.455   H. ori.245.78   V. ori.:320.70

                Exit to node 6
                Height:1.090   Width:0.984   H. ori.199.80   V. ori.:137.82

                Exit to node 5
                Height:1.130   Width:0.825   H. ori.253.48   V. ori.:58.83
...
...
...
```

Túneles y sus segmentos:

Terminal:

```
------Tunnels input------

Tunnel:0        Ending nodes: 0-1       Length:1        Number of segments:1

                -Segment 0-1    Length:1.153   Height:1.201   Width:1.134   Slope:9.402


Tunnel:1        Ending nodes: 1-6       Length:19.3336  Number of segments:4

                -Segment 1-1    Length:6.231   Height:0.788   Width:1.0237 Slope:13.105
                -Segment 1-2    Length:6.842   Height:1.066   Width:0.904   Slope:14.649
                -Segment 1-3    Length:2.624   Height:1.327   Width:0.942   Slope:0.521
                -Segment 1-4    Length:1.532   Height:1.332   Width:0.635   Slope:7.594

...
...
...
```

Conexiones del grafo:

Terminal:

```
------Node adjacency list------

Node --> (LinkedNode, Distance)

0 --> (1,1)
1 --> (0,1) (6,19) (5,20)
2 --> (3,6) (6,17) (4,10)
3 --> (2,6) (7,15) (6,20)
4 --> (2,10) (6,7) (7,2) (8,6)
5 --> (1,20) (9,10) (8,7)
6 --> (1,19) (2,17) (3,20) (4,7)
7 --> (3,15) (4,2) (8,4) (9,4)
8 --> (5,7) (7,4) (4,6) (9,6)
9 --> (5,10) (8,6) (7,4)
```

3.4.3. Configuración inicial y monitorización mediante la ventana de comandos

Una vez cargado el mapa en forma de grafo, el programa pregunta al usuario si desea realizar una búsqueda de ruta entre dos nodos. En caso afirmativo, se ejecuta el algoritmo de búsqueda en grafos.

Terminal:

```
Do you want to perform a path search between two nodes? (Y/N).
```

Elección de nodo origen y nodo destino

Para su funcionamiento, este requiere la entrada de dos parámetros, el nodo de origen y el nodo de destino.

Terminal:

```
Select start mode.
1. Lost start mode
2. Known start node
_____________________
Mode:
```

El programa plantea dos modos:

1. Modo perdido. En este modo, el robot comienza desde una posición que el usuario elige, pero de la cual el robot es desconocedor.

   Se ejecuta entonces la función de relocalización para tratar de encontrar la posición más probable en la que se encuentra el robot.

Una vez el programa de relocalización ofrece la posición más probable, se toma como nodo de origen.

El nodo de destino es también requerido por el programa para que sea introducido por el usuario.

Terminal:

```
Mode:1
Set start lost node:
0
Set goal node:
18
----- Robot lost -----

Looking for the most probable current node according to similarities.
Node (0) is the most probable current node.
```

2. Comienzo desde un nodo conocido. El nodo de origen es seleccionado y el robot es conocedor de su posición inicial.

   El nodo de destino es también requerido por el programa para que sea introducido por el usuario.

Terminal:

```
Mode:2
Set start node:
0
Set goal node:
18
```

Navegación

Una vez establecidos el nodo de origen y destino, se ejecuta el algoritmo de búsqueda en grafos, que planifica una ruta a seguir, mostrando al usuario la secuencia de nodos planificada.

Terminal:

```
-------Path search required-------

Finding path from node 0 to node 18.
Solution node sequence: 0 1 5 8 10 14 18
```

A partir de aquí, se inicia la navegación. Durante esta, el programa irá informando al usuario sobre la posición actual del robot, así como los túneles que atraviesa en cada

momento y sus características.

Un ejemplo de cómo muestra el programa al usuario que el robot ha detectado un OOI, reiniciando el índice de incertidumbre, se puede observar en la siguiente imagen:

Terminal:

```
Node 1 arrived.
Nodes to compare:  real (1) and expected (1)
Node check: P = 0.844242
OOI ID_0 identifyed.
Checking OOI with the database.
Position confirmed by the OOI.
Position uncertainty = 1
Tunnel transition.
```

Para el caso de que el algoritmo fuerce a equivocarse de salida al robot, la información que imprime en la ventana de comandos tiene la siguiente forma. Se informa de que se ha elegido un mal túnel y el procedimiento de toma de decisiones que el programa toma:

Terminal:

```
Tunnel transition.
***BAD TUNNEL CHOSEN***.

Tunnel:23
*****Impresion de los datos del tunel recortada*****

Node 13 arrived.
Nodes to compare:  real (13) and expected (14)
Node check: P = 0.18305
Exits number different.
Re-scaning.
Nodes to compare:  real (13) and expected (14)
Number of exits different by 1. Penalization.
Node 13 missed. Checking for mistakes in previous tunnel choice.
Checking the map for detecting tunnel mistakes.
Mistaken tunnel choosed confirmed by map simmilarities.
------Path search required------
```

Una vez que el robot llega a su destino, se notifica del final del programa, terminando la emisión de información a los nodos de representación en Rviz.

Terminal:

```
***** End node reached correctly *****
Ending program...

[ INFO] [1601073335.404734221]: Shutting down.
[rviz_visual_tools_demo-1] process has finished cleanly
all processes on machine have died, roslaunch will exit
shutting down processing monitor...
... shutting down processing monitor complete
done
```

Monitorización en Rviz

El grafo es generado en 3D en la herramienta de ROS, Rviz. En él, se observan los siguientes elementos, que son representados de la siguiente forma:

- Nodos: Esferas azules.
- Túneles: Aristas negras.
- Nodo de origen: Esfera verde.
- Nodo de destino: Esfera roja.
- Ruta planificada: Aristas blancas.
- Posición del robot : Esfera o arista naranja.

En las tres próximas figuras se muestra visualmente mediante la herramienta Rviz cómo un robot se sale de la ruta planificada, perdiéndose en un nodo al que no esperaba llegar. Tras darse cuenta, se muestra un mensaje de replanificación en rosa, el robot realiza la planificación y se muestra la nueva ruta planificada.

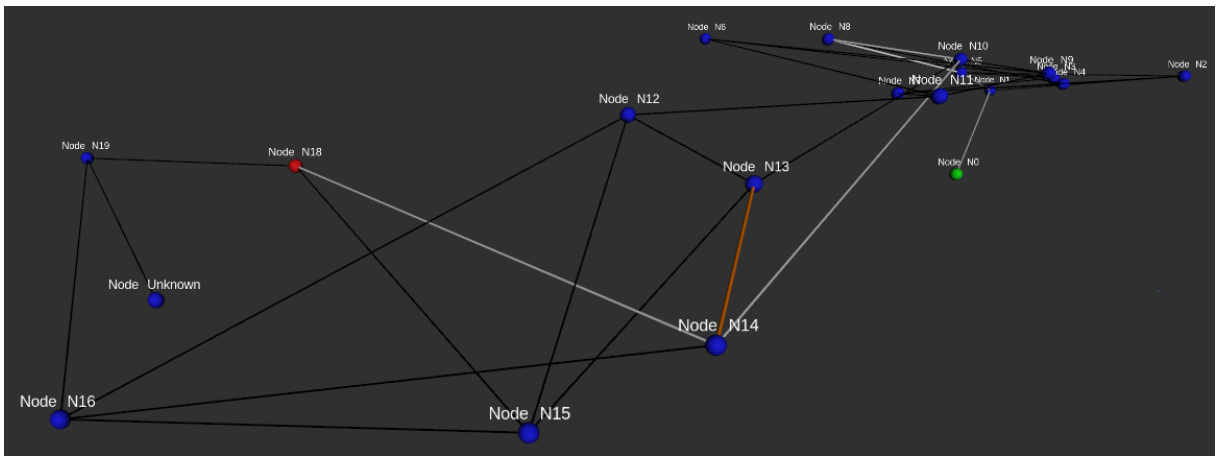


Figura 3.11 Visualización en la herramienta Rviz del grafo y el robot saliéndose de la ruta planificada.

Ya que se han expuesto todas las partes que componen este trabajo, se muestra en la Figura 3.14 un esquema UML (lenguaje unificado de modelado) que modela el sistema programado en C++, mostrando como interacciona cada una de las partes. Esto permitirá al lector tener una vista más esquemática del programa.

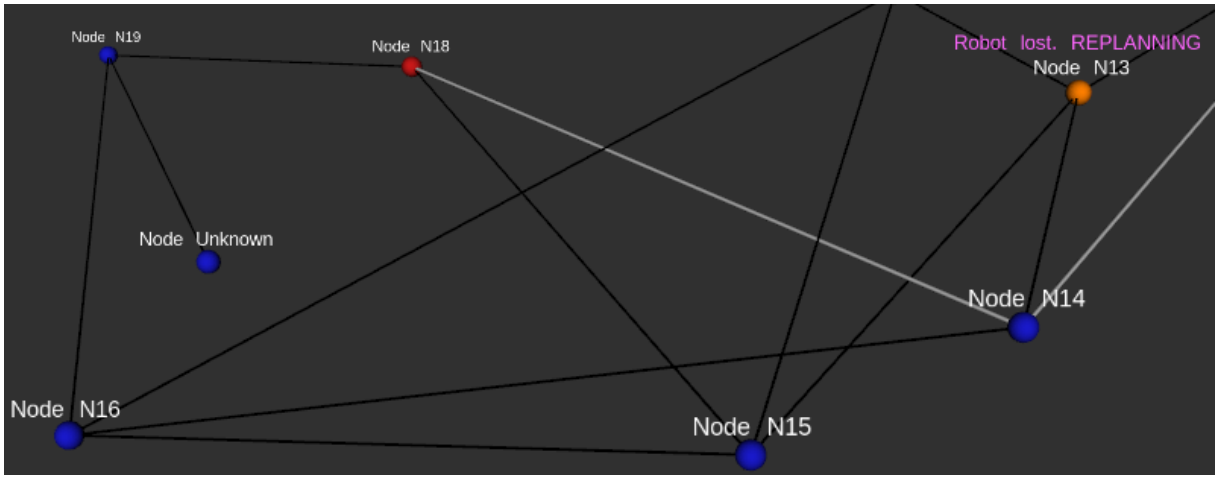


Figura 3.12 Visualización en la herramienta Rviz del grafo y el robot dándose cuenta de que se ha perdido y realizando una replanificación.

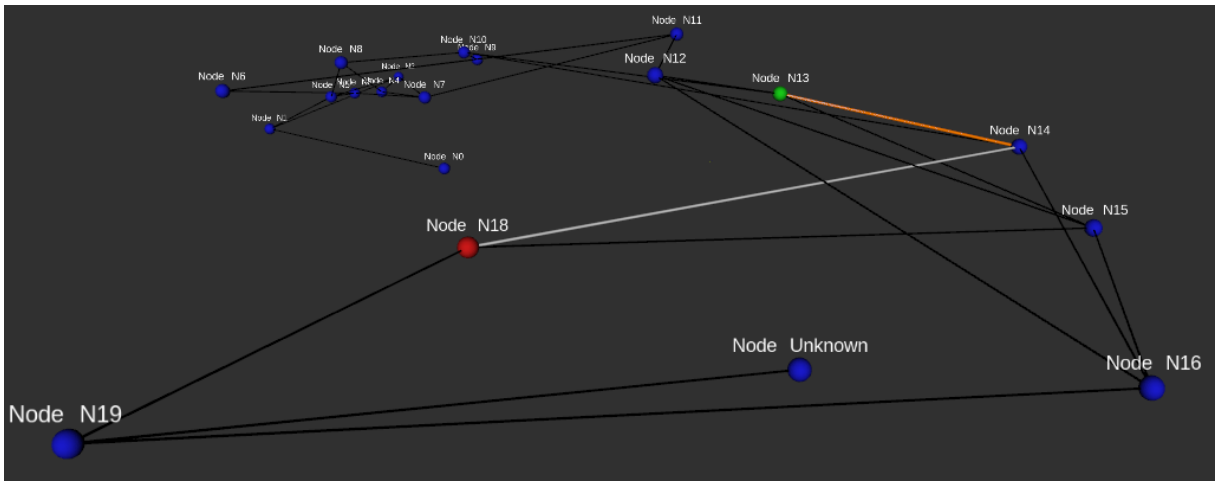


Figura 3.13 Visualización en la herramienta Rviz del grafo y el robot navegando por la nueva ruta planificada. Vista rotada.

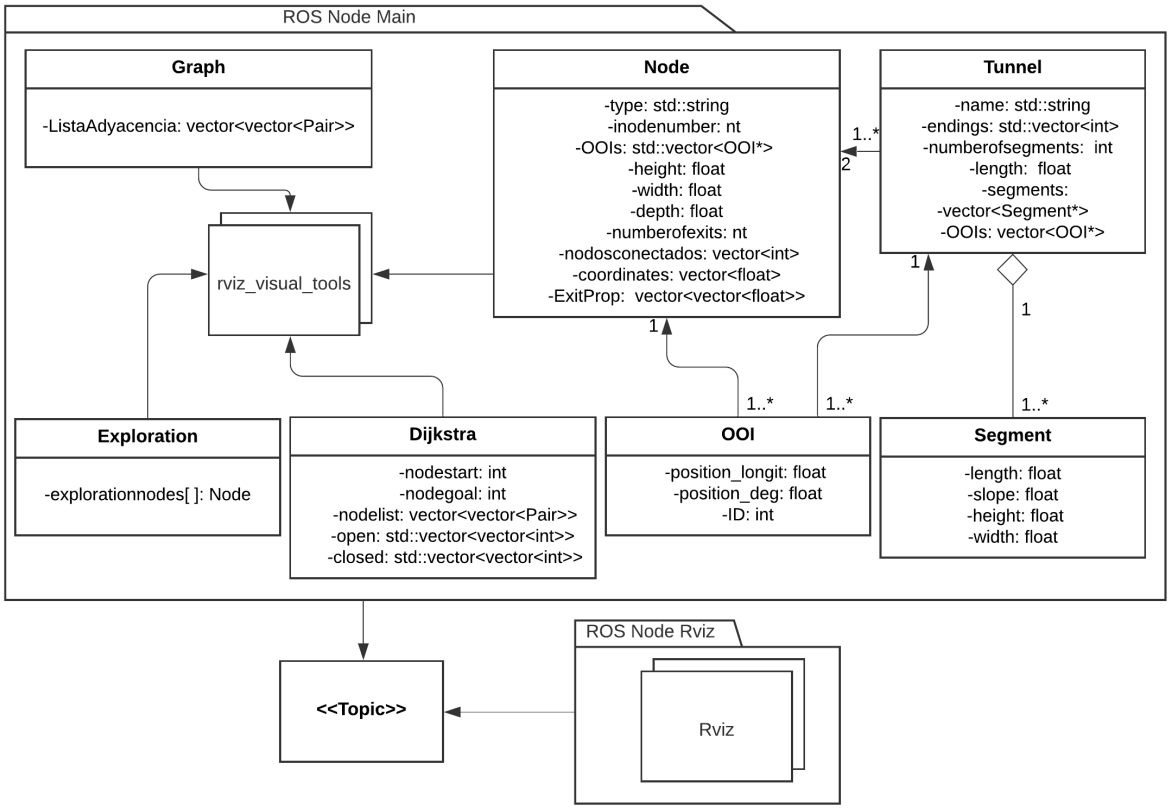


Figura 3.14 Diagrama en UML del programa desarrollado en este trabajo.

Capítulo 4

Resultados y discusión

La creación de grafos, tanto desde un archivo de texto como generados aleatoriamente, han sido puesto a prueba mediante un gran número de ejecuciones, y en todas el sistema se comportó correctamente.

En cuanto al módulo de planificación, el algoritmo Dijkstra ha sido puesto a prueba, comprobando la ruta proporcionada por este respecto a rutas alternativas. En el 100 % de las ocasiones dicho módulo ha ofrecido la ruta más rápida y directa, lo cual cumple el objetivo de dicho módulo.

El algoritmo que realiza la búsqueda del modo perdido ha sido puesto a prueba extensamente , variando los parámetros de ejecución para ver su comportamiento ante distintas situaciones.

Se ha creado un algoritmo que pone a prueba dicha parte del programa para entornos con distintos números de nodos, poniendo el robot en modo perdido en un nodo aleatorio del grafo y comprobando si se encuentra correctamente.

Ante las condiciones de generación de nodo siguientes, los resultados obtenidos se resumen en la Tabla 4.1:

Características de la generación de nodos distintos:

- **Altura nodo**. Mínimo: 1.5 metro — Máximo: 4 metros.
- **Anchura nodo**. Mínimo: 1.5 metro — Máximo: 2 metros.
- **Profundidad nodo**. Mínimo: 1 metro — Máximo: 4 metros.
- **Altura salida de nodo**. Mínimo: 0.5 metro — Máximo: 1.5 metros.

- **Anchura salida de nodo**. Mínimo: 0.5 metro — Máximo: 1.5 metros.
- **Orientación vertical salida de nodo**. Mínimo: 0º— Máximo: 360º.
- **Orientación horizontal salida de nodo**. Mínimo: 0º— Máximo: 360º.
- **Número de salidas en cada nodo**. Mínimo: 2 salidas — Máximo: 4 salidas.

Características de la generación de nodos similares:

- **Altura nodo**. Mínimo: 1.5 metro — Máximo: 2 metros.
- **Anchura nodo**. Mínimo: 1.5 metro — Máximo: 2 metros.
- **Profundidad nodo**. Mínimo: 1.5 metro — Máximo: 2 metros.
- **Altura salida de nodo**. Mínimo: 0.5 metro — Máximo: 1 metros.
- **Anchura salida de nodo**. Mínimo: 0.5 metro — Máximo: 1 metros.
- **Orientación vertical salida de nodo**. Mínimo: 0º— Máximo: 180º.
- **Orientación horizontal salida de nodo**. Mínimo: 0º— Máximo: 180º.
- **Número de salidas en cada nodo**. Mínimo: 2 salidas — Máximo: 3 salidas.

Tabla 4.1 Tabla de porcentaje total de aciertos del algoritmo de modo perdido para distintos parámetros.

	Número de nodos			
	20	50	100	200
Acierto en nodos de características distintas	100 %	100 %	100 %	100 %
Acierto en nodos de características similares	100 %	100 %	100 %	100 %

Los resultados obtenidos arrojan una efectividad del 100 % por parte del módulo de modo perdido ante distintas características de nodos. Al hacer los nodos más similares, la diferencia entre la comparación entre distintos nodos se reduce, sin embargo sigue siendo lo suficientemente significativa como para reconocer el nodo en el que se encuentra el robot.

Respecto al módulo de exploración, se diseño un algoritmo para poder ejecutar dicho modulo numerosas veces y así poder evaluar su funcionamiento. Se mandó al modulo realizar la exploración entre dos nodos aleatorios de grafos generados aleatoriamente, obteniendo un 100 % de éxito todas las veces (1000 ejecuciones). Esto ocurre porque el módulo de exploración está diseñado para no detenerse hasta que el robot alcance el objetivo, para ello se ayuda del módulo de replanificación y de modo perdido.

Tabla 4.2 Valor medio del índice de incertidumbre al final de la exploración en función del tamaño del grafo.

	Número de nodos			
	20	50	100	150
Índice de incertidumbre final medio	0.804	0.765	0.788	0.701

La tabla 4.2 muestra cómo el índice de incertidumbre medio a la hora de llegar el robot a la posición de destino no muestra ninguna conclusión relevante, variando en su media en torno a una décima sobre el 70 %. Analizando estos valores, se pueden considerar satisfactorios, aunque sin duda se podrían añadir más métodos para mejorar dicha cifra, los cuales se verán en el apartado de trabajos futuros.

A modo de ver el impacto de forzar equivocaciones a la hora de tomar una salida del nodo sobre el tiempo de exploración, se han realizado pruebas ante distintas características. Los resultados han mostrado que ante equivocaciones, el robot no explora muchos más nodos de los planeados inicialmente. Esto es un dato de interés pues si el robot explorase más de lo planeado inicialmente podría ser un problema ya que el robot puede quedarse sin batería.

Tras el análisis de los datos y las rutas de replanificación, se ha sacado la conclusión de que al ser la ruta inicial la más corta y óptima, en caso de equivocarse de túnel, el robot vuelve al nodo de la ruta inicial porque sigue siendo la más óptima. Esto garantiza que el robot no malgastará batería en navegaciones extras por equivocaciones.

Para evaluar el algoritmo de comparación, se ha ejecutado 1000 veces para distintos valores de números de nodos que compone el grafo, dando el mismo resultado. Puesto que esa evaluación no era concluyente, se ha procedido a enfocar su análisis en la variación de los datos mediante el ruido generado, algo que podría confundir al algoritmo de comparación y que resulta de mayor interés.

Para ello, se tiene como referencia la diferencia entre el índice de similitud máximo y la media de índices de similitud de todos los nodos. Esto es porque el módulo de comparación funciona eligiendo como nodo correcto el de mayor índice de similitud, por lo que

la diferencia del valor máximo con el resto de valores resulta fundamental.

Tabla 4.3 Valor medio del porcentaje de diferencia entre el índice de similitud del nodo correcto y el índice de similitud del resto de nodos del grafo, en función de la variación del ruido en la lectura de sensores.

	Porcentaje de ruido aplicado a la lectura de los sensores			
	±10 %	±25 %	±40 %	±50 %
Porcentaje de diferencia entre índices de similitud máximo y medio	139,81 %	147,69 %	145.52 %	142.77 %

Como se muestra en la tabla 4.3, el módulo de comparación no se ve alterado por el nivel de ruido en los sensores. Tras evaluar los datos resultó revelador ver como evolucionaba el índice de similitud máximo según se iba incrementando el ruido en las lecturas. Este iba disminuyendo drásticamente, pero los índices de similitud de los nodos que no se correspondían con el nodo actual del robot disminuyen en la misma medida, conservándose la relación del porcentaje de diferencia.

Para poner un ejemplo numérico. Con una variación del ±10 % mediante ruido en las lecturas de los sensores, el índice de similitud del nodo correcto resulta en torno al valor 0.85, y el de los nodos no correctos en torno al 0.10.

En el caso del ruido al ±50 %, el valor del índice de similitud del nodo correcto es de 0.33, mientras que de los nodos erróneos es de 0.06.

Se concluye entonces que pese al incrementarse el ruido en las lecturas de sensores, el módulo de comparación entre nodos sigue funcionando correctamente.

En cuanto a la navegación se refiere, es interesante el efecto que tiene la variación del ruido en las lecturas de los sensores con el desarrollo de esta. Tras realizar numerosas pruebas, se ha concluido que la variación del nivel de ruido no afecta al módulo de modo robot perdido, ya que compara todos los porcentajes de similitud. Sin embargo, a la hora de navegar, el robot se basa en el índice de incertidumbre y en el índice de similitud al llegar a un nodo, el cual es comparado con las propiedades del nodo esperado en la base de datos. La variación por el ruido hace que el robot no sea capaz de reconocer el nodo aunque sea correcto, dado a que el aumento de ruido reduce el valor del índice de similitud como se ha expuesto anteriormente.

Como solución a este problema, se presenta regular el valor mediante el cual el algoritmo considera el nodo correcto según un índice de similitud mínimo. Cabría esperar un ajuste previo a la navegación según el nivel de ruido de los sensores, que determinaría

dicho valor mínimo admisible.

De esta forma, el algoritmo estaría adaptado a esas grandes variaciones de las lecturas de los sensores producidas por el ruido, y se procedería a navegar correctamente.

Se ha realizado la prueba realizando dichos ajustes y el robot ha conseguido navegar correctamente.

En cuanto a la visualización en Rviz por parte de la plataforma ROS, se ha obtenido el rendimiento esperado, presentando en muy raras ocasiones pequeños retrasos en la representación, que se solventan de forma inmediata, pues se publican representaciones de marcadores cada muy poco tiempo.

El tiempo de ejecución de la simulación completa depende en gran medida de la ruta seleccionada por el robot y su exploración, ya que la generación del grafo y la planificación es casi inmediata. La exploración contiene pausas para hacer más sencillo el seguimiento por parte del usuario, que son las que condicionan el tiempo total de ejecución.

Capítulo 5

Conclusión

5.1. Conclusiones

Este Trabajo de Fin de Máster se planteó como todo un reto a la hora de perfeccionar conceptos aprendidos durante la impartición de este Máster, como la teoría de grafos, algoritmos de búsqueda en grafos, navegación en robots, simulación. También se han adquirido nuevos conceptos, como a la hora de trabajar en un proyecto grande en C++, pensando siempre en la programación modular, pues ofrece claridad y ventajas para añadir o eliminar funcionalidades. El uso de la plataforma QtCreator y el uso de todo su potencial, así como el estudio de ROS y su aplicación en este proyecto. La fusión de QtCreator y ROS y el uso de paquetes adjuntos. En definitiva, este trabajo no solo ha mejorado los conocimientos ya aprendidos, sino que ha añadido muchos otros más y de gran valor.

Se comenzó evaluando el estado del arte, las técnicas más usadas en cuanto a la creación de mapas topológicos, su navegación y las búsquedas en grafo. Esto permitió seleccionar la mejor metodología para desarrollar este proyecto.

Una vez conocida la metodología a seguir, se comenzó desarrollando una estructura de grafo completamente desde cero. Esta estructura se fue mejorando mediante la adicción de nuevos elementos y mediante la selección del modo de creación, permitiendo utilizar datos de minas reales mediante archivos de texto, o sacar partido de la simulación generando grafos aleatorios configurables por el usuario.

Se ha implementado satisfactoriamente un algoritmo de búsqueda en grafos Dijkstra, el cual se ha usado para realizar la planificación de ruta a seguir por el robot dentro del grafo. Dicho módulo ha demostrado se más que eficiente y lograr siempre proporcionar la

ruta más rápida, corta y óptima.

Se han diseñado distintos módulos que permiten el correcto desarrollo de la navegación del robot por el grafo, como el módulo de comparación entre nodos, módulo de simulación sensorial, módulo de modo perdido y módulo de monitorización. Mediante estos, se ha construido una lógica deliberativa, que permite al robot tomar decisiones ante distintas situaciones y características percibidas por los sensores. Este consigue llegar a su objetivo correctamente a pesar de los errores forzados en la navegación, lo cuales son incluidos para evaluar su capacidad de resolución de problemas.

La navegación del robot ofrecida en este trabajo se ha sometido a pruebas y se ha demostrado su robustez, así como eficiencia. El principal objetivo de este trabajo se ve cumplido con el algoritmo desarrollado, así como los objetivos secundarios.

Un sistema de monitorización de la posición 3D ha sido desarrollado en ROS para el seguimiento por parte del usuario, e implementado en la plataforma QtCreator satisfactoriamente, permitiendo la ejecución de todas las partes del trabajo con un solo clic. Se ha hecho uso de paquetes y de la herramienta de visualización Rviz de ROS, integrando todo en un solo programa.

La visualización del grafo 3D ofrece al usuario una mejor visión del entorno y de la posición del robot. Además, se puede observar los diferentes caminos que puede seguir el robot, y la ruta que el robot ha planeado.

Toda la información de la toma de decisiones del robot es secuencialmente imprimida en la ventana de comandos a medida que el robot avanza. Además, se imprimen las características del entorno.

La inspiración del proyecto UNEXMIN y la motivación de intentar recrear un módulo de su complejo desarrollo ha resultado excitante y ha planteado numerosos retos que han sido solventados mediante aprendizaje y trabajo.

Se concluye entonces con un trabajo satisfactorio, que cumple todos los objetivos propuestos inicialmente, y que permite al usuario interactuar con la simulación y monitorizar el desarrollo de la navegación. Además ha fomentado adquirir nuevos conocimientos imprescindibles en el ámbito de la robótica como es el uso de ROS y su implementación en otras plataformas.

5.2. Mejoras y líneas futuras

Tras la finalización de este trabajo, se plantean posibles mejoras y líneas futuras, que permitirían la continuación del desarrollo del trabajo.

Dado a que este trabajo se inspira en el desarrollo del proyecto UNEXMIN, se podría continuar con el desarrollo de otro módulo directamente relacionado con el de la navegación, como podría ser el sistema de visión por computador y mapeado 3D mediante técnicas láser. Otra opción sería tratar de integrar este módulo desarrollado en el conjunto del sistema compuesto por módulos del proyecto general.

Como mejoras se podría plantear la incorporación de una navegación ciega, donde mediante la ayuda de los sensores se fuese generando el grafo recogiendo toda la información. Los nodos desconocidos planteados en este trabajo se podrían continuar explorando entonces.

Una mejora de la identificación de OI sería también una gran aportación, desarrollando la aplicación de visión por computador, asociada a una base de datos, que permitiese el registro de OI, y su posterior identificación.

El uso de la odometría durante la navegación del robot mejoraría las prestaciones del algoritmo. La pose del robot podría añadirse al módulo de comparación, dependiendo el índice de similitud entre nodos de la pose del robot también junto a las otras características.

Es viable en un futuro adaptar este algoritmo a un entorno real, mediante la incorporación a un software de un robot, debido a la gran modularidad del programa diseñado. Este robot podría navegar de forma autónoma en un entorno controlado haciendo uso del software, permitiendo observar directamente la toma de decisiones en tiempo real.

Este algoritmo se podría aplicar a otros entornos, no solo a la navegación submarina, ya que se apoya en mapas topológicos. Podrían añadirse nuevos elementos al grafo que interfiriesen en la toma de decisiones, como evaluación de situaciones excepcionales, detección de obstáculos o simulaciones de averías.

5.3. Impacto

La aplicación de el software desarrollado en entornos reales podría colaborar en el diseño de robots autónomos capaces de navegar y explorar numerosos entornos.

Asociado con el proyecto UNEXMIN, si este software formase parte de él, contribuiría directamente a su objetivo. Esto permitiría la exploración y documentación de minas inundadas abandonadas, actualizando sus mapas y estados y permitiendo a profesionales volver a evaluar su apertura aplicando los novedosos métodos de extracción que antes no existían.

En el caso de que mediante este robot se produjese la reapertura de alguna mina, esto supondría una nueva fuente de materias primas. Esto liberaría ligeramente al mercado europeo de la dependencia de estos materiales mediante importaciones. La liberación de importaciones hace a Europa más independiente de otras naciones, con las que a veces no interesa comerciar por motivos políticos pero es necesario ya que son casi los únicos proveedores.

Este trabajo contribuye al desarrollo de robots autónomos, línea que se encuentra en auge en la actualidad y en continuo crecimiento. La navegación topológica es una parte importante del desarrollo de estos y como tal tiene su impacto.

Mediante la aplicación de este trabajo se pueden ahorrar costes y tiempo en planificación de rutas dado a su rapidez de evaluación dado un entorno conocido.

Dado a su modularidad, el programa diseñado puede incorporarse en multitud de proyectos con facilidad. Esto facilitaría su desarrollo y añadiría una parte vital en muchos otros.

5.4. Responsabilidad legal, ética y profesional

Con todo desarrollo en el ámbito de la ingeniería, y sobretodo de la robótica, conviene hacer una reflexión sobre los aspectos éticos asociados este. La responsabilidad legal, ética y profesional es una realidad adjunta a todo trabajo, pues este tiene un impacto.

En cuanto a la responsabilidad legal, el desarrollo de este Trabajo de Fin de Máster no atiende a ninguna restricción legal, pues se ha desarrollado íntegramente en forma de software, y trabaja sobre simulaciones, las cuales no afectan a ningún entorno real.

Sí cambiaría si el software diseñado se aplicase a entornos reales, realizando pequeñas

modificaciones, pero dicho supuesto queda fuera del alcance del objetivo de este trabajo.

En cuanto a ámbitos éticos y profesionales, se pueden llegar a numerosos planteamientos. La introducción de un cuerpo extraño como es un robot, en entornos que antes eran explotaciones pero tras su inundación se han convertido en entornos naturales genera discusiones medioambientales.

Si bien es cierto que este robot trata de explorar el entorno sin ser invasivo, provocando el mínimo cambio en el entorno, las corrientes que su desplazamiento puede generar así como la luz que emite puede afectar a la fauna y flora generada en el entorno.

Otro aspecto a evaluar sería el impacto del abandono del robot en la mina en caso de fallo fatal, o el coste y complejidad de su recuperación. En el caso de que se abandonase el robot en el interior de la mina inundada, este generaría contaminación, y existiría riesgo de explosión por la degeneración de las baterías.

Volviendo a la simulación, en esta no se genera ningún dilema moral ni ningún compromiso físico, ya que es íntegramente simulación por medio de software.

El software diseñado podría adaptarse utilizándose para otros fines más allá de la exploración, como podría ser un supuesto transporte de personas o materiales mediante su algoritmo de toma de decisiones y selección de ruta. En dicho caso se plantearían problemas como tener en cuenta las dimensiones del entorno para una posible evacuación en caso de fallo, o comprobaciones de que el material transportado no se pierde durante el transporte.

Capítulo 6

Planificación temporal y presupuesto

Planificación temporal

La planificación temporal resulta crucial en todo proyecto, pues da estructura a las tareas de trabajo y sirve de referencia a medida que se desarrolla el proyecto.

En la Figura 6.1 se puede observar el desarrollo del diagrama de Gantt del proyecto. Este se ha dividido en tareas, distribuidas en el tiempo. Se ve claramente como unas tareas dependen de la finalización de otras para su comienzo.

En la Figura 6.2 se muestra el EDP, o diagrama de la estructura de descomposición del proyecto. En él, se observa como queda el proyecto descompuesto en partes, y como estas a su vez se desglosan en pequeños módulos que las componen. Esta imagen ofrece una vista general del proyecto y de su desarrollo.

Presupuesto

Para el desarrollo del algoritmo que compone este trabajo, sólo ha sido necesario el uso de software, y el hardware en el que este funciona, en este caso un ordenador únicamente. Al coste de del ordenador hay que añadirle la licencia de software de Windows 10.

Dado a que se trata de desarrollo software y se prueba mediante simulación, no se ha requerido instrumentación alguna o uso de un robot físico.

El programa utilizado ha sido QtCreator, en su versión gratuita, por lo que no añade coste alguno. En el se ha desarrollado todo el programa en C++.

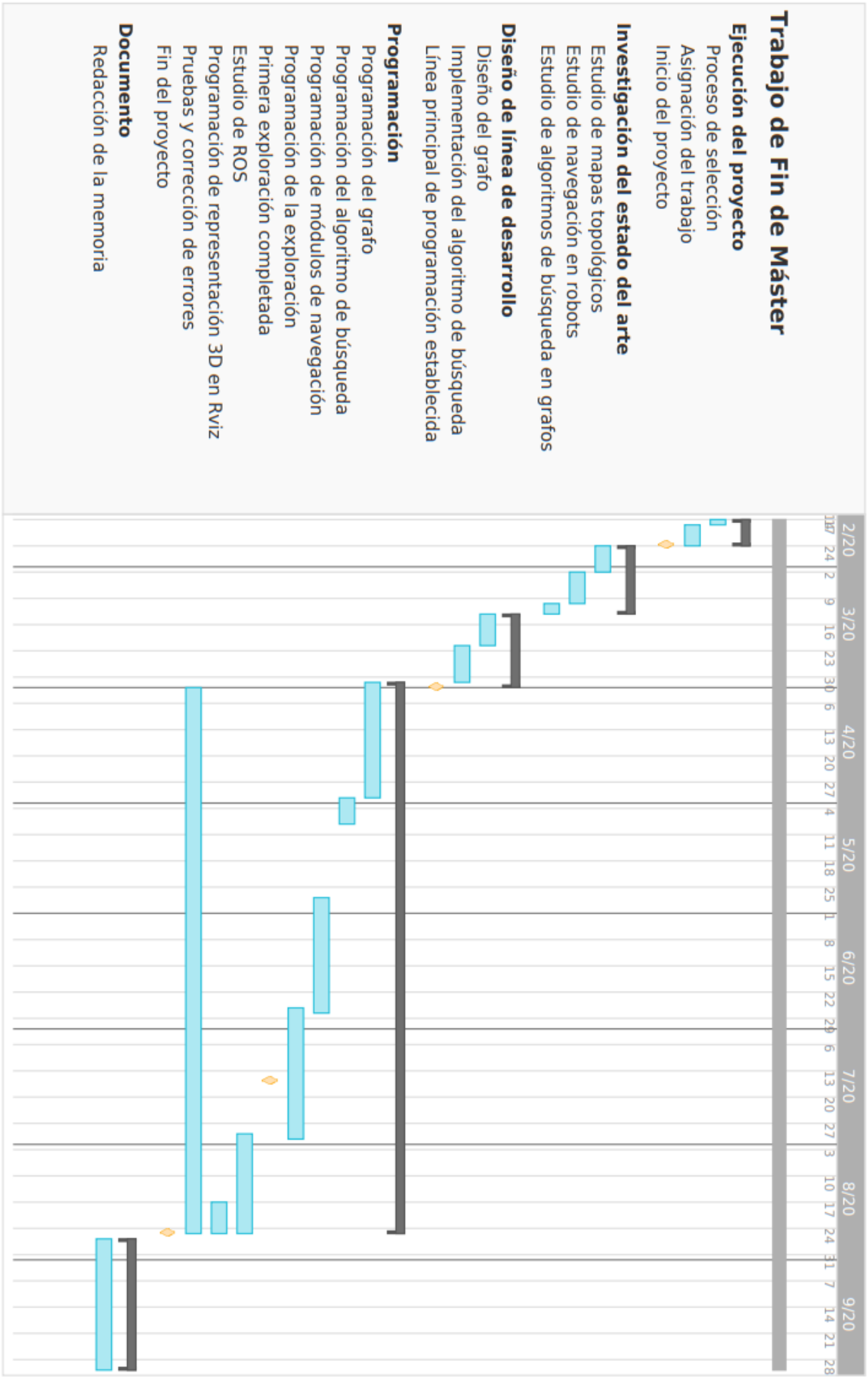


Figura 6.1 Diagrama de Gantt del proyecto.

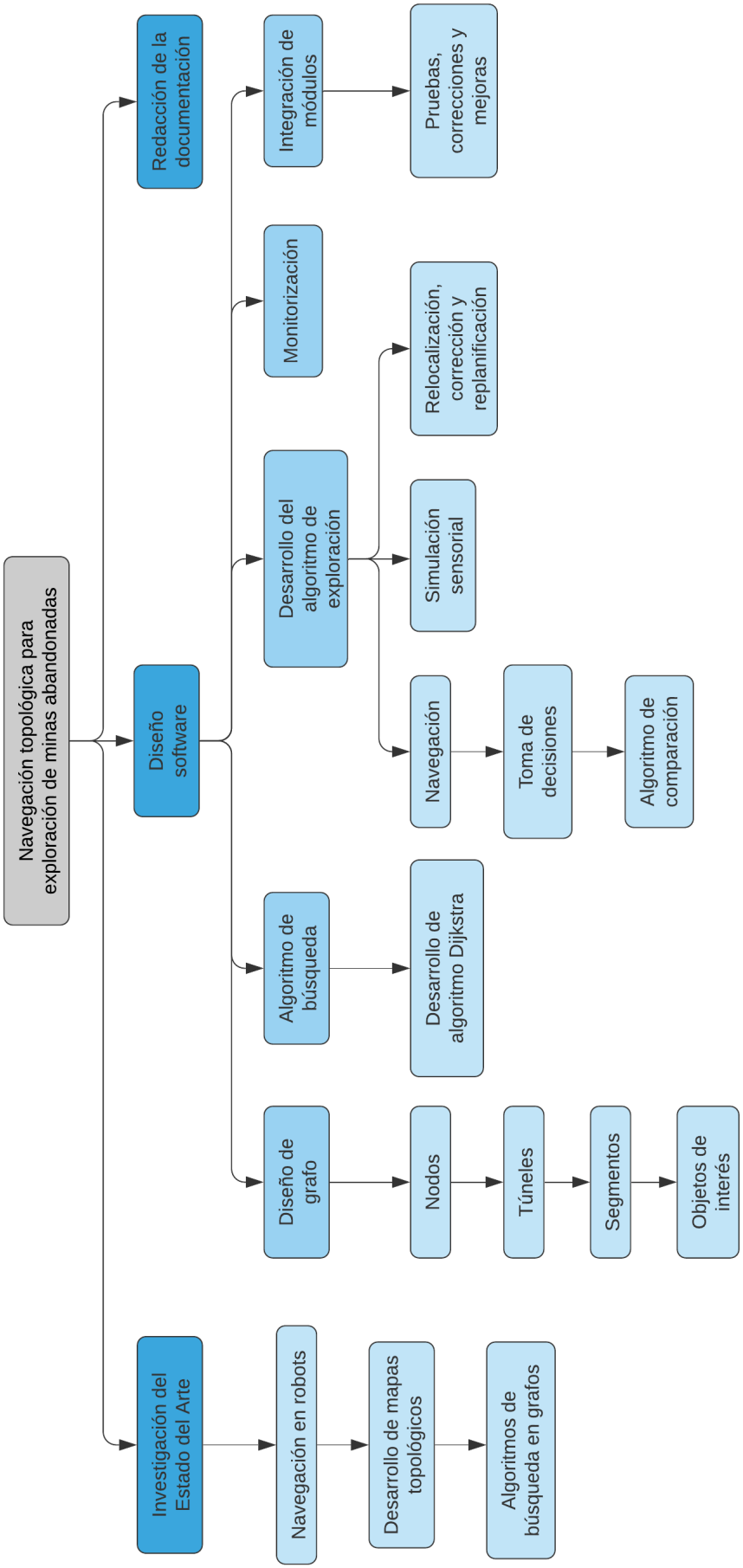


Figura 6.2 Estructura de descomposición del proyecto.

Los recursos utilizados y su coste se analizan en la siguiente tabla:

Tabla 6.1 Tabla de costes del trabajo.

Descripción	Precio(€)/unidad	Unidades	Coste (€)
Ordenador	1	550	550
Licencia Windows 10	1	13	13
QtCreator	0	1	0
Mano de obra Ingeniero Junior	11	380	4180
Tutoría	20	25	500
Coste total			5243 €

Capítulo 7

Bibliografía

[1] H. Awasthi. *Classical Search Algorithms*. Medium, 2018.

[2] P. L. N. Carrasco, F. Bonin-Font, M. M. Campos y G. O. Codina. *Stereo-Vision Graph-SLAM for Robust Navigation of the AUV SPARUS II*. 2015.

[3] COM. *The raw materials initiative.* 699. 2008.

[4] European Commission. *Autonomous Underwater Explorer for Flooded Mines*. `https://cordis.europa.eu/project/id/690008`. 2015.

[5] R. A. S. Fernández, Z. Milošević, S. Dominguez y C. Rossi. *Motion Control of Underwater Mine Explorer Robot UX-1: Field Trials*. 2019. DOI: `10.1109/ACCESS.2019.2930544`.

[6] J. Hartmann, J. H. Klussendorff y E. Maehle. *A Unified Visual Graph-Based Approach to Navigation for Wheeled Mobile Robots*. 2013.

[7] M. E. Jefferies y W.-K. Yeap. *Robotics and Cognitive Approaches to Spatial Mapping*. 2008.

[8] K. Konolige, E. Marder-Eppstein, B. Marthi y W. Garage. *Navigation in Hybrid Metric–Topological Maps*. 2011.

[9] I. Kostavelis, K. Charalampous, A. Gasteratos y J. K. Tsotsos. *Robot navigation via spatial and temporal coherent semantic maps*. 2015.

[10] L. Lopes, N. Zajzon, S. Henley, C. Vörös, A. Martins y J. Almeida. *UNEXMIN: a new concept to sustainably obtain geological information from flooded mines*. 44. European Geologist, 2017, págs. 54-57.

[11] T. Lozano-Pérez y M. Wesley. *An algorithm for planning collision-free paths among polyhedral obstacles.* 1979.

[12] M. N. Mahyuddin y M. R. Arshad. *Classes of Control Architectures for AUV: A brief survey*. 2008.

[13] S. Mussmann y A. See. *Graph Search Algorithms*. Standford, s.f.

[14] M. Pfingsthorn, B. Slamet y A. Visser. *A Scalable Hybrid Multi-Robot SLAM Method for Highly Detailed Maps.* 2007.

[15] D. Portugal y R. P. Rocha. *Extracting topological information from grid maps for robot navigation*. 2012.

[16] T. Praczyk. *Using Neuro–Evolutionary Techniques to Tune Odometric Navigational System of Small Biomimetic Autonomous Underwater Vehicle – Preliminary Report*. 2020.

[17] ROS Industrial. *ROS Qt Creator Plug-in wiki*. `https://ros-qtc-plugin.readthedocs.io/en/latest/`. 2017.

[18] ROS.org. *ROS Documentation*. `http://wiki.ros.org/Documentation`. s.f.

[19] C. Rossi. *Algoritmos de búsqueda*. Universidad Politécnica de Madrid, 2019.

[20] S. Schwertfeger y A. Birk. *Map Evaluation using Matched Topology Graphs*. 2016.

[21] G. Stasi, I. Fernandez e Y. Vanbrabant. *Exploration of flooded mines as support to the battery industry and the energy transition in Europe*. `https://eurogeologists.eu/stasi-exploration-of-flooded-mines-as-support-to-the-battery-industry-and-the-energy-transition-in-europe/`. 2015.

[22] R. Szabó. *Mobil robotok szimulációja*. 2001.

[23] R. Szabó. *Topological Navigation of Simulated Robots using Occupancy Grid*. 2004.

[24] S. Thrun. *Learning metric-topological maps for indoor mobile robot navigation*. 1998.

[25] UNEXMIN. *An autonomous underwater explorer for flooded mines*. `https://www.unexmin.eu/`. s.f.

[26] J. Wallgrun. *Hierarchical voronoi-based route graph representations for planning, spatial reasoning, and communication.* 2004.

[27] J. Yin, L. Carlone, S. Rosa y B. Bona. *Graph-based Robust Localization and Mapping for Autonomous Mobile Robotic Navigation*. 2014.

Apéndice A

Monitorización de una exploración completa

Terminal:

```
Do you want to perform a path search between two nodes? (Y/N).
y
Select start mode.
1. Lost start mode
2. Known start node
_____________________
Mode:2
Set start node:
0
Set goal node:
18
------Path search required------

Finding path from node 0 to node 18.
Solution node sequence: 0 1 5 10 11 14 18



[ INFO] [1601073273.579088018]: Sleeping 5 seconds before running

------Exploration------

Initial node: 0
Tunnel transition.

Tunnel:0
*****Impresion de los datos del tunel recortada*****



Node 1 arrived.
Nodes to compare:  real (1) and expected (1)
Node check: P = 0.813027
Position uncertainty = 0.813027
Tunnel transition.
```

```
Tunnel:1
*****Impresion de los datos del tunel recortada*****



Node 5 arrived.
Nodes to compare:  real (5) and expected (5)
Node check: P = 0.879953
OOI ID_2 identifyed.
Checking OOI with the database.
Position confirmed by the OOI.
Position uncertainty = 1
Tunnel transition.

Tunnel:17
*****Impresion de los datos del tunel recortada*****

Node 10 arrived.
Nodes to compare:  real (10) and expected (10)
Node check: P = 0.819836
Position uncertainty = 0.819836
Tunnel transition.

Tunnel:18
*****Impresion de los datos del tunel recortada*****

Node 11 arrived.
Nodes to compare:  real (11) and expected (11)
Node check: P = 0.888516
Position uncertainty = 0.728437
Tunnel transition.

Tunnel:23
*****Impresion de los datos del tunel recortada*****

OOI ID_55 identifyed.
Checking OOI with the database.
Position in tunnel confirmed by the OOI.
Position uncertainty = 1
Node 14 arrived.
Nodes to compare:  real (14) and expected (14)
Node check: P = 0.861263
Position uncertainty = 1
Tunnel transition.
***BAD TUNNEL CHOSEN***.

Tunnel:23
*****Impresion de los datos del tunel recortada*****

Node 11 arrived.
Nodes to compare:  real (11) and expected (18)
Node check: P = 0.18305
Exits number different.
Re-scaning.
Nodes to compare:  real (11) and expected (18)
Number of exits different by 1. Penalization.
Node 18 missed. Checking for mistakes in previous tunnel choice.
Checking the map for detecting tunnel mistakes.
```

```
Mistaken tunnel choosed confirmed by map simmilarities.
------Path search required------

Finding path from node 11 to node 18.
Solution node sequence: 11 14 18


------Exploration------

Initial node: 11
Tunnel transition.

Tunnel:23
*****Impresion de los datos del tunel recortada*****

Node 14 arrived.
Nodes to compare:  real (14) and expected (14)
Node check: P = 0.861263
Position uncertainty = 0.861263
Tunnel transition.

Tunnel:31
*****Impresion de los datos del tunel recortada*****

Node 18 arrived.
Nodes to compare:  real (18) and expected (18)
Node check: P = 0.841631
Position uncertainty = 0.724866

***** End node reached correctly *****
Ending program...

[ INFO] [1601073335.404734221]: Shutting down.
[rviz_visual_tools_demo-1] process has finished cleanly
all processes on machine have died, roslaunch will exit
shutting down processing monitor...
... shutting down processing monitor complete
done
```