Universidad Zaragoza
1542

Trabajo Fin de Grado

Estrategias Óptimas de Exploración de Grafos Parcialmente Desconocidos

Optimal Traversal Strategies in Partially Unknown Graphs

Autor

Alberto Sancho Pina

Directores

Julio Alberto Placed Perales

José Ángel Castellanos Gómez

Escuela de Ingeniería y Arquitectura

2021

Resumen

Durante los últimos años, muchas de las tareas en el campo de la robótica móvil se han comenzado a resolver mediante una formulación basada en grafos, en las que los nodos codifican las poses del robot y los arcos las restricciones entre estas. Una de las aplicaciones de esta formulación, consiste en resolver el problema de exploración autónoma de entornos reales a través de la exploración de grafos, entendiendo esta última como el proceso por el cual se recorren los arcos del mismo hasta visitar sus vértices atendiendo a un determinado criterio. Esta herramienta, se puede utilizar para resolver el paradigma de SLAM (*Simultaneous Localisation and Mapping*) Activo, el cual hace referencia a la capacidad por parte del robot de elegir la secuencia de movimientos que le permita minimizar la incertidumbre de su localización y del entorno que está reconstruyendo mientras ejecuta SLAM.

En este Trabajo Fin de Grado se han estudiado, implementado y evaluado distintos algoritmos de planificación de rutas en grafos, con el objetivo de comparar las ventajas y limitaciones de cada uno de ellos, y de encontrar el camino que ofrezca la mínima incertidumbre para así poder resolver el problema de SLAM Activo. Para ello, se han analizado varios grafos basados en poses construidos a partir de entornos reales. Además, se ha estudiado el uso de varias métricas de la matriz de covarianza, con el objetivo de analizar cuál es la que permite encontrar el conjunto de acciones que proporcione una incertidumbre óptima durante las tareas de exploración.

Se ha conseguido implementar un algoritmo que representa el estado del arte que es capaz de encontrar el camino de mínima incertidumbre entre dos puntos. Además, se ha demostrado que la única manera de garantizar estas trayectorias es incluyendo una métrica de la misma en el algoritmo de búsqueda. En concreto, la que se basa en el determinante de la matriz de covarianza o de incertidumbre (criterio de optimalidad D). Finalmente, se ha comprobado que cuando se trabaja con una representación reducida del grafo se obtiene una clara mejora en los tiempos de cómputo al realizar varias búsquedas.

Abstract

Recently, many problems in the mobile robotics community have started to be resolved using a formulation based on graphs, in which nodes encode the robot poses and edges encode the constraints between them. One of the applications of this formulation is to solve the problem of autonomous exploration in real environments, through the exploration of graphs; understanding the latter as the process by which the edges are traversed until reaching the nodes, attending to a certain criteria. This tool can be used to solve the Active SLAM (Simultaneous Localisation and Mapping) paradigm, which refers to the ability of the robot to choose a sequence of movements that allows to minimise the uncertainty of its location and that of the enviroment representation (map) while executing SLAM.

In this thesis, different path planning algorithms have been studied, implemented and evaluated in order to compare the advantages and limitations of each of them; and to find the path that offers minimum uncertainty to solve the Active SLAM problem. To do so, several pose graphs built from real enviroments have been analysed. In addition, the use of different metrics from the covariance matrix has been studied, to find which one produces the set of actions with minimum uncertainty during exploration tasks.

A state-of-the-art algorithm that is capable of finding the path of minimum uncertainty between two points has been implemented. Moreover, we have proved that the only way to guarantee these trajectories' optimality is by including a metric of the uncertainty in the search algorithm. Specifically, the one based on the determinant of the covariance/information matrix (D-optimality criterion). Finally, we have shown that working with a reduced representation of the graph results in a clear reduction of the computational load when carrying out several searches.

ÍNDICE

1. Introducción

En este capítulo se presenta el objetivo, contexto, alcance y estructura de este Trabajo Fin de Grado.

1.1. Contexto, Alcance y Objetivos

En los últimos años, la construcción de mapas de entornos sin la ayuda de sistemas de posicionamiento externos ha sido uno de los focos de investigación de la comunidad robótica. En este contexto se presenta el problema de SLAM (*Simultaneous Localisation and Mapping*) como la capacidad por parte del robot de conocer cuál es su localización mientras construye el mapa del entorno por el que se está moviendo, convirtiéndose en un problema de los más importantes en la búsqueda de la construcción de vehículos realmente autónomos. Si al problema anterior se le suma la capacidad por parte del robot de elegir el conjunto de movimientos óptimo mientras realiza tareas de exploración que le permita realizar la mejor reconstrucción posible del entorno (SLAM activo), la complejidad aumenta. Pese a que se trata de un problema que no es fácil de abordar, en la actualidad existen múltiples algoritmos basados en las medidas de los sensores que lleva equipado el robot, que tratan de mejorar la eficiencia en la resolución de este problema y que por tanto, pueden ayudar a crear nuevas aplicaciones de gran interés en el campo de la robótica móvil. Uno de los métodos más empleados es la formulación de SLAM basado en grafos (*graph* SLAM), con la que se trabaja en este proyecto.

Para resolver el problema anterior, es de gran importancia poder calcular la trayectoria de mínima incertidumbre entre dos puntos. En este Trabajo Fin de Grado se va a tratar la exploración de grafos con el objetivo de encontrar la trayectoria según la premisa anterior, para lo cual es necesario introducir una métrica de la incertidumbre dentro del propio algoritmo. Para ello, se pretende implementar y comparar distintos algoritmos de planificación de rutas tanto clásicos como aquellos que representan el estado del arte y evaluarlos en términos de la incertidumbre en grafos ya existentes provenientes de entornos reales (parcialmente desconocidos).

Este trabajo se enmarca dentro del grupo de Robótica, Percepción y Tiempo Real del DIIS (Departamento de Informática e Ingeniería de Sistemas) y del I3A (Instituto Universitario de Investigación en Ingeniería de Aragón). Como se explica en el Capítulo 3, se apoya principalmente en la herramienta *Graph and Networks Algorithm* que dispone Matlab y en la información de varios grafos elaborados a partir

de entornos reales, así como en el algoritmo FAMUS (*Fast Minimum Uncertainty Search*) [1], para abordar los problemas que se han planteado. Cabe destacar que el alcance de este trabajo se enmarca dentro de los objetivos propuestos, quedando fuera de este alcance la creación de grafos reales.

1.2. Estructura

La memoria se ha estructurado de la siguiente manera:

- En el Capítulo 1 se explica el contexto en el que se realiza el trabajo, así como los objetivos, alcance y estructura de la memoria.
- En el Capítulo 2 se explican los conceptos de SLAM y SLAM activo, y se analizan en qué consisten los diferentes métodos existentes para resolver este problema. De entre ellos, se estudia en profundidad la formulación del problema mediante grafos basados en poses. Posteriormente se presentan los criterios de optimalidad que permiten caracterizar la incertidumbre de las poses del grafo, y se hace una breve introducción a la teoría de grafos.
- El Capítulo 3 contiene diferentes *pose graphs* construidos a partir de entornos reales, a partir de los cuales se analizan distintos métodos de planificación de rutas. Estos son los que tratan de encontrar el camino con menor número de nodos, menor distancia euclídea y menor incertidumbre (sobre una representación reducida del grafo), para lo cual se utilizan los algoritmos *Breadth First Search*, Dijkstra y FAMUS respectivamente.
- En el Capítulo 4 se estudia el potencial y la limitación de cada algoritmo (del capítulo anterior) mediante una comparación entre ellos, y se propone aquel que sea capaz de paliar los inconvenientes de gran parte de ellos. Posteriormente se realiza una comparación entre los distintos criterios de optimalidad que permiten caracterizar la información o incertidumbre de los nodos de un grafo, para lo cual se han utilizado un gran número de grafos basados en poses distintos entre sí.
- Finalmente, el Capítulo 5 recoge las principales conclusiones del trabajo, así como ideas de trabajo futuro.

Adicionalmente, se incluyen los siguientes anexos en el trabajo:

- El Anexo I hace hincapié en el problema de optimización de grafos y presenta el problema que se conoce como de asociación de datos.
- El Anexo II contiene un breve caso práctico de grafo no dirigido basado en poses sobre el que se calculan las principales matrices que permiten

caracterizarlo y a partir de las cuales se verifican las relaciones que guardan entre ellas.

- En los anexos III y IV se explica con un mayor nivel de detalle en qué consisten los algoritmos *Breadth First Search* y Dijkstra respectivamente, y se ilustran con un breve caso práctico cada uno de ellos.
- El Anexo V contiene el pseudocódigo con el que se ha elaborado el algoritmo de FAMUS en Matlab.
- Finalmente, en el Anexo VI se muestran tres ejemplos de aplicación del algoritmo de reducción de grafos.

2. Antecedentes

2.1. SLAM.

En este trabajo se van a abordar el SLAM (*Simultaneous Localisation and Mapping*) y el SLAM activo.

Muchas de las tareas que se le pueden exigir a un robot móvil sin ayuda de un sistema de referencia externo requieren de un mapa del entorno. Partiendo de un robot que se encuentra en una localización y un entorno totalmente desconocidos, el objetivo del SLAM es poder construir un mapa de dicho entorno $\mathcal{M}$, a la vez que se calcula la trayectoria por la que pasa el robot $X_{0:T}$, a partir de las mediciones u observaciones realizadas a través de los sensores sin ayuda de ningún tipo de sistema de referencia externo como pudiera ser el GPS. Para ello, es necesario conocer en cada instante de tiempo dónde está el robot.

Este problema es uno de los más populares en el campo de la robótica móvil, sobre el que se ha estado trabajando durante las últimas décadas. Trabajos recientes han ayudado a mejorar la eficiencia en la resolución de esta tarea. En la actualidad existe una gran variedad de métodos para resolver el problema del SLAM. Estos métodos se pueden clasificar principalmente en dos categorías: de filtrado o de optimización. Los primeros, también conocidos como SLAM en línea, utilizan lo que se denomina una estimación de estados en línea, de modo que el problema se define como la probabilidad de que el estado del robot y el mapa del entorno sean, respectivamente, x_t y $\mathcal{M}$ a partir de la última observación (z_t) y acción (u_t) realizadas por el robot [2]:

$$P(x_t, \mathcal{M} \mid z_t, u_t) \tag{1}$$

Los algoritmos que utilizan SLAM en línea suelen ser incrementales, de modo que procesan un único conjunto de datos en cada iteración. La incorporación de nuevas medidas ayuda a mejorar la precisión en la estimación del estado del sistema. Filtros de información, de partículas o de Kalman, son algunas de las técnicas más usadas.

Por otro lado, los métodos de optimización consisten en estimar la trayectoria completa del robot ($X_{0:T}$) a partir de todas las medidas realizadas por el robot hasta el momento, es decir, se trata de calcular la probabilidad sobre $X_{0:T}$ y $\mathcal{M}$ a partir de los datos disponibles:

$$P(X_{0:T}, \mathcal{M} \mid Z_{1:T}, U_{1:T}) \quad (2)$$

Donde $Z_{1:T}$ y $U_{1:T}$ representan respectivamente el conjunto de observaciones y acciones realizadas por el robot hasta el momento. Estos métodos resuelven lo que se denomina el problema del SLAM completo.

Además, para resolver el problema del SLAM, es necesario definir un modelo que permita relacionar las acciones con las ubicaciones del robot, y un modelo que relacione las observaciones del entorno realizadas por el robot con las posiciones de este. Estos modelos se pueden representar mediante distribuciones probabilísticas, $P(x_t|x_{t-1}, u_t)$ y $P(z_t|x_t, \mathcal{M})$ respectivamente. Es decir, a partir de la acción u_t se calcula cual es la probabilidad de que el robot se encuentre en una pose determinada x_t, si anteriormente se encontraba en la pose x_{t-1}. Del mismo modo, se pretende calcular a partir del mapa del entorno $\mathcal{M}$, cual es la probabilidad de realizar la observación z_t sabiendo que el robot se encuentra en la ubicación x_t.

Uno de los métodos más utilizados para resolver el problema del SLAM, y en el que se centra este trabajo, es el SLAM basado en grafos (*Graph SLAM*). Se trata de un método de optimización que consiste en representar el mapa en el que se encuentra situado el robot mediante una serie de nodos y arcos que codifican, respectivamente, las localizaciones del robot a lo largo del tiempo y/o marcas en el entorno, y las medidas realizadas por los sensores. A su vez se puede trabajar con dos tipos de grafos: grafos que trabajan únicamente con poses (lo que incluye la posición y orientación) del robot (*pose graph*) y los denominados grafos completos.

El grafo basado en poses es en el que se va a centrar este trabajo. Los nodos representan únicamente las poses del robot que forman parte de la trayectoria. Por otro lado, los arcos pueden ser de dos tipos: de odometría o de re-observación. Los primeros son aquellos que tienen asociados una medida que se corresponde con el movimiento del robot a lo largo del tiempo, y que por tanto unen nodos consecutivos. En cuanto a los arcos de re-observación, estos se obtienen a partir de las observaciones realizadas por el robot sobre el entorno. Cuando desde dos nodos no consecutivos, el robot observa una misma marca (o un conjunto de ellas), se genera un arco de re-observación entre ambos. En este momento, es cuando el algoritmo de SLAM puede realizar una optimización conocida como cierre de bucle (*loop closure* o *global bundle adjustment*), disminuyendo la incertidumbre asociada a los nodos.

Cabe destacar, que los nodos que definen a un grafo basado en poses vienen representados por su posición y orientación. La posición de cada vértice viene dada por las coordenadas cartesianas, mientras que la orientación mediante los ángulos de Euler, o, de forma equivalente, cuaternios. A su vez, los arcos contienen los

vectores diferenciales que definen la distancia relativa entre los nodos, así como la incertidumbre asociada a este movimiento relativo. Esta incertidumbre proviene del error en el cálculo por odometría de las posiciones del robot, o en la medición derivada de las observaciones realizadas por el robot sobre el entorno, dependiendo de si los arcos unen nodos consecutivos o no. Aunque esta incertidumbre está asociada a los nodos, se codifica mediante matrices de covarianza o de incertidumbre en los arcos.

La matriz de covarianza es una matriz cuadrada, simétrica y semi-definida positiva que suele venir representada por números pequeños, lo cual tiene lógica, puesto que la incertidumbre suele ser pequeña. El tamaño de la matriz es de dimensión tres cuando se trabaja en dos dimensiones, o por el contrario de dimensión seis si se trabaja en tres dimensiones. La matriz de información es la inversa de la matriz de covarianza (por lo que suele venir representada por números grandes), y lo que pretende es cuantificar la incertidumbre asociada a los arcos. Puesto que las matrices de covarianza son densas, el uso de estas puede suponer una limitación al algoritmo de SLAM con respecto a su capacidad de gestionar mapas con un tamaño grande, problema que no ocurre con las matrices de información, al ser más dispersas. El motivo por el que estas últimas son dispersas es porque no requieren de la representación directa de las dependencias estadísticas de todos los elementos como ocurre en las matrices de covarianza.

A partir de la matriz de información de cada uno de los arcos que conforman el grafo, se puede construir la matriz de información del grafo. Se trata de una matriz de bloques que tiene tantas filas y columnas como nodos tiene el grafo, de modo que cada bloque contiene la matriz de información del arco que une los nodos correspondientes a la posición de la fila y la columna.

En lo que respecta a los grafos completos, la principal diferencia proviene del hecho de que en este caso existen dos tipos de nodos y tres tipos de arcos. Existen nodos que representan las poses del robot a lo largo del tiempo, y nodos que representan marcas del entorno observadas por el robot a lo largo de su trayectoria. En ambos casos, todos los nodos tienen unos valores de posición que permiten posicionar los nodos en el mapa. Por otro lado, los arcos pueden ser de tres tipos: (a) arcos que unen nodos consecutivos y que por lo tanto representan mediciones derivadas del movimiento del robot, (b) arcos que unen una pose del robot con una marca del entorno y que por tanto representan una observación y (c) arcos de cerrado de bucle o de re-observación, que son aquellos que unen dos poses del robot desde las cuales se está observando una misma marca del entorno, o un conjunto de ellas.

En la Figura 1 [3] se muestra un ejemplo de grafo completo en el que se han dibujado los nodos mediante triángulos azules y estrellas amarillas, que representan respectivamente las poses del robot y las marcas que forman parte del mapa $\mathcal{M}$. A su vez, también se pueden distinguir los tres tipos de arcos que se han comentado: de odometría (color azul), de observación (línea punteada) y de re-observación (color rojo). Si únicamente se tuviese en cuenta la parte del grafo que representa la trayectoria del robot (línea discontinua), se tendría un ejemplo de grafo basado en poses.

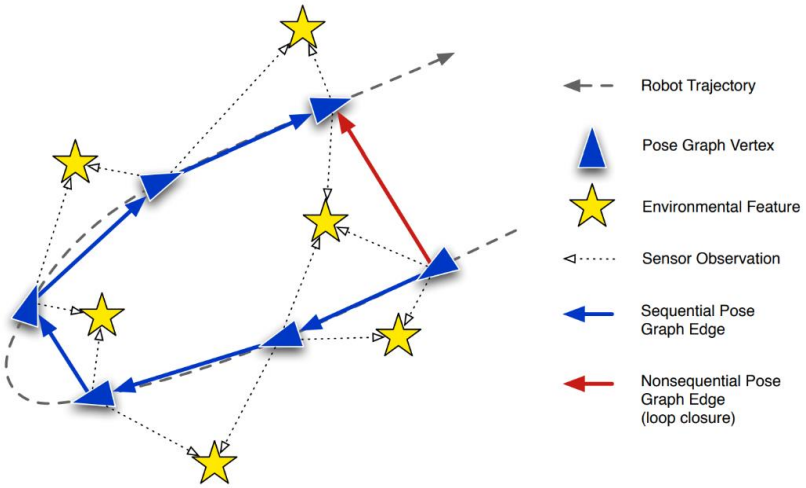

Figura 1. Ejemplo de grafo completo

Por tanto, la resolución del problema de SLAM basado en grafos consiste en resolver dos tareas. En primer lugar, es necesario construir un grafo a partir de las medidas realizadas por los sensores. Para ello es necesario resolver un problema probabilístico de estimación de la trayectoria del robot y las marcas del entorno, a partir de la posición inicial del robot y de las medidas asociadas a los arcos, ya sean de odometría o provenientes de observaciones de marcas del entorno.

$$P(X_{0:T}, \mathcal{M} |\ Z_{1:T}, U_{1:T}, X_0) \tag{3}$$

En segundo lugar, hay que encontrar la configuración más probable de los nodos a partir de los arcos que conforman el grafo. Este último paso es lo que se conoce como optimización del grafo y consiste en resolver un problema de minimización del error, con el objetivo de construir un mapa lo más parecido posible al real.

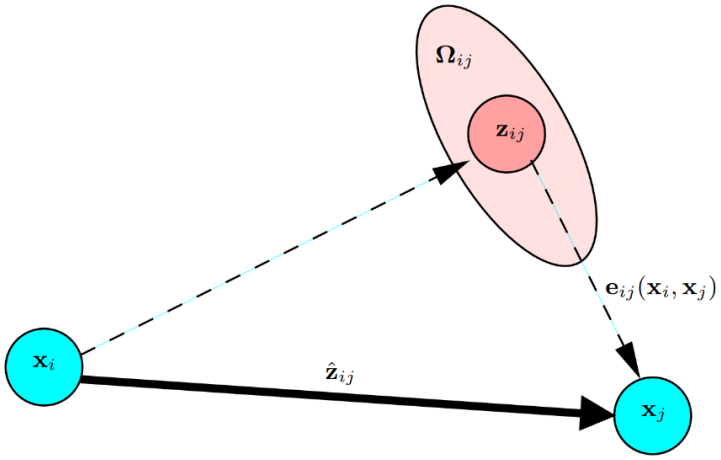

Figura 2. Problema de minimización de error

La Figura 2 [4] ayuda a entender este problema de optimización mediante un caso sencillo. Suponiendo que desde la pose x_i (que representa la posición del robot) se realiza una estimación de la pose x_j (a partir de observaciones relativas, por ejemplo), el objetivo es encontrar la configuración de los dos nodos que minimicen la diferencia entre la medida real (z_{ij}) y la estimada ($\hat{z}_{ij}$). Esta diferencia se conoce como función de error:

$$e_{ij}(x_i, x_j) = z_{ij} - \hat{z}_{ij}(x_i, x_j) \tag{4}$$

Cuanta más información se tenga del entorno como por ejemplo a partir de observaciones de marcas del entorno o mediciones de odometría entre poses sucesivas, menor será el error que se cometa en la estimación de las poses del robot. Las técnicas de minimización del error por mínimos cuadrados son las más habitualmente utilizadas para resolver este tipo de problemas.

En grafos en los que el número de nodos es elevado, el problema se vuelve complejo puesto que para encontrar las configuraciones de los nodos que minimizan el error se deben modificar nodos que, aunque puedan ayudar a mejorar una parte del grafo, pueden empeorar otra parte de este. Esta complejidad computacional en la resolución del problema de minimización hizo que el *graph* SLAM no se hiciera popular hasta varios años después de su formulación en 1997 por Lu y Millios [5]. Los avances en el campo del álgebra lineal y en la estructura del problema de SLAM, permitieron la aparición de nuevos enfoques eficientes en la resolución del problema de optimización que condujeron a que este método se convirtiera en una de las técnicas más avanzadas en cuanto a velocidad y precisión. Algoritmos como el de Gauss-Newton y el de Levenberg-Marquardt son algunos de los más utilizados hoy en día. Se puede encontrar más información acerca del problema del SLAM en [6], [7] y [8], y en sus referencias.

Supuesto un robot realizando tareas de exploración en un entorno desconocido, una vez que el grafo ya ha sido construido, puede que sea necesario realizar otro tipo de optimización a todo el grafo (*global bundle adjustment*) con el objetivo de que este se parezca lo máximo posible al entorno real. La principal diferencia con respecto a las optimizaciones que tienen lugar a partir de los arcos de cerrado de bucle es que en estas últimas el algoritmo únicamente optimiza los nodos que se encuentran dentro del bucle, a diferencia de la primera en la que se ejecuta una optimización a todos los nodos a partir de cada uno de los arcos del grafo.

En la Figura 3 se muestra un ejemplo mediante dos imágenes, ambas basadas en el entorno de *Intel*, antes (izquierda) y después (derecha) de hacer la optimización a todo el grafo, con el objetivo de comparar las diferencias entre ambos. Este grafo ha sido construido con ayuda de Matlab a partir de la información de los nodos y arcos recogida en [9]. La construcción de los grafos se explica con mayor detalle en la Sección 3.1 de esta memoria.

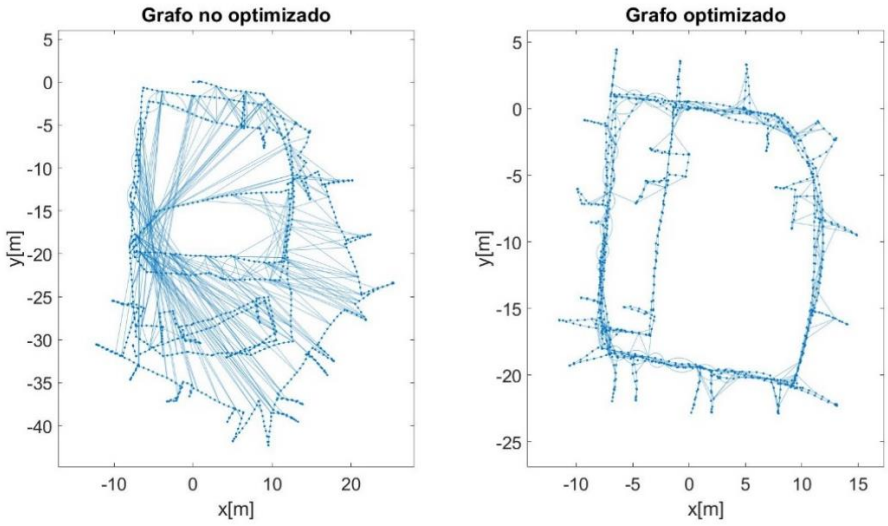

Figura 3. Comparación entre grafo antes (izquierda) y después (derecha) de la optimización

Aunque las imágenes de la figura anterior son bastante diferentes, no tiene por qué ser tan notable la diferencia entre un grafo optimizado y otro que no lo esté en todos los casos. Dependiendo del entorno que se esté analizando, la mejora tras la optimización será mejor o peor dependiendo del número de optimizaciones previas y lo buenas que sean, así como de la precisión en las medidas de los sensores que lleva equipado el robot.

En el Anexo I se muestra otro ejemplo de optimización de un grafo para un entorno diferente.

2.2. SLAM activo

Hasta ahora se ha definido el problema de SLAM como la capacidad del robot de construir un mapa del entorno y simultáneamente conocer en cada instante en que posición se encuentra. Sin embargo, no se han tenido en cuenta las decisiones que permiten al robot elegir hacia donde moverse.

El paradigma del SLAM activo se define como la capacidad del robot de elegir la mejor combinación de acciones de movimiento de entre todas las posibles que permita realizar la tarea de exploración autónoma mientras ejecuta el algoritmo de SLAM [10]. Es decir, supuesto un robot realizando una tarea de exploración autónoma, se pretende que el robot elija la siguiente mejor opción de exploración de entre todas las posibles con el objetivo de alcanzar un equilibrio entre re-visitar áreas por las que ya ha pasado (con el fin de conocer con un mayor nivel de detalle, y por tanto reducir la incertidumbre de la representación del mapa y de las poses del robot) y explorar nuevas áreas (*exploration-exploitation dilemma*) [11].

El SLAM activo se puede dividir en tres tareas [12]. En primer lugar, el robot identifica todas las posibles localizaciones que puede visitar basándose en su posición actual. Posteriormente, el robot realiza una estimación de cuál es la mejor opción y la ejecuta. En este punto, se pretende que el robot escoja la opción que permita al algoritmo de SLAM tener un mayor conocimiento del entorno y de su localización. Es decir, suponiendo que un robot puede elegir viajar a dos posiciones distintas, si en una de ellas el robot tiene la posibilidad de ver una marca que ya había observado desde otra pose, esto podría generar un arco de cerrado de bucle que permitiría al algoritmo de SLAM ejecutar una optimización y por tanto minimizar el error del grafo que se está construyendo. En este ejemplo, viajar a esta pose podría ser la elección adecuada. Finalmente, tras ejecutar el conjunto de acciones considerado como óptimo, el robot decide si es necesario seguir realizando tareas de exploración, o si por el contrario ya ha terminado, de acuerdo a un criterio conocido como criterio de parada (*stopping criteria*).

En la segunda fase del SLAM activo, el robot calcula cual es la mejor opción de movimiento cuantificando la incertidumbre asociada a las poses del robot y al mapa. Para ello, se utilizan las matrices de covarianza y el mapa que se han estimado hasta el momento. Puesto que utilizar matrices de covarianza puede convertirse en una tarea computacionalmente pesada y por tanto difícil de tratar en enfoques de SLAM completo, la información recogida en estas matrices suele simplificarse por un único escalar [13]. Habitualmente, esta cuantificación se realiza en base a la Teoría del

Diseño Óptimo de Experimentos (*Theory of Optimal Experimental Design*, TOED) o en base a la Teoría de la Información (*Information Theory*, IT).

La caracterización de la matriz de covarianza mediante estos números escalares son las funciones de utilidad. Como se verá más adelante, los criterios de optimalidad son un tipo de estas funciones. En [14] se muestra un ejemplo de utilización de una función de utilidad basada en la entropías de Shannon y Rényi para ayudar al robot en su tarea de exploración autónoma.

2.2.1. Criterios de optimalidad

Como se ha explicado anteriormente, en el SLAM activo el robot trata de cuantificar la incertidumbre de todas las posibles opciones de movimiento con el objetivo de seleccionar la más adecuada. La incertidumbre está caracterizada por una matriz de covarianza semi-definida positiva $\Sigma \in \mathbb{R}^{\ell x \ell}$, siendo ℓ la dimensión del vector que se quiere estimar (esto es, la pose del robot).

Kiefer [15] en base a la Teoría del Diseño Óptimo de Experimentos (TOED) demostró la existencia de una familia de funciones $||\Sigma||_p \rightarrow \mathbb{R}$ capaces de cuantificar la incertidumbre mediante un escalar dependientes de un único parámetro *p*:

$$||\Sigma||_p = \begin{cases} \left(\frac{1}{\ell} \sum_{k=1}^{\ell} \lambda_k^p \right)^{\frac{1}{p}} & si\ p \neq 0, p \leq 1 \\ exp\left(\frac{1}{\ell} \sum_{k=1}^{\ell} log(\lambda_k) \right) & p = 0 \end{cases} \quad (5)$$

Siendo $(\lambda_1, \ldots, \lambda_\ell)$ los valores propios de Σ. Se deducen cuatro criterios de optimalidad de la ecuación anterior:

- El criterio T captura la media de la varianza y se define como la media de los valores propios de la matriz de interés.

$$T - opt = \frac{1}{\ell} \sum_{k=1}^{\ell} \lambda_k \quad (6)$$

- El criterio A captura la media armónica. Se calcula según indica la ecuación (7).

$$A - opt = \left(\frac{1}{\ell} \sum_{k=1}^{\ell} \lambda_k^{-1} \right)^{-1} \quad (7)$$

- El criterio D (ecuación (8)) captura el volumen de la (hiper)elipsoide de la matriz de incertidumbre asociada.

$$D - opt = exp\left(\frac{1}{\ell}\sum_{k=1}^{\ell} log(\lambda_k)\right) \quad (8)$$

- Los criterios E (ecuación (9)) y E* (ecuación (10)) capturan respectivamente el mínimo y el máximo valor propio.

$$E - opt = min(\lambda_k : k = 1, \ldots, \ell) \quad (9)$$

$$E^* - opt = max(\lambda_k : k = 1, \ldots, \ell) \quad (10)$$

Por tanto, la idea es, a partir de los valores propios que definen la matriz, utilizar una serie de criterios de optimalidad que permitan caracterizar lo mejor posible la matriz de la que se está extrayendo la información. De este modo se consigue trabajar con números escalares en lugar de matrices, lo que conlleva una reducción en la complejidad. No hay que perder de vista que los criterios de optimalidad no dejan de ser mapas escalares de una matriz, y por lo tanto al utilizarlos se está perdiendo información.

En la Figura 4 se representa mediante 1000 puntos de color azul una distribución normal con matriz de covarianza de dimensión dos $\mathcal{N}([2\ 3], [1\ 1{,}5;\ 1{,}5\ 3])$ que permite simular un sub-espacio del espacio físico de dos dimensiones en el que se podría encontrar el robot. En esta figura, se ha representado la covarianza de la distribución normal mediante una elipse de color naranja, así como las estimaciones de la covarianza de la distribución normal según los distintos criterios de optimalidad mediante elipses de varios colores. Por tanto, dependiendo del criterio de optimalidad utilizado, la elipse que aproxima la matriz de covarianza será diferente.

Suponiendo que se pretende utilizar el criterio E-opt (es decir aquel que captura el mínimo valor propio) para aproximar la matriz de covarianza, esto se traduce en una elipse que abarca un espacio más reducido que la distribución asociada a la matriz de covarianza. Es decir, con este criterio se está suponiendo que la incertidumbre es mucho menor que la real. Por el contrario, si se utiliza el criterio E*-opt, a diferencia del criterio anterior se está suponiendo que la incertidumbre es mucho mayor que la real puesto que se está utilizando una elipse que abarca una superficie bastante mayor que la distribución de la matriz.

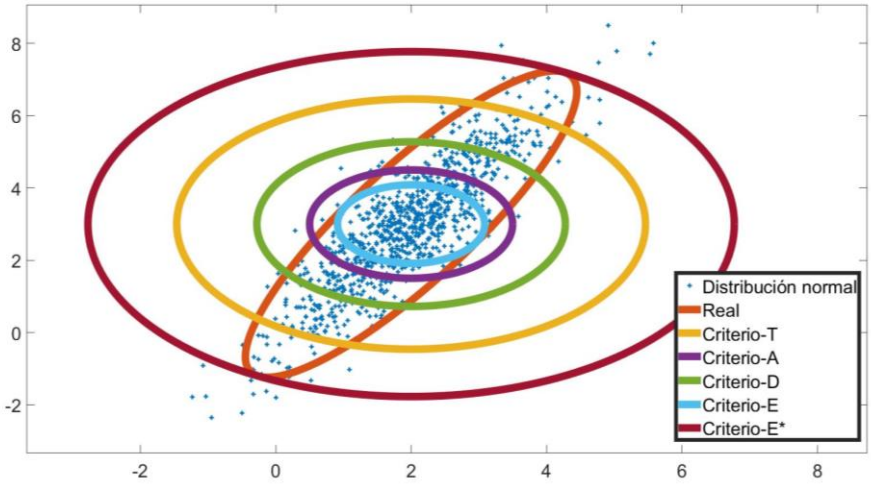

Figura 4. Representación de los distintos criterios de optimalidad

De todos los criterios que se han explicado, el D-opt es el más utilizado, puesto que es el único que captura el volumen de la matriz de covarianza completa mediante una aproximación elíptica.

Hay que destacar que este mismo análisis que se ha hecho a la matriz de covarianza, es aplicable a la matriz de información. Las expresiones que permiten relacionar los criterios de la matriz de covarianza con los de la matriz de información son las siguientes [16]:

$$T - opt(Y) = A - opt(\Sigma)^{-1} \tag{11}$$

$$D - opt(Y) = D - opt(\Sigma)^{-1} \tag{12}$$

$$A - opt(Y) = T - opt(\Sigma)^{-1} \tag{13}$$

$$E - opt(Y) = E^{*} - opt(\Sigma)^{-1} \tag{14}$$

2.3. Introducción a la teoría de grafos

Se entiende por grafo como el conjunto de objetos conocidos como vértices o nodos que se relacionan con otros nodos mediante arcos o aristas. El concepto de grafo se definió por primera vez en 1736 por Leonhard Euler [17] con el objetivo de analizar el problema de los puentes de Königsberg. A partir de ese momento se inició el desarrollo de una teoría que tiene multitud de aplicaciones hoy en día como el modelado de redes neuronales de un cerebro, planificación de producción, redes de comunicación, etc.

Los grafos se pueden clasificar en grafos dirigidos y no dirigidos. Los primeros están formados por un conjunto de vértices y arcos, de modo que cada arco asocia de manera unidireccional un par de nodos, por tanto, cualquier arco que forme parte del grafo se puede decir que tiene un destino hacia un nodo. Los arcos, dependiendo del sentido se pueden definir con carácter entrante o saliente. Por otro lado, los grafos no dirigidos están formados por un conjunto de vértices conectados mediante una serie de arcos de manera no direccional. En este tipo de grafos, los arcos pueden ser recorridos en cualquier dirección.

Además, estos dos tipos de grafos se pueden ponderar. Se denomina grafo pesado, ponderado o con costo, a aquel que tenga asociado a cada arco un valor que representa una magnitud como el peso, la longitud, la incertidumbre, etc.

En la Figura 5 se muestran un grafo no dirigido y un grafo dirigido ponderado a modo de ejemplo.

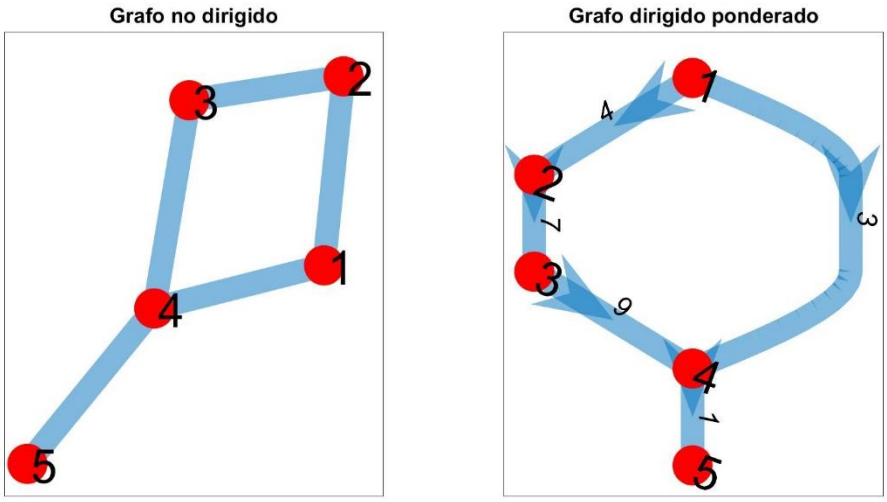

Figura 5. Ejemplo de grafo no dirigido (izquierda) y de grafo dirigido ponderado (derecha)

Una vez comentados los distintos tipos de grafos, se van a definir las matrices más utilizadas que proporcionan información sobre los grafos o sobre los arcos que lo conforman, y que por tanto permiten representarlos. Entre ellas destacan la matriz Laplaciana (L), de adyacencia (A), de grado (D) y de incidencia (Q).

Para entender mejor como se construye cada matriz, se va a trabajar sobre el grafo no dirigido de la Figura 5. Cuando se quiere conocer el número de arcos que conectan a cada nodo de un grafo, la matriz de grado resulta muy útil. Esta matriz recoge en la diagonal principal el número de arcos que conectan un nodo determinado. La matriz de grado de la Figura 5 es:

$$D = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Otra forma de representar la información recogida en un grafo es mediante la matriz de adyacencia. Al igual que antes, se trata de una matriz cuadrada cuyas filas y columnas representan los nodos del grafo. Aquellos nodos que están unidos entre sí mediante un arco se representan con 1 en la matriz, y con 0 para el caso contrario. Para el ejemplo del grafo anterior la matriz de adyacencia es:

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

En lo que respecta a la matriz de incidencia, se trata de una matriz rectangular que está formada por tantas filas y columnas como nodos y arcos tiene el grafo respectivamente. Cada columna representa un arco de modo que los únicos elementos no nulos son el nodo de origen y el de fin que se representan con -1 y 1 respectivamente. Para el ejemplo de la figura anterior la matriz de incidencia resultante es:

$$Q = \begin{pmatrix} -1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Finalmente, la matriz Laplaciana (L) es una matriz cuadrada que tiene tantas filas y columnas como nodos tiene el grafo. La diagonal principal de la matriz Laplaciana viene dada por el grado (es decir el número de vértices conectados al nodo) de cada uno de los nodos que forman parte del grafo, mientras que los elementos que se encuentran fuera de la diagonal principal se fijan a -1 si los nodos están conectados. En este caso la matriz sería:

$$L = \begin{pmatrix} 2 & -1 & 0 & -1 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ -1 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

Cabe destacar, que esta matriz es útil para recoger la información de un grafo sin pesos en los arcos. Para el caso en el que el grafo estuviera formado por arcos con pesos asociados, se emplearía la matriz Laplaciana pesada con el fin de que esta tuviera también en cuenta los pesos. En este caso, los elementos fuera de la diagonal principal de la matriz Laplaciana pesada se fijarían con el valor negativo del peso del arco si los vértices están conectados, mientras que los elementos que forman parte de la diagonal principal se calcularían como el sumatorio de los pesos de cada arco que conectan con el vértice correspondiente.

Las matrices anteriormente vistas se pueden relacionar entre sí para obtener la matriz Laplaciana:

$$L = D - A \tag{15}$$

$$L = Q * Q^T \tag{16}$$

En el Anexo II se muestra información adicional sobre esta matriz mediante un ejemplo en el que se elaboran y operan las diferentes matrices que representan un grafo sencillo.

3. Exploración óptima de grafos

En este capítulo se va a hablar sobre la búsqueda y selección de trayectorias óptimas en grafos complejos. A la hora de buscar una trayectoria que permita ir entre dos puntos conocidos del mapa de un entorno, se pueden tener en cuenta varios factores para seleccionar un camino u otro. En este capítulo se van a tener en cuenta tres criterios para la búsqueda y selección de trayectorias, con el objetivo de comparar para varios grafos y poses del robot, las distintas trayectorias alternativas obtenidas entre dos puntos, y poder seleccionar la que podría ser óptima para el robot cuando está ejecutando SLAM Activo. Esta comparación entre trayectorias es de utilidad cuando el robot está valorando distintos lugares a los que ir, o cuando el robot quiere ir de un punto a otro sobre un grafo que ya conoce. El primer criterio busca minimizar el número de nodos, mientras que el segundo busca las trayectorias con menor distancia euclídea entre los nodos de inicio y fin. Finalmente, se buscarán aquellas trayectorias entre un par de nodos que ofrezcan un camino con la mínima incertidumbre, para lo cual se utilizará el algoritmo FAMUS (*Fast Minimum Uncertainty Search*) [1].

Antes de trabajar con entornos complejos, se ha estudiado como construir y analizar grafos a partir de casos simples. En los anexos III, IV y V se dispone de información adicional sobre los tres algoritmos que se han utilizado para la búsqueda de trayectorias.

3.1. Importación de grafos reales

En este apartado de la memoria se van a importar varios conjuntos de datos de *pose graphs* de entornos reales al software de Matlab para su posterior representación y análisis. Se han utilizado los *datasets* en 2 dimensiones *Bicocca* [18], *New College* [19] e *Intel* [9].

Dependiendo del *dataset*, los ficheros pueden tener formato *g2o* [20] o *toro* [21]. En ambos casos, contienen todos los datos relacionados con los nodos y los arcos que conforman el grafo. En la primera mitad del fichero aparecen los nodos numerados empezando desde cero, con sus correspondientes coordenadas X e Y, y los ángulos θ que permiten reconstruir la trayectoria del robot. La segunda mitad del fichero contiene la información relativa a los arcos. En cada arco se definen los dos nodos que lo conforman, la distancia relativa entre ellos (referenciada al último nodo) y la matriz de información correspondiente a dicho desplazamiento.

En lo que respecta a la matriz de información, dependiendo si se trata de un fichero con formato *g2o* o *toro* vendrá definida en un orden diferente. En el caso de dos dimensiones, para *g2o* la información de cada arco se representa del siguiente modo:

$$dx\ dy\ d\theta\ I_{11}\ I_{12}\ I_{13}\ I_{22}\ I_{23}\ I_{33}$$

Mientras que para el formato *toro*:

$$dx\ dy\ d\theta\ I_{11}\ I_{12}\ I_{22}\ I_{33}\ I_{13}\ I_{23}$$

Siendo la matriz de información:

$$\mathrm{I}=\begin{pmatrix} I_{11} & I_{12} & I_{13} \\ I_{12} & I_{22} & I_{23} \\ I_{13} & I_{23} & I_{33} \end{pmatrix}$$

Para representar, modificar y analizar los diferentes grafos que se han utilizado se ha hecho uso de la herramienta de Matlab *Graph and Network Algorithms* [22]. En las figuras 6, 7 y 8 se muestran los grafos de los entornos *Bicocca*, *New College* e *Intel*, respectivamente.

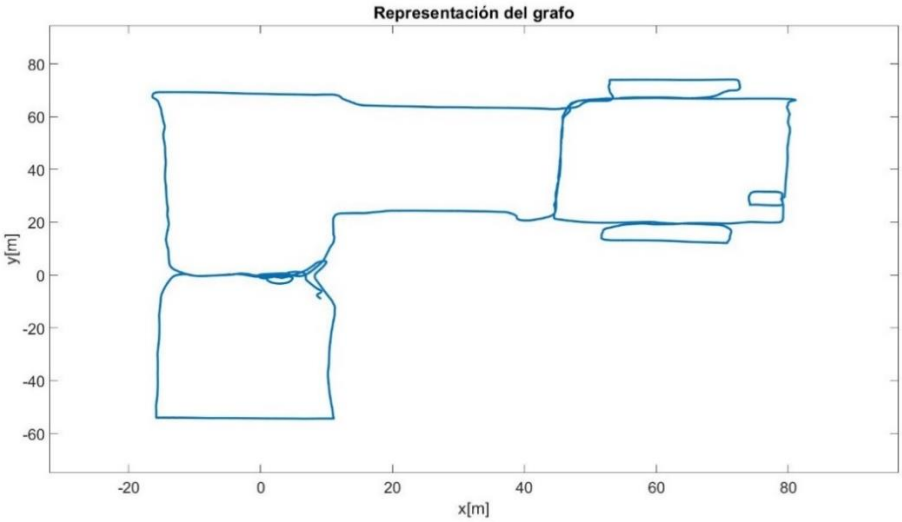

Figura 6. Representación del grafo de Bicocca

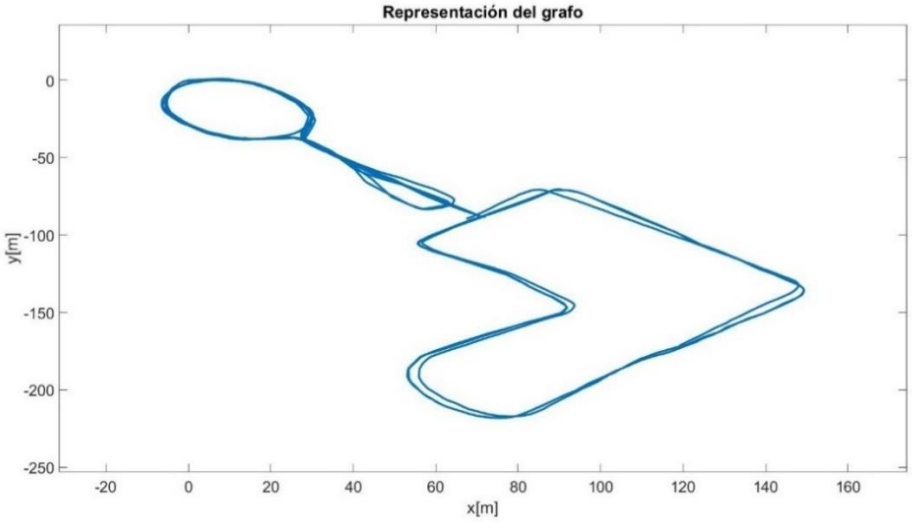

Figura 7. Representación del grafo de New College

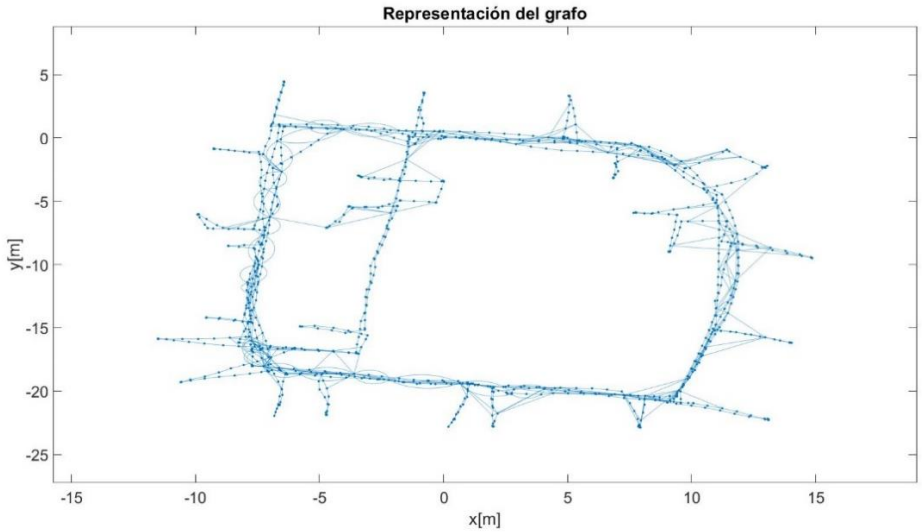

Figura 8. Representación del grafo de Intel

Como se puede observar, se trata de grafos complejos con un elevado número de nodos y arcos. En la Tabla 1 se muestra para cada grafo la información relativa al número de nodos y arcos, la longitud de la trayectoria por la que ha pasado el robot, el número de nodos por metro, así como el número de arcos por nodo, que se cuantifica mediante la variable $d=2m/n$, siendo m y n el número de arcos y de nodos del grafo respectivamente.

	Bicocca	**New College**	**Intel**
Número de nodos	8358	12816	1228
Número de arcos	8513	13171	1504
Longitud[m]	750,6	2418,8	497,7
Número de nodos por metro	11,4	5,3	2,5
d	2,04	2,06	2,45

Tabla 1. Información de los distintos grafos

La tabla anterior muestra como el grafo del entorno de *New College* tiene un mayor número de vértices con respecto a los otros dos grafos, sobre todo respecto al de *Intel*. Esta diferencia se debe a que el tamaño recorrido en cada entorno no es el mismo. Un grafo con un mayor número de nodos por metro indica que la trayectoria del robot es más compleja, realizando así más mediciones. Esto se puede observar en el entorno de *Bicocca* donde a pesar de que la longitud de la trayectoria es inferior a la de *New College*, el número de nodos por metro es superior, lo que indica que la trayectoria es más compleja (Figura 6). En cuanto al número de arcos por nodo, es el entorno de *Intel* el que tiene un mayor número, lo cual indica que la reconstrucción del entorno es más precisa puesto que el número de arcos de re-observación es mayor en proporción. Sin embargo, en los entornos de *Bicocca* y *New College* se puede ver como este número es muy próximo a dos, lo que hace indicar que la mayor parte de los arcos que forman parte del grafo son de odometría, esto es, cada vértice está unido únicamente con los vértices inmediatamente anterior y siguiente.

Con el objetivo de visualizar los arcos de re-observación (son los que hacen que el número de nodos y arcos sea distinto) para cada grafo de la tabla anterior y conocer su disposición en el grafo, en la Figura 9 se han representado de manera circular los grafos de *Bicocca*, *Intel* y *New College*, sin tener en cuenta la información espacial de los nodos, únicamente su numeración. En este tipo de representación los nodos de cada grafo (representados en color rojo) se disponen a lo largo del perímetro de una circunferencia, de modo que estos quedan unidos por los arcos de

odometría. Por otro lado, los arcos asociados a re-observaciones (representados en color azul) unen nodos no consecutivos de modo que quedan inscritos en la circunferencia, por lo que a la vista de la siguiente imagen (y como se ve en la última fila de la Tabla 1) se puede comprobar que el entorno de *Intel* tiene un mayor número de arcos de este tipo en proporción con el tamaño del grafo. Además, también se puede ver como en *Bicocca* y *New* College los arcos de observación están concentrados en zonas muy localizadas frente al de *Intel*, lo cual tiene sentido puesto que si se observa la forma de cada entorno se puede comprobar como *New College* y sobre todo *Bicocca* están definidos en su mayoría por trayectorias largas y rectas, por lo que cuando el robot pase por estas zonas, estas vendrán definidas por arcos de odometría.

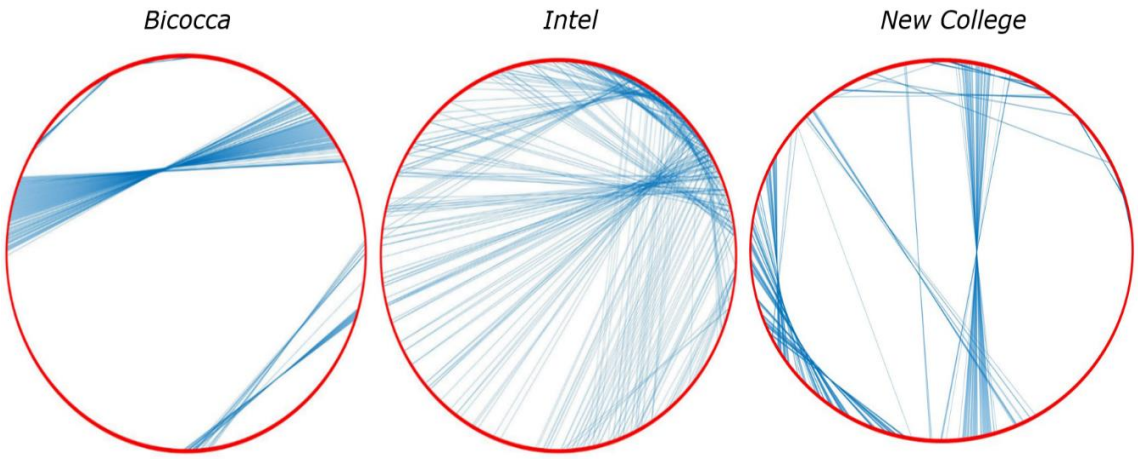

Figura 9. Representación de los grafos de Bicocca, Intel y New College

3.2. Camino con menor número de nodos

En este apartado, se pretende resolver el problema de búsqueda del camino más corto entre un par de nodos teniendo en cuenta únicamente el criterio de que el camino este formado por el mínimo número de vértices posible. Para ello se dispone de una función en Matlab que permite encontrar una ruta según la premisa anterior, y obtener el número de nodos que la forman.

Esta función requiere como argumentos de entrada un grafo previamente construido, los nodos entre los que se desea buscar la trayectoria más corta, así como el método utilizado, que en este caso se basa en el algoritmo *Breadth First search* (BFS), muy útil para la búsqueda de rutas que estén formadas por el mínimo número de nodos entre un par de vértices. Se trata de un algoritmo de búsqueda sin información que no tiene en cuenta la topología del grafo ni por tanto la trayectoria del robot, sino únicamente el número de nodos, para lo cual analiza secuencialmente todos los nodos que forman parte del grafo empezando desde un vértice de origen. El algoritmo funciona de la siguiente manera: en primer lugar, se parte desde un vértice de origen sobre el que se exploran todos sus nodos vecinos, posteriormente se exploran los vecinos de cada uno de estos, y se repite el proceso hasta alcanzar el nodo final. Si se encuentran varias rutas posibles, el algoritmo escoge la que este formada por el mínimo número de nodos. En el Anexo III se dispone de información adicional sobre el algoritmo BFS.

Para entender mejor esta función, en la Figura 10 se muestra un ejemplo sencillo de búsqueda de la trayectoria con menor número de vértices entre dos nodos en un grafo con diez nodos y trece arcos. Suponiendo que se desee ir desde el nodo uno hasta el cuatro, la trayectoria con menor número de nodos sería 1-6-4 (marcada en color rojo).

Si se utiliza esta misma función en grafos donde el número de nodos es muy elevado el procedimiento sería el mismo. En la Figura 11 se muestra un ejemplo de búsqueda de la ruta más corta entre dos nodos seleccionados aleatoriamente en el entorno de *Intel*. La trayectoria resultante, formada por 41 nodos, se muestra en color rojo.

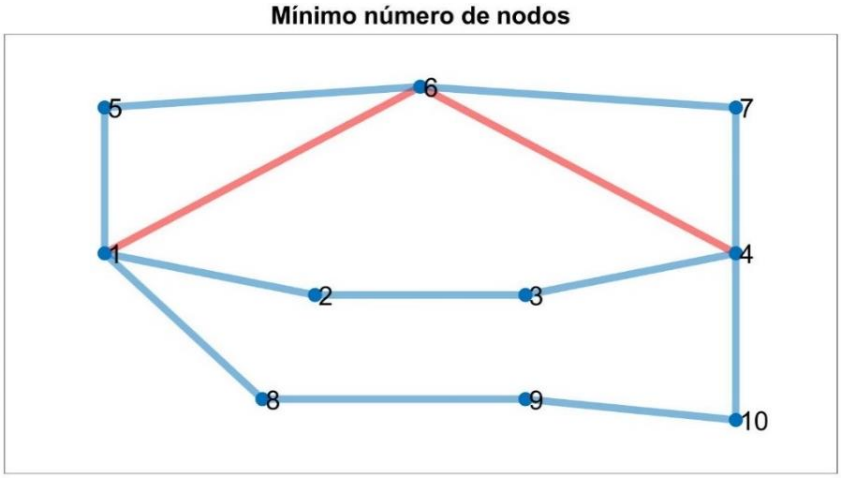

Figura 10. Ejemplo búsqueda del camino con menor número de nodos en grafo sencillo

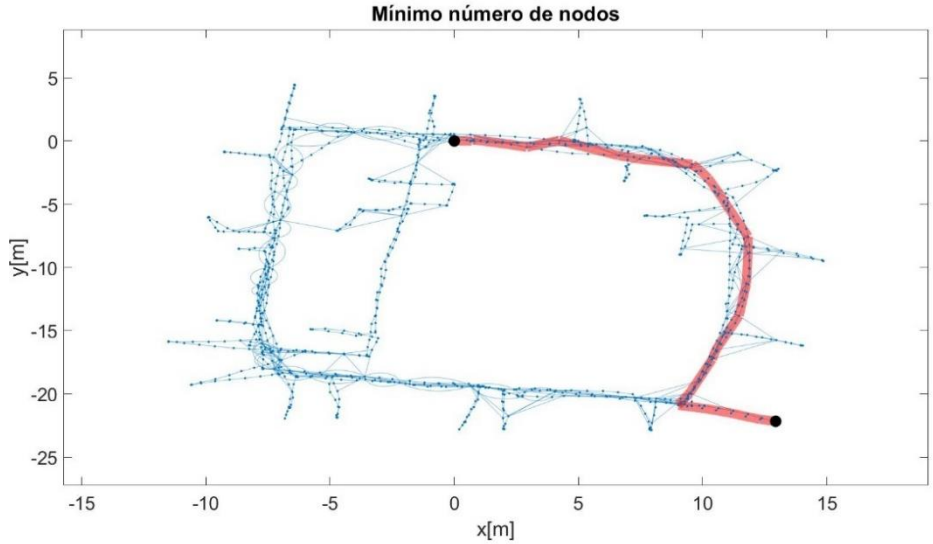

Figura 11. Ejemplo de búsqueda del camino con menor número de nodos en Intel

3.3. Camino más corto

La búsqueda de una trayectoria según la premisa de que este formada por el menor número de nodos, no siempre implica que la distancia euclídea sea mínima entre los nodos de inicio y fin. Esta diferencia depende del número de vértices y arcos del grafo, así como de la disposición espacial de los nodos.

La solución a esta limitación es usar la mínima distancia euclídea como criterio a la hora de buscar la ruta más corta entre dos nodos. Para ello, se han utilizado dos funciones, ambas basadas en el algoritmo de Dijkstra. Este algoritmo trata de buscar todos los caminos más cortos desde un vértice de origen hasta el resto de los vértices que forman el grafo, para lo cual va evaluando secuencialmente los nodos vecinos de entre los cuales va escogiendo aquellos con una menor distancia euclídea. Una vez obtenidos todos los caminos más cortos el algoritmo se detiene.

En una de estas funciones, se aplica el algoritmo de Dijkstra sobre las matrices que contienen la información de los nodos y los arcos que forman el grafo, mientras que la otra función aplica el algoritmo de Dijkstra directamente sobre el objeto grafo, pesado con las distancias euclídeas en los arcos.

En el Anexo IV se dispone de información adicional sobre el algoritmo de Dijkstra.

3.3.1. Naive Dijkstra

La primera función con la que se va a trabajar requiere como argumentos de entrada la información de los nodos y los arcos que forman el grafo, así como de los nodos de inicio y fin a partir de los cuales se pretende buscar el camino más corto.

Por tanto, dicha función no requiere de un grafo previamente definido. La información de los nodos se importa a través de una matriz que debe recoger la información de la numeración de los nodos y de las coordenadas de cada uno de ellos en todas las dimensiones. Por otro lado, los arcos se deben importar a través de una matriz con tres columnas que representen la numeración de cada arco, y los nodos de inicio y de fin que forman el arco. Esta función opera según el algoritmo de Dijkstra, tal y como se explica en el Anexo IV.

En la Figura 12 se muestra un ejemplo de búsqueda del camino más corto para el mismo grafo del Apartado 3.2:

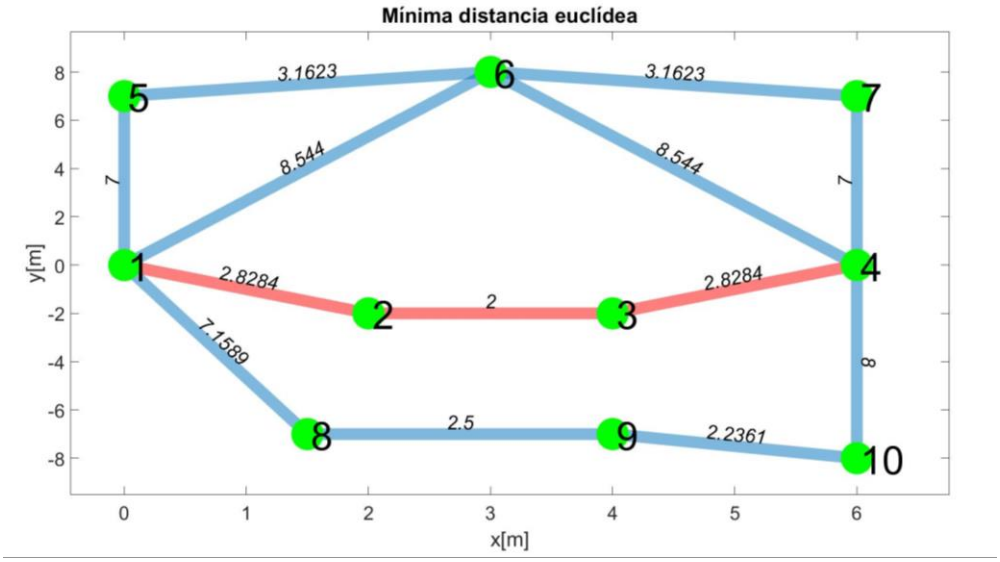

Figura 12. Ejemplo de búsqueda según Naive Dijkstra en grafo sencillo

Como se ha comentado, aunque esta función no trabaja sobre un grafo previamente ponderado si no con las respectivas matrices de información de nodos y arcos, para una mayor claridad se han añadido sobre el grafo las distancias euclídeas como pesos en los arcos del grafo.

En la siguiente tabla se puede observar como a diferencia del apartado anterior, en este caso es la opción 3 la escogida como ruta más corta puesto que, pese a que el número de nodos que forman la trayectoria es superior al de la opción 2, la distancia euclídea resultante es inferior.

	Trayectorias	**Distancia euclídea (metros)**
Opción 1	1-5-6-7-4	20.3
Opción 2	1-6-4	17.1
Opción 3	1-2-3-4	7.7
Opción 4	1-8-9-10-4	19.9

Tabla 2. Posibles rutas para el ejemplo de la figura 12

Aplicando esta misma función al entorno de *New College*, se puede obtener el camino más corto entre un par de nodos aleatorios:

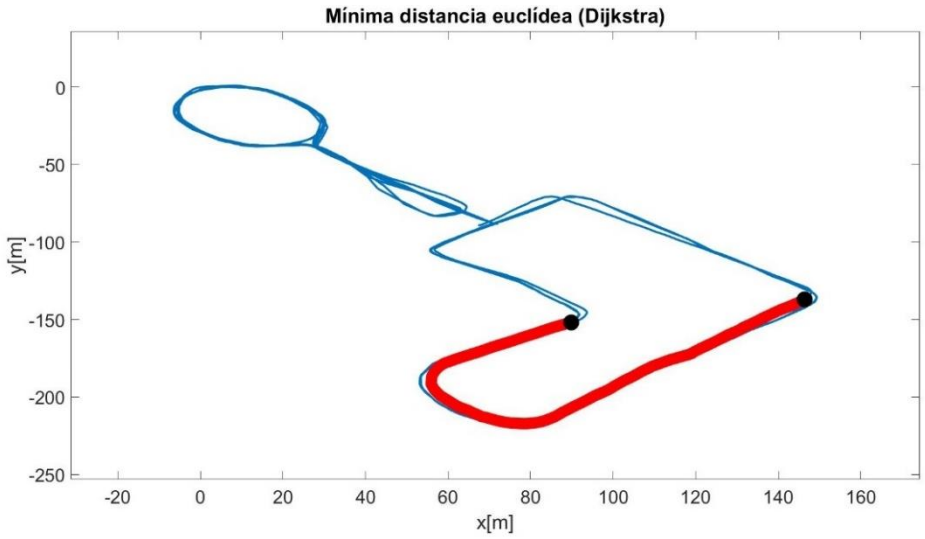

Figura 13. Ejemplo de búsqueda según Naive Dijkstra en New College

3.3.2. Grafos pesados

En esta sección se ha utilizado la función *shortestpath* de la toolbox *Graph and Network Algorithms* que dispone Matlab, basada en el algoritmo de Dijkstra para encontrar el camino más corto entre dos nodos con la mínima distancia euclídea, y a su vez poder compararla con la función anterior.

Se trata de una función parecida a la que trata de buscar el mínimo número de nodos, y a su vez basada en el algoritmo que utiliza Naive Dijkstra. La principal diferencia con respecto a las anteriores reside en que en este caso el grafo se ha ponderado con las distancias euclídeas en los arcos, de forma que la función busca la trayectoria más corta entre los puntos de inicio y fin teniendo en cuenta el peso de cada uno de los arcos que conforman el grafo. Por tanto, para poder hacer uso de esta función se requiere construir previamente un objeto de tipo grafo ponderado, en el que cada arco este pesado con la distancia euclídea que separa los dos nodos que une dicho arco. Los pesos en los arcos deben ser escalares reales positivos.

Si se buscara la ruta más corta entre el nodo 1 y 4 (como en el apartado anterior) según esta última función, es fácil darse cuenta como el resultado obtenido sería el mismo que el de la Figura 12, lo cual tiene sentido puesto que se está utilizando el mismo algoritmo.

Aplicando esta misma función entre dos nodos aleatorios al entorno de Bicocca se obtiene la trayectoria marcada en color rojo de la Figura 14, la cual está formada por 77 metros.

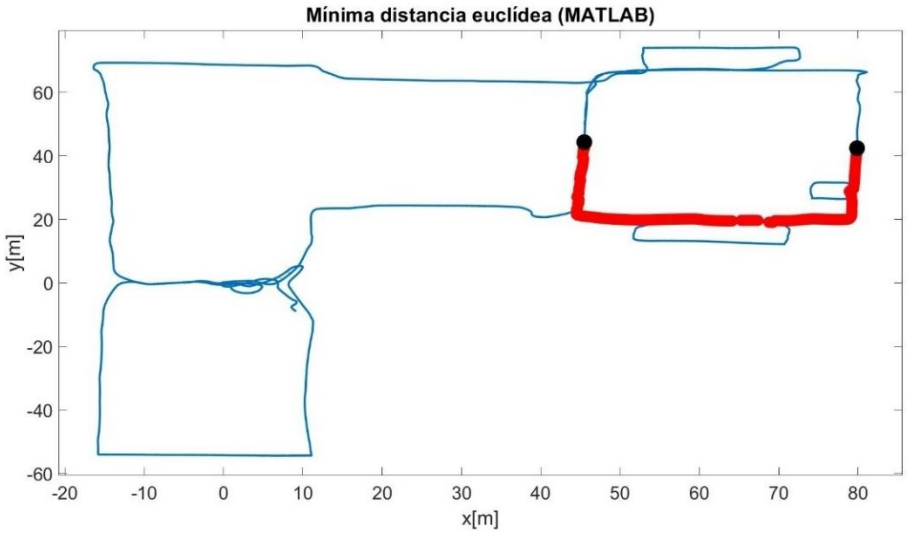

Figura 14. Ejemplo de búsqueda de camino más corto en Bicocca

En la Tabla 3 se muestran los resultados de comparar 300 veces en los entornos de *Bicocca*, *New College* e *Intel* para diferentes pares de nodos, las rutas obtenidas según los dos métodos que utilizan como premisa la mínima distancia euclídea. Puesto que la mayoría de las veces que las rutas obtenidas según la función que aplica Dijkstra sobre el objeto de tipo grafo pesado tienen una distancia euclídea inferior, y el tiempo de cómputo necesario también es más pequeño en comparación con el segundo método, de ahora en adelante cuando se haga referencia al método de búsqueda de la trayectoria más corta se entenderá que la función que se ha utilizado es la que aplica Dijkstra sobre el grafo pesado.

	Bicocca	***New College***	***Intel***
Porcentaje de veces que la función que aplica Dijkstra sobre el grafo pesado obtiene una trayectoria más corta	85%	81%	75%
Tiempo de cómputo medio[s] (Dijkstra sobre grafo pesado)	0,00085	0,001	0,00036
Tiempo de cómputo medio[s] (Dijkstra)	0,517	0,884	0,034

Tabla 3. Comparación entre los dos métodos que permiten buscar el camino más corto

3.4. Camino con menor incertidumbre

En este capítulo se va a trabajar con el algoritmo de FAMUS (*Fast Minimum Uncertainty Search*) [1], el cual fue presentado en 2012 en la conferencia internacional *Intelligent Robots and Systems*. Se trata de un algoritmo que pretende resolver el problema de planificación de rutas sobre una representación reducida del entorno sobre el que se está trabajando, utilizando como criterio la incertidumbre. Su aplicación se puede dividir en cinco fases: (a) en primer lugar se cuantifica la incertidumbre asociada a cada arco que forma parte del grafo completo calculando el criterio D-opt de la matriz de covarianza correspondiente, (b) posteriormente se realiza una búsqueda de vecinos cercanos sobre cada uno de los nodos con el objetivo de crear arcos de re-observación entre trayectorias muy próximas que no estuvieran unidas entre sí, (c) después se aplica una reducción al grafo y (d) se calcula sobre este el camino con mínima incertidumbre. Finalmente (e) se reconstruye la trayectoria sobre el grafo completo.

La utilización de FAMUS resulta muy interesante para paliar el principal inconveniente del método de planificación de rutas según el camino más corto, que no garantiza obtener la trayectoria con la menor incertidumbre. FAMUS trata de priorizar caminos con arcos con menor incertidumbre en el proceso de planificación de rutas, con el objetivo de conseguir incertidumbres acumuladas bajas, aunque la ruta resultante sea más larga. Además, permite una reducción en el tiempo de cómputo de búsqueda de trayectorias, sobre todo en grafos en los que el número de vértices y de arcos es elevado, ya que esta planificación la realiza sobre el grafo equivalente reducido.

Relacionándolo con el SLAM activo, la utilización de FAMUS puede resultar muy útil ya que cuando el robot tiene que elegir la siguiente zona a explorar, si se dispone del grafo reducido, este algoritmo puede realizar rápidamente la búsqueda de la trayectoria con la menor incertidumbre que le ayude al robot a optimizar la ejecución del algoritmo de SLAM. En este trabajo se plantea el problema de planificación de rutas desde un punto de vista parecido al del robot cuando ejecuta SLAM activo. Partiendo desde un nodo de origen, en lugar de evaluar sobre un grafo parcialmente conocido las distintas trayectorias que le permitan al robot alcanzar un destino que pueda ser diferente (SLAM activo), lo que se va a hacer es evaluar sobre un grafo conocido las diferentes rutas que le permitan al robot alcanzar un vértice de destino fijo.

Por tanto, inicialmente se debe ponderar el grafo con un criterio de optimalidad (inicialmente se considerará el criterio D-opt) con el objetivo de recoger la

incertidumbre asociada a cada arco. Como se recordará del Apartado 2.2.1, estos criterios son muy útiles para representar la matriz de información o covarianza en un único número escalar puesto que resulta más fácil trabajar con grafos pesados con números escalares en los arcos en vez de con matrices de dimensión tres o seis.

3.4.1. Link Prediction

Una vez que se ha pesado el grafo calculando el criterio D-opt sobre la matriz de covarianza de cada arco, sobre el grafo inicial construido a partir de la información de los nodos y de los arcos, se realiza lo que se denomina búsqueda de nodos cercanos, una estrategia de *link prediction*. En este punto, lo que se pretende es para cada uno de los nodos que forman el grafo, realizar una búsqueda de todos aquellos nodos cuya separación sea inferior a una distancia umbral, de modo que si no están conectados entre sí se realice una unión entre ellos mediante la creación de un nuevo arco. De esta manera, lo que se consigue es unir dos trayectorias distintas muy próximas entre sí que previamente no estaban conectadas debido posiblemente a que el sistema de SLAM fuera incapaz de encontrar un arco de cerrado de bucle entre ambas. Es decir, si el robot no detecta que es la misma marca la que se ha observado desde dos poses distintas, este no será capaz de crear un arco entre ambos nodos y por lo tanto las dos trayectorias (muy próximas entre sí) quedarán desconectadas. Cabe destacar que, para que este algoritmo no cree arcos entre trayectorias relativamente cercanas separadas por un obstáculo físico (como pudiera ser un muro), la distancia umbral debe ser muy restrictiva. En este caso se ha elegido una distancia umbral de cinco centímetros.

Con el objetivo de evitar crear nuevos arcos entre nodos muy cercanos que están conectados indirectamente a través de arcos de odometría consecutivos y que por tanto forman parte de la misma trayectoria, se ha creado una lista con todos los nodos vecinos cercanos que preceden y suceden consecutivamente al nodo sobre el que se está haciendo la búsqueda. De este modo si el vecino candidato a unirse al nodo sobre el que se ha realizado la búsqueda pertenece a dicha lista no se crea un nuevo arco. En la Sección A del Anexo V se muestra el pseudocódigo que se ha utilizado para aplicar la estrategia de *link prediction*.

En la Figura 15 se muestra un ejemplo de aplicación de la lista en una zona muy localizada del entorno de *Bicocca*. Como se puede observar, todos los nodos que aparecen en la imagen están separados entre sí por una distancia inferior a la distancia umbral previamente definida, sin embargo, están unidos consecutivamente a través de arcos de odometría. De este modo, al utilizar una lista de nodos prohibidos se evita que se creen un número desmesurado de nuevos arcos en un

espacio muy pequeño entre los distintos nodos que están comprendidos entre los dos vértices marcados con color rojo.

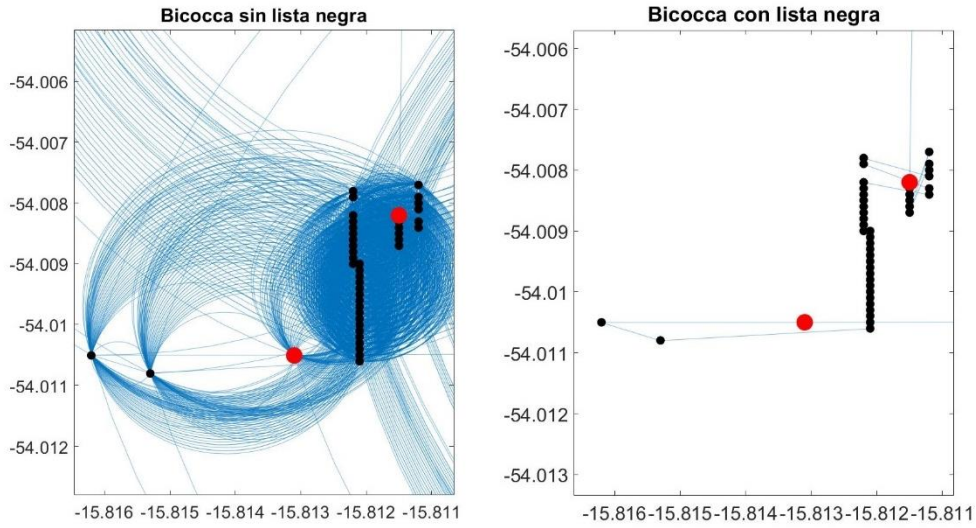

Figura 15. Ejemplo de aplicación sin usar la lista negra (izquierda) y usándola (derecha)

Para los diferentes entornos que se han utilizado en este trabajo se han creado los arcos que se muestran en la Tabla 4 tras aplicar la función comentada. Como se puede observar el número de arcos nuevos que se han creado no es muy elevado debido a la distancia umbral tan pequeña que se ha tenido en cuenta para la búsqueda de los nodos cercanos.

	Número de arcos del grafo original	**Número de arcos del grafo modificado**
Intel	1504	1536
Bicocca	8513	8832
New College	13171	13763

Tabla 4. Comparación del número de arcos antes y después de aplicar la estrategia de link prediction

En la Figura 16 se muestra una comparativa de dos trayectorias que pasan muy cerca una de la otra para una zona muy localizada del entorno de *Bicocca*. Esta imagen comparativa muestra como la función de creación de arcos une los nodos de las dos trayectorias que están por debajo de la distancia umbral.

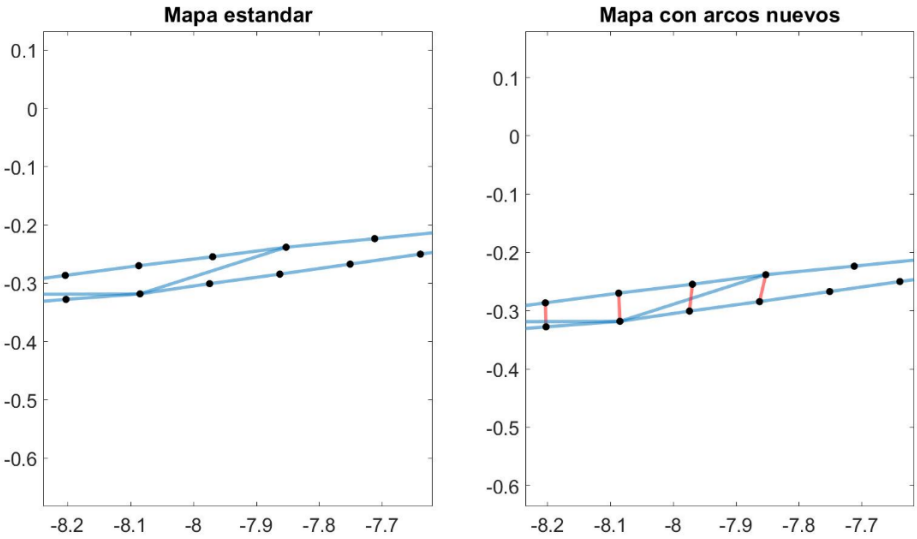

Figura 16. Comparación en una zona muy concreta del grafo de Bicocca antes (izquierda) y después (derecha) de aplicar la estrategia de link prediction

3.4.2. Reducción de grafos

En este apartado se va a llevar a cabo la tercera fase del algoritmo de FAMUS con el objetivo de reducir el número de nodos y arcos del grafo original. Para ello, se pretende unificar la información de nodos consecutivos de grado dos en un único nodo que recoja la información de todos ellos.

Esta reducción se va a aplicar sobre grafos basados en poses, cuya formulación resulta muy útil en el proceso de planificación de trayectorias, puesto que el robot ya ha pasado por las diferentes rutas potencialmente transitables durante el proceso de mapeo. Como se recordará, estos grafos están formados por un único tipo de nodos (posiciones del robot) y dos tipos de arcos, de odometría y de cerrado de bucle.

Para facilitar la tarea de reducción del grafo se van a clasificar los nodos según su grado. Los nodos con grado inferior a tres son todos aquellos que están unidos únicamente por arcos de odometría y por tanto, a la hora de planificar una ruta, no permiten elegir cual es el siguiente nodo por el que continuar la trayectoria. Visto desde el punto de vista del robot, cuando este avanza en la dirección de un arco que conecta con un nodo de grado dos, el robot únicamente puede continuar en la dirección del arco que le conecta con el nodo que le sucede.

Por otro lado, los nodos de decisión son todos aquellos que tienen un grado superior a dos, i.e. están conectados con más de dos arcos. Estos nodos se llaman de decisión porque cuando el robot está situado en uno de estos, puede elegir a cuál de los siguientes nodos (de entre los que están conectados) quiere avanzar puesto que está conectado a más de un nodo sucesor. Es decir, suponiendo que en una parte del grafo hay 20 nodos unidos secuencialmente por arcos de odometría (cada uno de

ellos con sus respectivos pesos w_i), el objetivo será sustituir todos estos nodos por dos vértices unidos por un arco que recoja el peso acumulado de todos los arcos que han sido eliminados $w_n = \Sigma w_i$. De esta manera se consigue construir un grafo que únicamente esté formado por nodos de decisión y los respectivos pesos asociados de los arcos que los unen.

En la Sección B del Anexo V se muestra el pseudocódigo utilizado para realizar la reducción del grafo.

Sabiendo como está construido un *pose graph*, es fácil darse cuenta de que la mayoría de los arcos que lo constituyen son de odometría. Por tanto, si se aplicase una reducción, se perdería la topología del grafo. Poniendo como ejemplo el entorno de *Bicocca*, tal y como se muestra en la Figura 17, si se aplica la reducción al grafo completo, la información espacial del entorno se pierde completamente, de modo que varios de los arcos de la imagen podrían estar atravesando obstáculos físicos. En esta imagen los nodos se han representado en color negro y los arcos en azul.

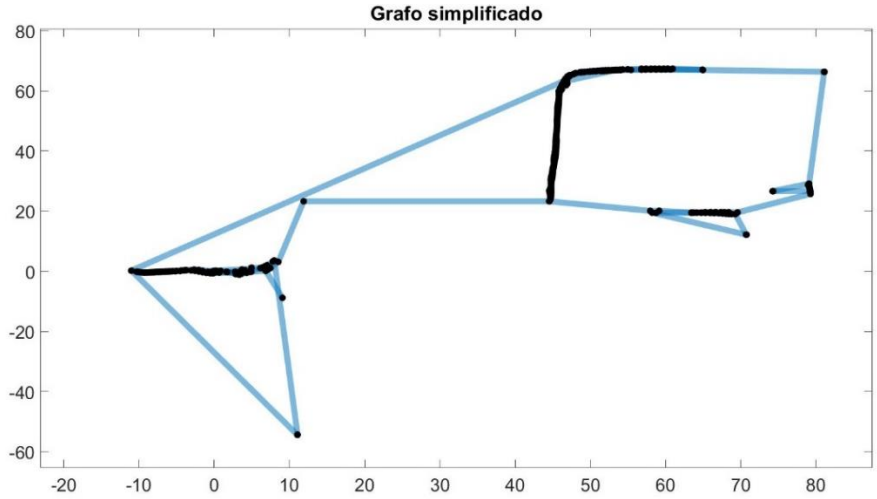

Figura 17. Reducción del grafo de Bicocca

Para evitar este problema, se ha evaluado para todos los nodos de grado dos del grafo el sumatorio de los ángulos que codifican la orientación relativa de cada nodo para una ventana de 50 nodos. Si este valor es superior a un valor límite (en este caso 90 grados), en el proceso de reducción del grafo no se borra el ultimo nodo de la ventana de 50 vértices. Si por el contrario el sumatorio de los ángulos es inferior al valor límite, se borra la información del primer nodo de la ventana y se almacena la información del siguiente nodo. De esta manera, se consigue que en el proceso de reducción no se borren determinados nodos (principalmente los que representan un cambio en la trayectoria del robot) aunque sean de grado dos, permitiendo mantener la forma original del grafo.

Por tanto, en la lista negra se añadirán todos aquellos nodos que no se quieran borrar, como los nodos de inicio y fin que forman parte de la trayectoria que se desea encontrar, y aquellos nodos que contienen información sobre la topología del grafo. En la Sección B del Anexo V se adjunta el pseudocódigo utilizado para mantener la forma original del grafo.

Aplicando los cambios que se acaban de mencionar se obtiene el grafo reducido de la Figura 18 para el entorno de *Bicocca*. Como se puede observar, ahora la topografía del grafo se ha conservado. Aquellas zonas en la que la densidad de vértices es mayor, habrá una mayor cantidad de nodos de decisión.

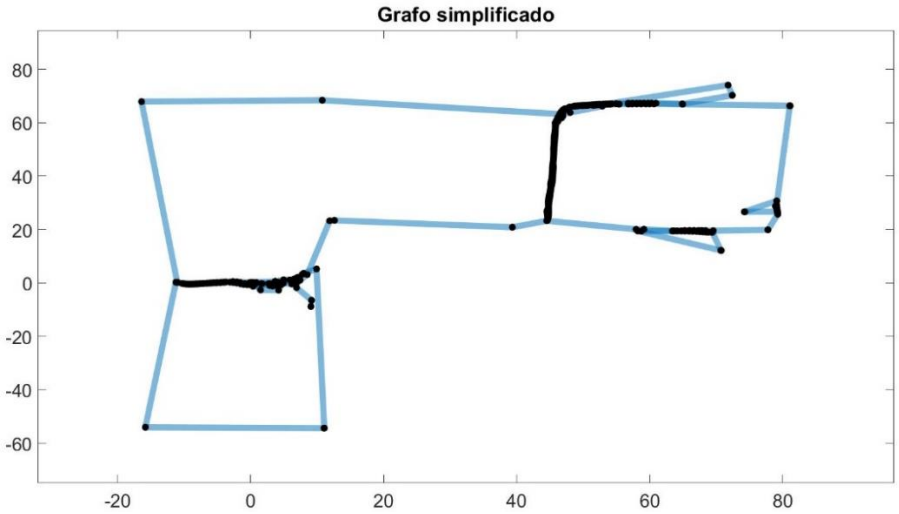

Figura 18. Reducción del grafo de Bicocca aplicando la lista negra

Dependiendo del tipo de grafo sobre el que se esté trabajando, habrá una mayor o menor reducción del número de nodos y arcos. En entornos en los que el número de nodos de decisión no es muy elevado (como el de la imagen anterior), debido por ejemplo a que el robot pasa por largas trayectorias rectas, la reducción del grafo será mayor puesto que estas partes del grafo podrán ser sustituidas por un reducido número de nodos.

Los resultados de aplicar la función de reducción de grafos se presentan en la Tabla 5, en la que se muestra la información correspondiente al número de nodos, arcos, al porcentaje de reducción de ambos, así como al número de nodos de decisión y de nodos no eliminados debido a que forman parte de una ventana de nodos cuyo ángulo acumulado es superior a 90 grados.

A la vista de los resultados, se puede comprobar que en el *dataset Bicocca* se obtiene una mayor reducción de nodos y arcos debido a que tiene un mayor porcentaje de arcos de odometría (85,56%) frente al resto. Es por esta misma razón, por la que en *New College* (80,13%) y sobre todo en *Intel* (66,15%) la reducción es inferior.

	Intel	Bicocca	New College
Nodos (Antes/después)	1228/212	8358/801	12816/1788
Arcos (Antes/después)	1536/520	8832/1275	13763/2735
Reducción de nodos	82,74%	90,42%	86,05%
Reducción de arcos	66,15%	85,56%	80,13%
Nodos de decisión	187	732	1631
Nodos con ángulo >90º	3	28	25

Tabla 5. Resultados de aplicar la reducción a Intel, Bicocca y New College

En el Anexo VI se muestran los grafos de *Intel*, *Bicocca* y *New College* antes y después de aplicar la reducción.

Una vez calculada la ruta sobre el grafo reducido hay que extrapolarla al grafo completo, ya que es este último el que realmente representa el entorno y sobre el cual se representa la trayectoria que realiza el robot durante las tareas de exploración. Para ello se puede utilizar una tabla que relacione los nodos entre los dos grafos.

3.4.3. Búsqueda del camino óptimo

El propósito de este apartado es describir la aplicación de la función que permite buscar la trayectoria con la menor incertidumbre sobre el grafo simplificado, para posteriormente extrapolarla al grafo completo y verificar que se trata de una trayectoria con un significado físico para el robot.

Para la búsqueda del camino con la mínima incertidumbre se va a utilizar una función que se basa en el algoritmo de Dijkstra (Anexo IV), análogo al que se utilizó para encontrar el camino con menor distancia euclídea. La principal diferencia proviene de que en este caso los arcos del grafo se ponderan con el criterio D-opt que caracteriza la matriz de covarianza de cada arco.

Si se aplica la función de búsqueda del camino con mínima incertidumbre entre un par aleatorio de nodos sobre el grafo reducido de *New College* se obtiene la siguiente trayectoria:

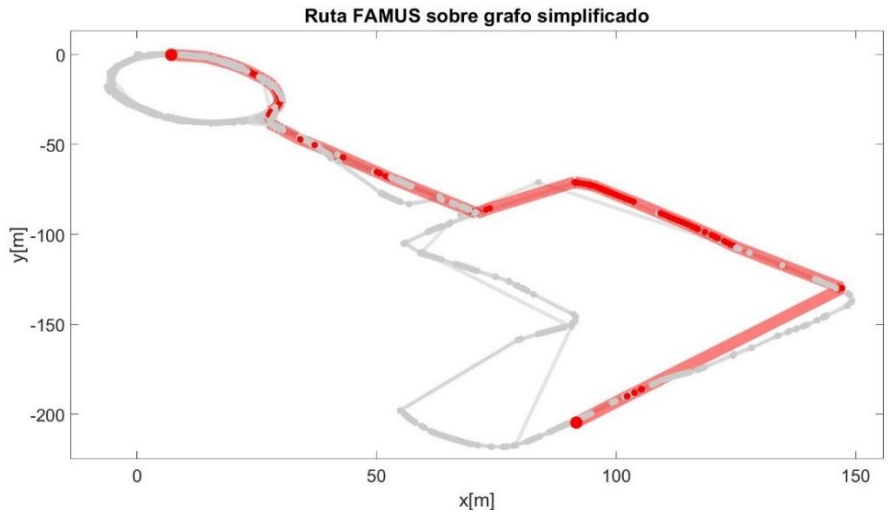

Figura 19. Aplicación de FAMUS en New College

Una vez calculada la trayectoria sobre el grafo reducido, se extrapola al grafo completo. Para ello se hace uso de la tabla de equivalencias entre nodos del grafo completo y del simplificado. Puesto que los únicos nodos que se han eliminado en la reducción son los que forman parte de arcos de odometría, para obtener la trayectoria extrapolada completa es necesario unir aquellos nodos que no estén unidos entre sí mediante los arcos de odometría que los separan.

En la Figura 20 se muestra la trayectoria previamente calculada extrapolada al grafo completo:

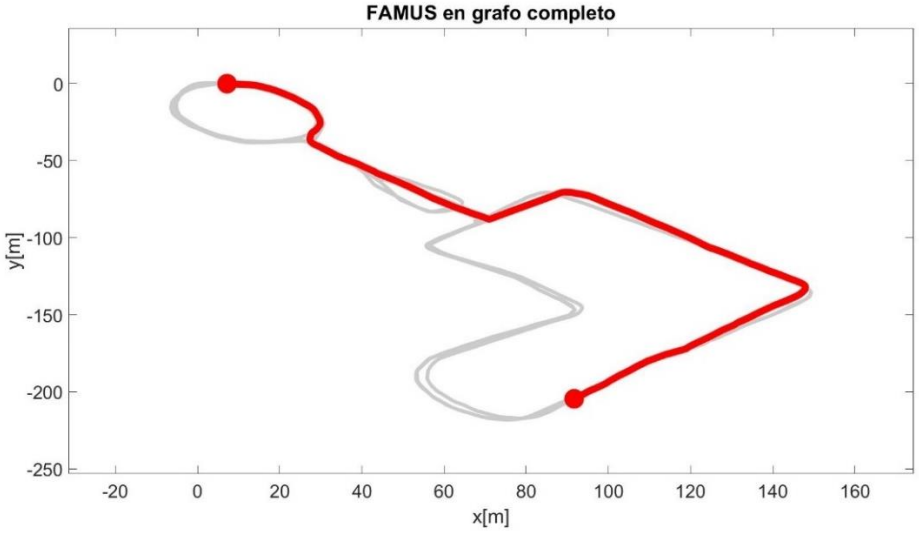

Figura 20. Trayectoria de grafo simplificado extrapolada a grafo completo

En la Tabla 6 se muestra información adicional sobre la trayectoria obtenida en el grafo simplificado, así como de la trayectoria extrapolada al grafo completo. Calculando el sumatorio del peso de los arcos de las dos trayectorias, o lo que es lo mismo, la incertidumbre acumulada (en este caso a partir del criterio D-opt), se

puede verificar como ambos valores son iguales y por tanto la extrapolación se ha realizado correctamente. Del mismo modo, también se ha podido verificar como la trayectoria y la incertidumbre acumulada obtenidas son las mismas si se realiza la búsqueda directamente sobre el grafo completo.

A la vista de los resultados se puede verificar la utilidad de esta herramienta de búsqueda de trayectorias sobre grafos simplificados con su correspondiente minimización del tiempo de cómputo, muy útil sobre todo en grafos en los que el número de nodos es muy elevado.

	Nodo de inicio	**Nodo de fin**	**Número de nodos de la trayectoria**	**Incertidumbre acumulada**
Grafo reducido	144	1776	184	0.7046
Grafo completo	617	11860	1519	0.7046

Tabla 6. Información de las trayectorias de las figuras 19 y 20

4. Resultados

Como ya se ha visto, en el proceso de planificación de rutas existen diferentes métodos que tratan de encontrar la trayectoria óptima según distintas premisas. En este capítulo se van a comparar las trayectorias obtenidas de trabajar con los distintos métodos vistos en el Capítulo 3 así como con los criterios de optimalidad presentados en el Apartado 2.2.1. El capítulo está estructurado como sigue:

- En primer lugar, se analizan las trayectorias obtenidas según los distintos métodos y se comparan entre ellas, haciendo hincapié en la incertidumbre acumulada de cada una de ellas (Sección 4.1).
- Posteriormente se estudian las diferencias entre los tiempos necesarios que necesita cada método para calcular la trayectoria óptima. En esta sección se pretende demostrar que FAMUS es capaz de encontrar rápidamente la ruta de mínima incertidumbre cuando se dispone de una representación reducida del entorno. A su vez, se presentan dos aplicaciones de planificación de rutas cuya eficiencia se mejora con la utilización del algoritmo de FAMUS (Sección 4.2).
- Finalmente, se estudian los distintos criterios de optimalidad que permiten caracterizar la incertidumbre de la matriz de covarianza, con el objetivo de demostrar que el criterio D-opt es el que permite capturar de la manera más acertada la matriz de incertidumbre (Sección 4.3).

4.1. Comparación entre métodos

Una vez definidos los distintos métodos utilizados para la búsqueda y selección de trayectorias óptimas entre dos nodos, en esta sección se van a comparar los algoritmos previamente explicados para la planificación de rutas en los distintos entornos que se han utilizado en la memoria.

En las imágenes siguientes se muestran tres ejemplos de búsqueda de la ruta óptima según los tres métodos ya explicados para diferentes pares de nodos aleatorios en los entornos de *Bicocca*, *Intel* y *New college*. En la parte izquierda de las figuras se representa el camino con el menor número de nodos, en el centro la ruta más corta según Dijkstra y en la derecha la ruta de mínima incertidumbre según FAMUS, todas entre el par de nodos marcados en color negro.

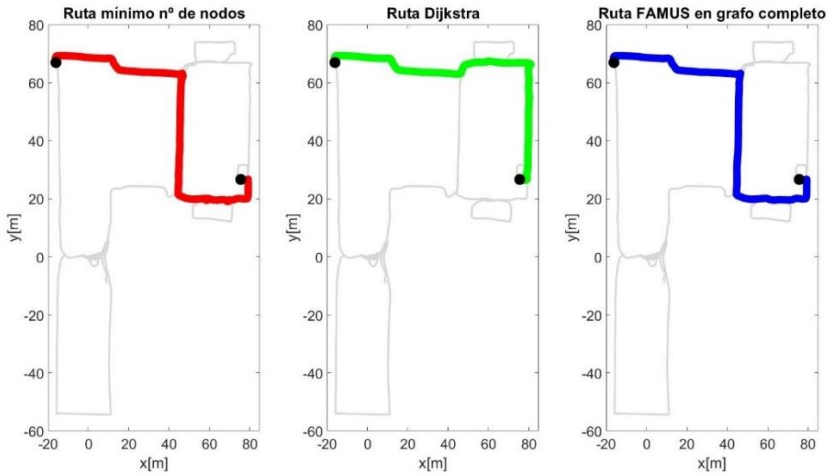

Figura 21. Comparación de los distintos métodos de búsqueda de trayectorias en Bicocca

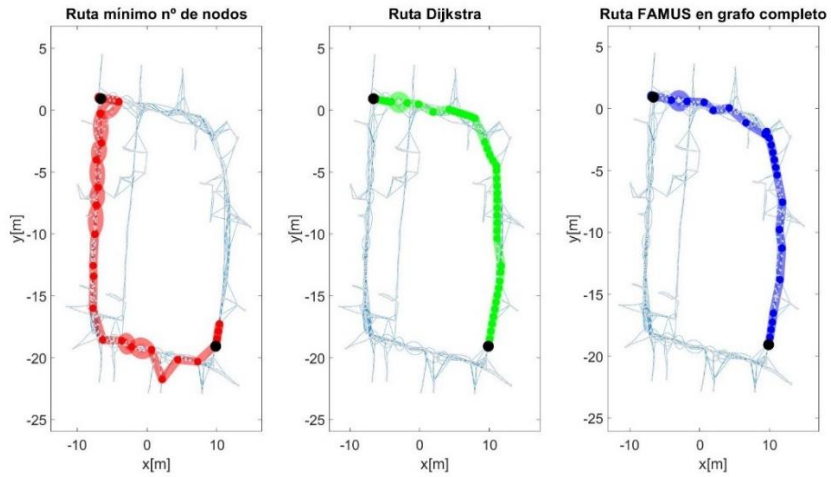

Figura 22. Comparación de los distintos métodos de búsqueda de trayectorias en Intel

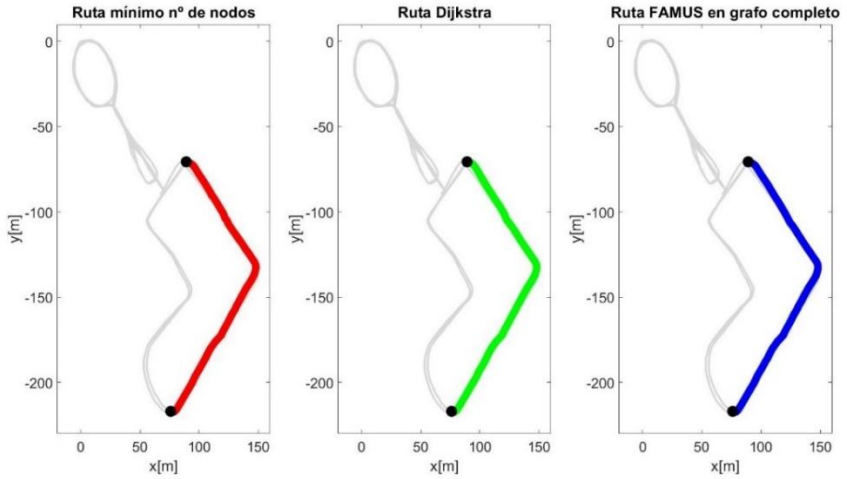

Figura 23. Comparación de los distintos métodos de búsqueda de trayectorias en New College

A la vista de las imágenes anteriores se puede comprobar como los tres métodos de búsqueda de trayectorias pueden dar resultados distintos, de modo que las rutas obtenidas según el criterio del menor número de nodos o la menor distancia euclídea, no implican que tengan menor incertidumbre. Como se puede ver, en las imágenes de *New College*, parecen coincidir las tres trayectorias, a diferencia de *Intel*, donde la ruta con la menor distancia es similar a la de menor incertidumbre, y de *Bicocca*, donde es la trayectoria con el menor número de nodos la que más se parece a la de menor incertidumbre.

En las figuras 24 a 26 se representa la incertidumbre acumulada según el criterio D-opt de las trayectorias anteriores obtenidas según las premisas del menor número de nodos (rojo), menor distancia euclídea según Dijkstra (verde) y mínima incertidumbre según FAMUS (azul). Puesto que, la exploración de entornos reales queda fuera del alcance de este trabajo, en la tarea de planificación de trayectorias se han utilizado grafos completos de entornos reales para estudiar este problema. Es por este motivo por el que las acumulaciones de incertidumbre de las trayectorias anteriores únicamente muestran un carácter creciente. En una situación real en la que el robot estuviese explorando un entorno, si el robot recorriese las trayectorias de las figuras anteriores, no solo percibiría los aumentos de incertidumbre debidos a la odometría, sino que tan pronto como este encontrase un arco de cerrado de bucle, podría re-localizarse con la consecuente disminución de la incertidumbre asociada a los nodos.

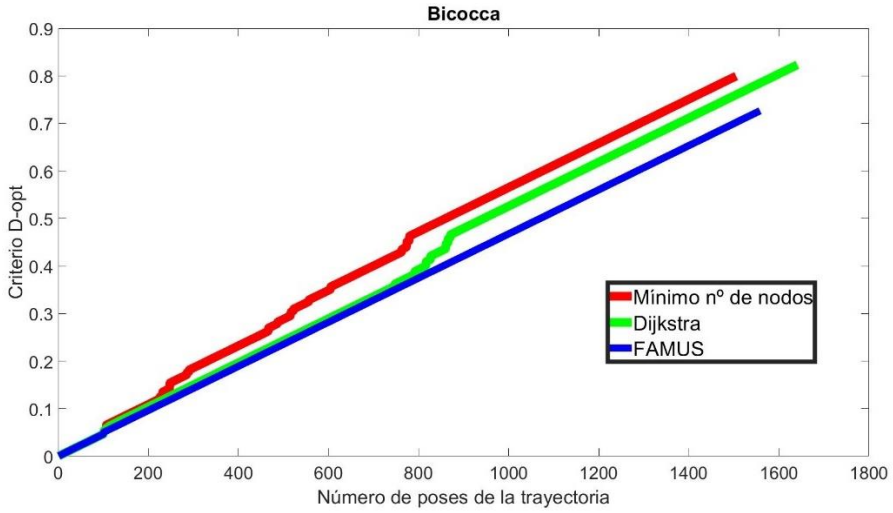

Figura 24. Evolución de la incertidumbre acumulada en Bicocca según el mínimo número de nodos (rojo), menor distancia euclídea (verde) y FAMUS (azul)

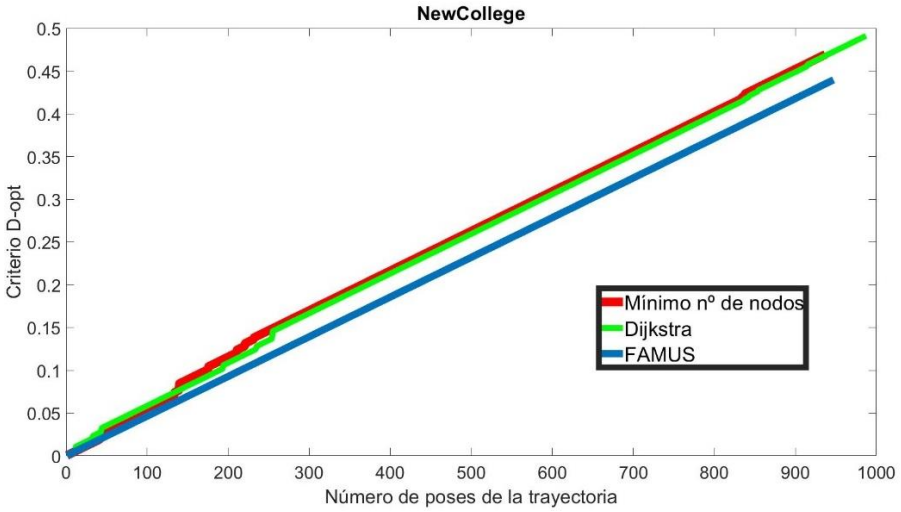

Figura 25. Evolución de la incertidumbre acumulada en New College según el mínimo número de nodos (rojo), menor distancia euclídea (verde) y FAMUS (azul)

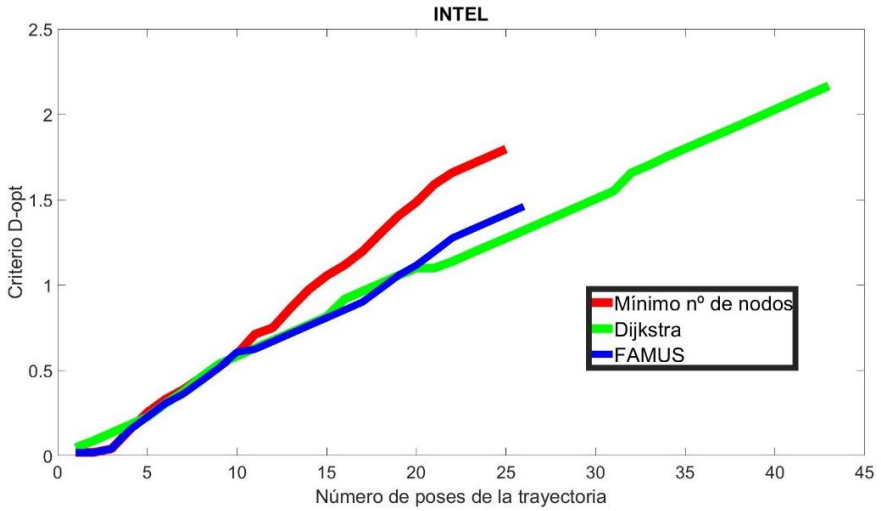

Figura 26. Evolución de la incertidumbre acumulada en Intel según el mínimo número de nodos (rojo), menor distancia euclídea (verde) y FAMUS (azul)

Como se puede observar, en todos los casos es el algoritmo de FAMUS (representado en azul) con el que se ha conseguido un valor de incertidumbre acumulada menor. Esto tiene sentido puesto que hay que recordar que trata de buscar el camino que minimiza el criterio D-opt (que caracteriza la matriz de covarianza) de cada arco.

Con el objetivo de extrapolar los resultados anteriores a un experimento más amplio estadísticamente relevante se han calculado 300 rutas según los distintos métodos entre pares de nodos aleatorios separados una distancia mínima del 5% y una distancia máxima del 60% sobre el total de nodos de los que está formado el grafo. Los resultados obtenidos se muestran en la Tabla 7, en la que se representan los porcentajes de veces que las trayectorias son idénticas, así como el número de vértices que comparten las distintas trayectorias mediante el porcentaje de solapamiento. Se puede comprobar que es el entorno de *Bicocca* en el que el porcentaje de nodos coincidentes entre distintas trayectorias es mayor, frente al de *Intel* que presenta el menor porcentaje de los tres. Esta diferencia se puede deber a que el número de arcos de odometría en proporción con los arcos que tiene cada grafo es mayor en los entornos de *Bicocca* y *New college* frente a *Intel*.

	Bicocca	***Intel***	***New College***
Número de vértices	8538	1228	12816
Porcentaje de rutas idénticas (FAMUS y mínimo número de nodos)	10%	13%	7,7%
Porcentaje de rutas idénticas (FAMUS y Dijkstra)	10%	11,3%	5,3%
Porcentaje de rutas idénticas (Mínimo número de nodos y Dijkstra)	12,3%	8%	5,7%
Porcentaje de solapamiento (FAMUS y mínimo número de nodos)	85,18%	69,37%	79,52%
Porcentaje de solapamiento (FAMUS y Dijkstra)	82,18%	63,84%	63,93%
Porcentaje de solapamiento (Mínimo número de nodos y Dijkstra)	88,77%	59,16%	66,03%

Tabla 7. Comparación de trayectorias para 300 pares de nodos aleatorios

A la hora de planificar una ruta con cualquiera de los tres métodos en grafos que dispongan de zonas en las que el número de nodos de decisión sea casi nulo

(como pasillos largos) será más probable que muchos de estos nodos consecutivos coincidan en las diferentes trayectorias. Es por esta razón, por la que en aquellos grafos en los que el porcentaje de reducción de nodos y arcos es mayor (Tabla 5), el porcentaje de solapamiento de nodos entre distintas trayectorias también es mayor (Tabla 7).

Por otro lado, pese a que el grafo de *Intel* es el entorno que tiene un mayor porcentaje de nodos de decisión en comparación con el resto, se puede observar como el número de veces que las trayectorias (según los tres métodos) son idénticas es mayor. Esto se debe a que el número de nodos que tiene el grafo de *Intel* es muy inferior al de los otros grafos y por tanto al ser las trayectorias más cortas, es más probable que algunas de ellas coincidan. Lo mismo sucede en el grafo de *Bicocca* frente al de *New College*, ya que el grafo de *Bicocca* tiene un menor número de nodos y además el número de nodos de decisión es menor en proporción.

Cabe destacar que tras haber completado el experimento anterior se ha verificado en los tres entornos que para las rutas calculadas según los métodos del mínimo número de nodos y el camino más corto, ninguna de las trescientas veces se ha conseguido obtener una trayectoria con una incertidumbre acumulada (según el criterio D-opt) inferior a la ruta obtenida según FAMUS. Estos resultados verifican la limitación de los dos primeros métodos de obtener rutas con la menor incertidumbre posible.

4.2. Sobre los tiempos de cómputo

En esta sección se van a estudiar los tiempos de cómputo que tarda cada algoritmo de planificación de rutas en encontrar el camino óptimo, para lo cual se ha utilizado la función de Naive Dijkstra (Apartado 3.3.1). Para poder utilizar esta función correctamente se debe trabajar con la incertidumbre (en lugar de las coordenadas) asociada a los nodos, para que el algoritmo sea capaz de encontrar los caminos de mínima incertidumbre (y no los de menor distancia euclídea). Para ello, se ha establecido el promedio de los resultados obtenidos a partir de 20 búsquedas entre distintos pares de nodos aleatorios. En la Tabla 8 se muestran los tiempos de cómputo medio que les cuesta a cada método encontrar las trayectorias óptimas para los distintos pares de nodos en los entornos de *Bicocca*, *Intel* y *New College*.

	Bicocca	***Intel***	***New College***
Tiempo de cómputo medio de FAMUS[s]	1,34	0,56	4,1
Tiempo en buscar ruta en grafo reducido[s]	0,03	0,0086	0,074
Tiempo en buscar ruta según mínimo número de nodos (grafo completo) [s]	0,56	0,045	1,25
Tiempo en buscar ruta con menor distancia euclídea (grafo completo) [s]	0,51	0,036	1,06

Tabla 8. Comparación de los tiempos de cómputo entre los distintos métodos de búsqueda de trayectorias

A la vista de los resultados de la tabla anterior se puede ver como el tiempo necesario para obtener las trayectorias según FAMUS es superior frente al resto de métodos. Esta diferencia se debe a que para la ejecución de este algoritmo se ha contabilizado el tiempo necesario para reducir el grafo, buscar la ruta con menor incertidumbre en el grafo simplificado, y posteriormente extrapolarla al grafo completo, siendo el tiempo necesario en reducir el grafo el que acapara casi la totalidad del tiempo de ejecución. Teniendo esto en cuenta, si sobre un mismo grafo se desean evaluar varias trayectorias es suficiente reducir el grafo una vez, y sobre este realizar la búsqueda de trayectorias para conseguir una disminución en el tiempo de cómputo. Poniendo como ejemplo los datos de la tabla anterior, para los entornos de *Bicocca*, *Intel* y *New College* se necesitarían realizar 4, 18 y 5 búsquedas respectivamente como mínimo para conseguir que FAMUS fuese más rápido que planificar rutas directamente sobre el grafo completo, verificando que en los grafos

en los que hay una mayor reducción se consigue que el algoritmo sea más eficiente. En lo que respecta a los tiempos necesarios en extrapolar la trayectoria del grafo simplificado al grafo completo, se consideran prácticamente despreciables (aproximadamente el 0,30% del total).

Cabe destacar que, aunque la comparación de tiempos se hubiera hecho con la función que aplica Dijkstra sobre el grafo pesado (Apartado 3.3.2), el tiempo necesario en buscar la trayectoria con menor incertidumbre sobre el grafo simplificado siempre hubiera sido menor que buscar la trayectoria sobre el grafo completo.

Otra de las aplicaciones del algoritmo de FAMUS que lo convierte en una opción interesante es la re-planificación de rutas. Una vez que el robot ha calculado cual es el camino que considera más oportuno para alcanzar un nodo de destino, este se puede encontrar con obstáculos dinámicos que hagan que el robot tenga que volver a planificar la ruta tantas veces como obstáculos encuentre. Para ello, se puede reducir el grafo una única vez y sobre este realizar todas las re-planificaciones que sean necesarias con el objetivo de mejorar la eficiencia de esta tarea. Con esta aplicación se pretende probar que el algoritmo de FAMUS es más rápido que cualquier otro método cuando se tienen que calcular varias trayectorias sobre un mismo grafo.

En la Figura 27-a se muestra el resultado de planificar una ruta (color rojo) en el grafo de *Bicocca* según FAMUS entre dos nodos muy alejados entre sí (color verde). Sobre esta ruta se ha introducido un obstáculo (color negro) (1), con el objetivo de que cuando el robot lo alcance, este re-planifique una nueva trayectoria desde el nodo anterior hasta la pose a la que inicialmente quería llegar. El obstáculo (1) que el robot debe evitar se ha marcado con una estrella (color magenta) sobre la nueva trayectoria re-planificada, tal y como se muestra en la Figura 27-b, donde se puede ver como la nueva trayectoria (color rojo) evita el obstáculo (1). En esta misma imagen se ha dibujado el camino (color verde) que el robot iba a seguir antes de realizar la re-planificación. Siguiendo el mismo procedimiento, en la Figura 27-c y Figura 27-d se han dibujado las nuevas trayectorias que se han re-calculado debido a los obstáculos (2) y (3) que se han añadido tras realizar la primera y la segunda re-planificación respectivamente.

El tiempo necesario en reducir el grafo una vez y calcular las cuatro trayectorias sobre el grafo simplificado ha sido de 1,96 segundos, frente a los 2 segundos que ha tardado en encontrar las cuatro rutas directamente sobre el grafo completo. Hay que destacar que al igual que antes se ha utilizado la función de Naive Dijkstra (Apartado 3.3.1) para encontrar los caminos de mínima incertidumbre.

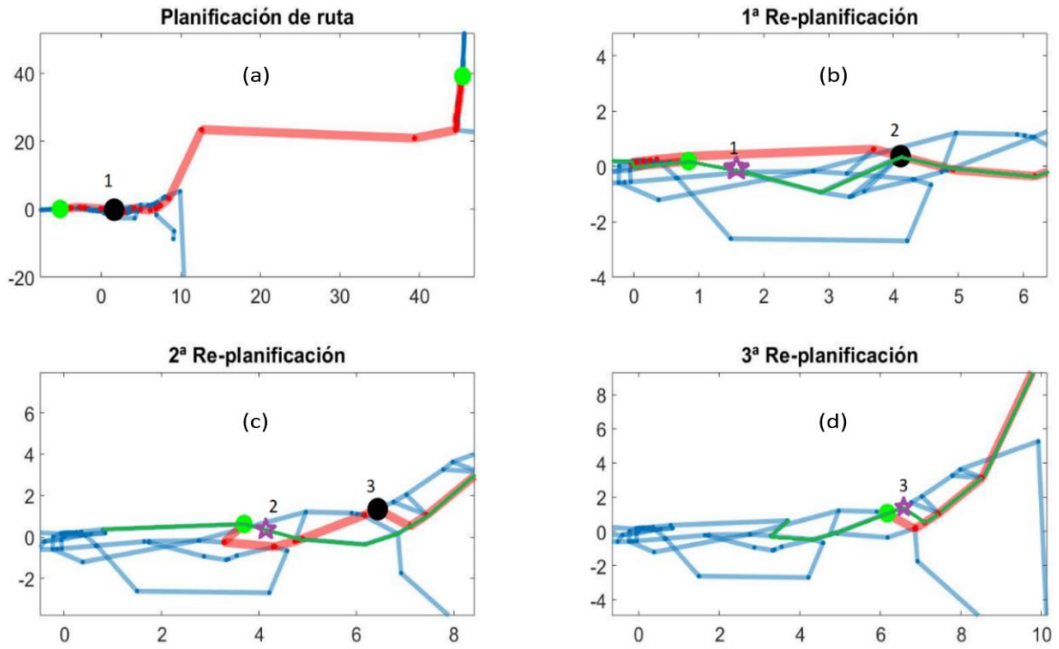

Figura 27. Experimento de re-planificación de rutas en Bicocca

Sobre estas mismas trayectorias, se ha aumentado el número de obstáculos y, por tanto, el número de veces que se tiene que volver a re-calcular la ruta, para poder comparar los tiempos obtenidos mediante los dos métodos. Los resultados de la Tabla 9 muestran una clara mejora en los tiempos necesarios para re-planificar rutas cuando se utiliza FAMUS. También se ha podido observar que cuando las rutas calculadas sobre el grafo completo están formadas por un gran número de nodos separados por arcos de odometría, la diferencia entre los tiempos necesarios en calcular esa misma ruta sobre el grafo completo y calcular la equivalente sobre el grafo simplificado es mucho más notable, lo cual tiene sentido puesto que la reducción del grafo es más efectiva en estas zonas.

Nº de re-planificaciones	**Tiempo en buscar sobre grafo completo (s)**	**Tiempo en buscar sobre grafo simplificado(s)**	**Tiempo en ejecutar FAMUS (s)**
4	2	0,17	1,96
5	2,55	0,23	2,02
6	2,70	0,27	2,06

Tabla 9. Comparación de los tiempos de búsqueda de trayectorias entre métodos al aumentar el número de obstáculos en Bicocca

Del mismo modo, este algoritmo se puede utilizar para mejorar la eficiencia de ejecución del SLAM activo. Puesto que cuando el robot está realizando tareas de exploración tiene que elegir la siguiente acción de movimiento (para lo cual debe analizar todas las posibles rutas a las distintas localizaciones que puede visitar), la

aplicación del algoritmo de FAMUS puede resultar muy útil, ya que, como se ha visto, cuando se evalúan múltiples trayectorias, el algoritmo funciona mejor en comparación con otros métodos de planificación de rutas.

Con el objetivo de simular el comportamiento del robot cuando está ejecutando tareas de SLAM activo se ha realizado un breve experimento (Figura 28) en el grafo de *Bicocca*, en el que un robot en una posición determinada (vértice en color azul), debe elegir cual es la siguiente localización a la que avanzar que le permita conocer con un mayor nivel de detalle el entorno sobre el que se encuentra. Para ello se han evaluado varias zonas posibles a las que podría ir (cada una marcada con un nodo de un color distinto) mediante la utilización de dos métodos, el algoritmo de FAMUS (imagen izquierda) de forma que únicamente es necesario reducir el grafo una vez y sobre este evaluar las distintas rutas, y la aplicación de Dijkstra directamente sobre el grafo completo (derecha).

Como se esperaba, se han obtenido las mismas trayectorias mediante los dos métodos. Sin embargo, FAMUS ha tardado 1,94 segundos frente a los 2,06 segundos de Dijkstra. Los resultados conducen a las mismas conclusiones que se han ido comentando anteriormente: contra mayor sea el número de rutas a evaluar, más eficiente será FAMUS en comparación con Dijkstra.

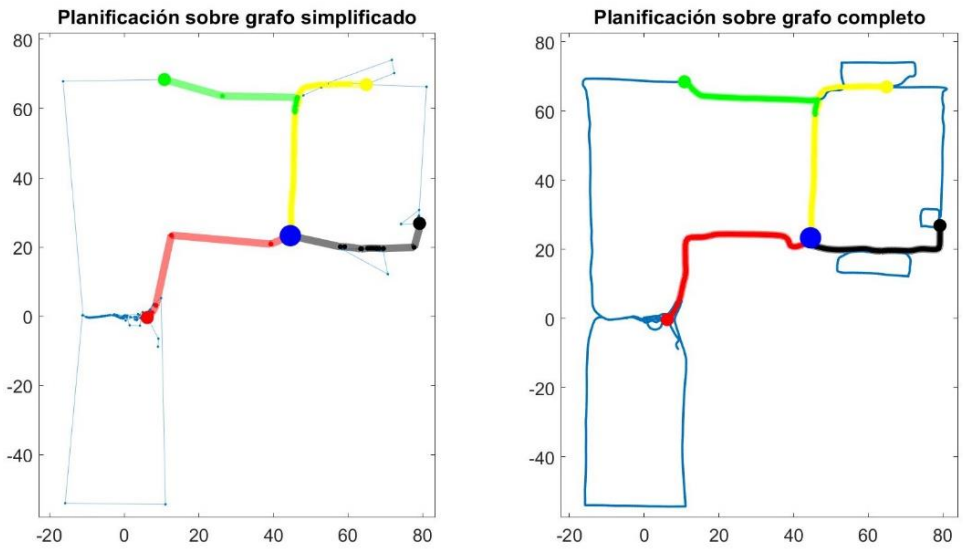

Figura 28. Representación en Bicocca de posibles rutas a las que puede ir el robot

4.3. Sobre los criterios de optimalidad

En este apartado se pretenden comparar las trayectorias de mínima incertidumbre, en grafos en los que los arcos se han pesado según los distintos criterios de optimalidad (definidos en la Sección 2.2.1). Para ello, se ha construido un *pose graph* con un tamaño fijo de 5000 nodos unidos por arcos de odometría y por una serie de arcos de re-observación que unen aleatoriamente pares de nodos separados por un número de vértices comprendido entre 10 y 1250. Cabe destacar que a cada arco de odometría y de cerrado de bucle se les ha asignado las matrices de información $\Upsilon=diag(\mathcal{N}(100,29);\mathcal{N}(100,29);\mathcal{N}(220,288))$ e $\Upsilon=diag(\mathcal{N}(1000,173);\mathcal{N}(1000,173);\mathcal{N}(2400,809))$ respectivamente, donde cada elemento de la diagonal forma parte de una distribución normal $\mathcal{N}(\mu,\sigma^2)$ de media μ y covarianza σ^2. A partir de este grafo, se han creado cuatro grafos adicionales idénticos, con la salvedad de que cada uno de ellos se ha pesado con los valores de incertidumbre calculados a partir de un criterio de optimalidad distinto que permita caracterizar la matriz de covarianza, con el objetivo de poder realizar la búsqueda del camino con menor incertidumbre según los distintos criterios entre cualquier par de nodos. Cabe destacar que la búsqueda se ha realizado entre un par de nodos muy alejados entre sí ya que de esta manera existe un mayor número de rutas alternativas entre los dos vértices (puesto que los arcos de re-observación están distribuidos a lo largo de todo el grafo) y por tanto es más probable que las trayectorias obtenidas sean distintas, de modo que la comparación entre estas permita obtener unos resultados más concluyentes.

Con el objetivo de que los resultados obtenidos a partir del experimento anterior sean concluyentes, se ha repetido el mismo procedimiento para 300 grafos diferentes, todos del mismo tamaño (5000 nodos). En todos los casos, la búsqueda se ha realizado para el mismo par de nodos fijos alejados entre sí.

A partir de la información de los 300 grafos, se ha obtenido un promedio de 665 arcos de re-observación por grafo, lo que implica una media de 2,27 arcos por nodo. En comparación con los grafos de la Tabla 1, la proporción del número de arcos de re-observación promedio de los grafos aleatorios es mayor que la de los entornos de *Bicocca* y *New College* y menor que la de *Intel*, sin embargo, el número de nodos por los que están formados los grafos aleatorios es menor que el de los entornos anteriores exceptuando el de *Intel*. Puesto que no se considera que ninguno de los grafos de la Tabla 1 tengan simultáneamente un tamaño y un número de arcos por nodo lo suficientemente grandes, y teniendo en cuenta que las matrices de información de gran parte de los arcos de los grafos de la Tabla 1 son muy parecidas, se ha optado por utilizar grafos aleatorios (según las premisas anteriores) para

realizar la comparación entre las rutas obtenidas según los distintos criterios de optimalidad, ya que de esta manera las diferencias entre las trayectorias son más notables.

En la Figura 29 se muestra uno de los grafos anteriores representado en forma circular de modo que los nodos (color rojo) están dispuestos a lo largo del perímetro, unidos por los arcos de odometría y por los de cerrado de bucle (color azul). Como se puede observar, los arcos de re-observación están dispuestos a lo largo de todo el grafo, sin embargo, nunca unen nodos muy separados entre sí, de modo que si se buscan trayectorias entre vértices muy alejados existirán varias rutas posibles.

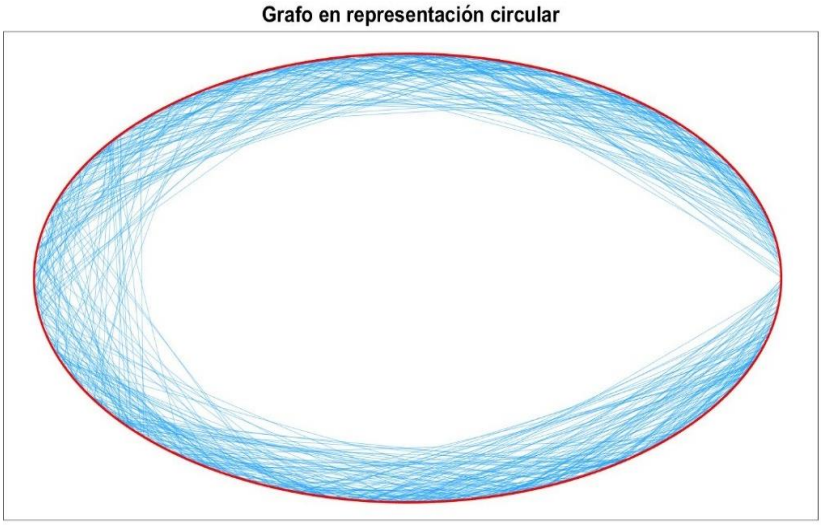

Figura 29. Representación circular de un grafo aleatorio del experimento anterior

Si a partir del grafo de la imagen anterior se realiza la búsqueda del camino con menor incertidumbre según los distintos criterios de optimalidad se obtienen las trayectorias de la Figura 30, donde los nodos de inicio y fin están indicados en negro. Como se puede observar, únicamente se muestran las rutas correspondientes a los criterios D-opt, E-opt y E*-opt, lo cual se debe a que las trayectorias obtenidas según los criterios A-opt y T-opt son las mismas que las calculadas a partir de E-opt y E*-opt, respectivamente. Esto tiene sentido puesto que, si las matrices de covarianza del grafo que codifican la incertidumbre de los nodos representan elipses muy excéntricas, el máximo y el mínimo valor propio van a ser muy predominantes con respecto al resto y, por tanto, estos valores serán muy parecidos a la suma de todos los valores propios (A-opt) y al sumatorio de la inversa de cada valor propio (T-opt) respectivamente, por lo que al obtener valores muy parecidos de incertidumbre en los arcos, cuando el algoritmo trate de buscar la trayectoria con la menor incertidumbre encontrará la misma ruta.

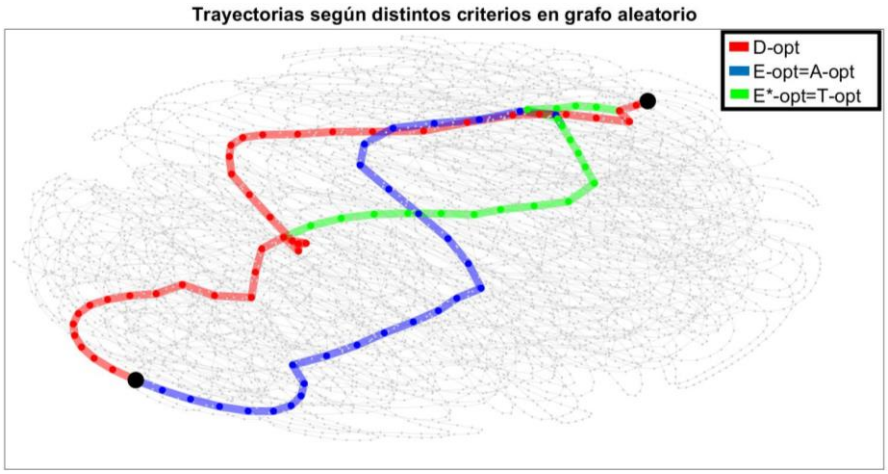

Figura 30. Caminos de mínima incertidumbre según los distintos criterios para el grafo aleatorio de la Figura 29

Si se realiza la búsqueda de los caminos con menor incertidumbre según los distintos criterios de optimalidad entre el par de nodos fijos seleccionado para cada uno de los 300 grafos, se obtienen los resultados que se muestran en la Figura 31. En este diagrama de dispersión se representan los valores de incertidumbre acumulada de cada una de las trayectorias obtenidas según los distintos criterios para cada uno de los grafos, junto con las líneas de ajuste que permiten representar la tendencia de cada conjunto de datos. Como se puede observar, los valores de incertidumbre acumulada según los distintos criterios mantienen el mismo orden que se vio en la Figura 4, de modo que el E-opt<A-opt<D-opt<T-opt<E*-opt.

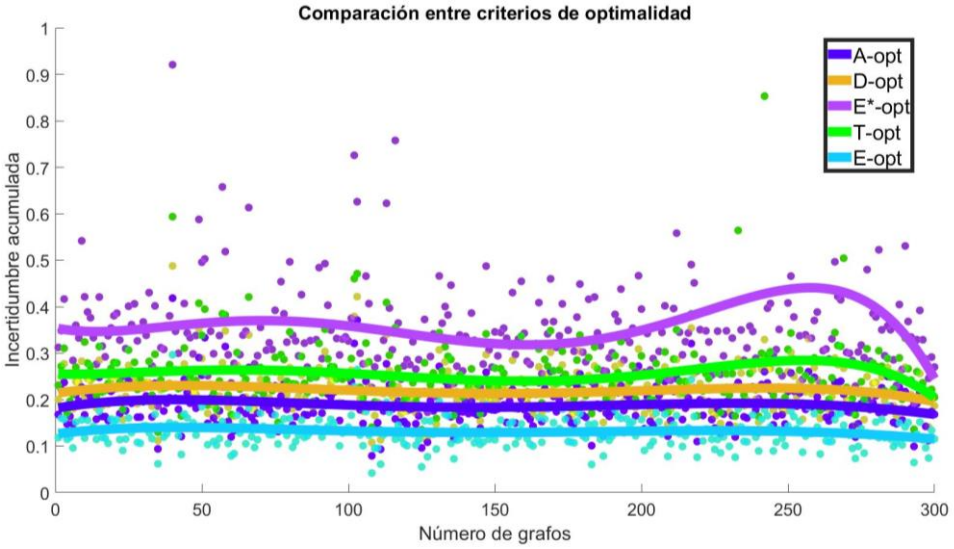

Figura 31. Incertidumbre acumulada de cada una de las rutas de mínima incertidumbre según los distintos criterios en los 300 grafos

Puesto que, dependiendo del criterio de optimalidad, el valor de incertidumbre puede ser distinto (tal y como se ha visto en la figura anterior), a partir de las rutas anteriores obtenidas de ejecutar FAMUS según los distintos criterios de optimalidad

en cada uno de los grafos, se ha utilizado un criterio común (en este caso el criterio D-opt) para comparar la incertidumbre acumulada de cada una de estas trayectorias. En la Figura 32 se muestran estos valores normalizados con los valores de incertidumbre acumulada de la trayectoria obtenida de aplicar FAMUS según el criterio D-opt en cada uno de los grafos.

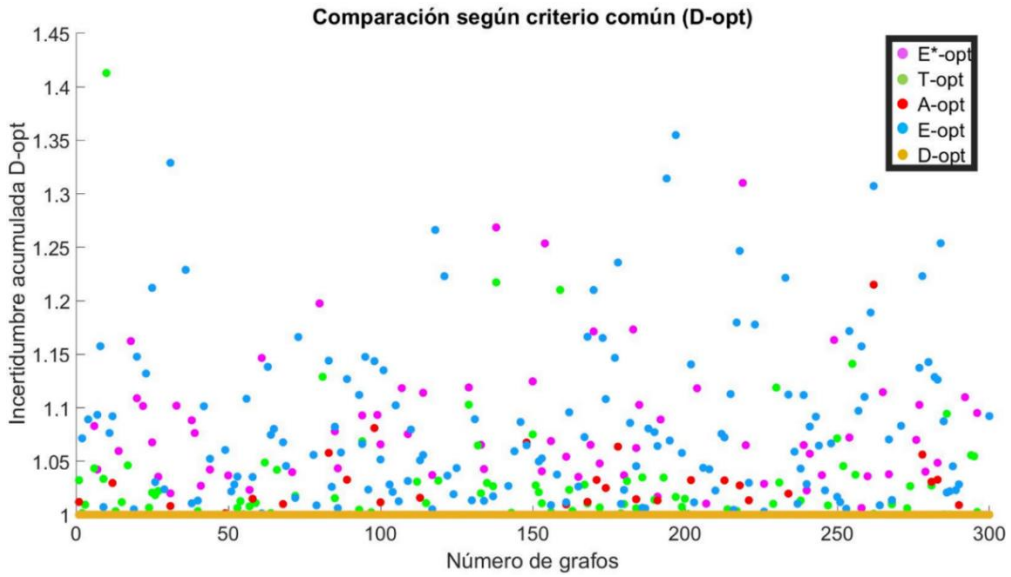

Figura 32. Incertidumbres acumuladas normalizadas (criterio D-opt) de cada una de las rutas de mínima incertidumbre obtenidas de aplicar FAMUS según los distintos criterios en los 300 grafos

Como se puede observar, ninguna de las rutas obtiene un valor de incertidumbre acumulada inferior a la unidad, lo que indica que las rutas obtenidas usando el criterio D-opt obtienen la incertidumbre acumulada más baja (según el criterio común D-opt) en comparación con el resto de los criterios. Este resultado es el esperado, puesto que como se comentó en la Sección 2.2.1, el criterio D-opt es el único que captura la matriz de covarianza completa, motivo por el cual se utilizó como criterio para pesar los arcos de los grafos del Apartado 3.4 en los que se presentó el algoritmo de FAMUS.

Por otro lado, también se ha podido observar que cuando se aplica FAMUS sobre dos grafos idénticos pero cada uno ponderado con un criterio de optimalidad distinto, los valores de incertidumbre acumulada (según el criterio común D-opt) coinciden en un porcentaje elevado de las ocasiones cuando los criterios utilizados para pesar los grafos aproximan las matrices de covarianza con valores muy próximos entre sí. Esto implica que cuando el algoritmo trate de buscar la ruta de mínima incertidumbre entre dos nodos sobre los grafos ponderados según estos criterios, este encontrará en un porcentaje elevado de las ocasiones el mismo camino.

En la Tabla 10 se muestra el porcentaje de veces que se obtienen las mismas rutas para diferentes pares de criterios de optimalidad. Efectivamente, se puede comprobar como las cuatro primeras filas de la tabla se corresponderían con los casos en los que los distintos pares de criterios aproximan la matriz de covarianza de cada arco con valores muy próximos, obteniendo unos porcentajes de coincidencia entre rutas superiores al 70% en la mayoría de los casos, frente a los pares de criterios definidos en las dos últimas filas (A-opt/T-opt y E-opt/E*opt) cuyos porcentajes son muy inferiores. Aunque estos dos últimos pares de criterios aproximan las matrices de covarianza mediante valores muy distintos entre sí, se puede ver como el porcentaje de coincidencia entre rutas es no nulo, lo cual se debe a que no todos los grafos que forman parte del experimento tienen un elevado número de rutas alternativas con una incertidumbre acumulada baja, por lo que si uno de estos caminos es considerablemente mejor que el resto, aunque se utilicen criterios distintos el algoritmo siempre encontrará la mismo trayectoria.

	Porcentajes de coincidencia de rutas
E-opt y A-opt	67%
A-opt y D-opt	73%
D-opt y T-opt	71%
T-opt y E*-opt	76%
A-opt y T-opt	50%
E-opt y E*-opt	25%

Tabla 10. Porcentaje de veces que las trayectorias obtenidas según los distintos criterios son las mismas

Por otro lado, con el objetivo de analizar los tiempos de cómputo necesarios en calcular las incertidumbres según los distintos criterios de optimalidad de los valores propios de las matrices de covarianza, se ha realizado un experimento en el que se han calculado 100000 veces tres valores propios aleatorios escogidos dentro un rango de valores coherente (cero y uno) a partir de los cuales se han medido los tiempos necesarios en obtener la incertidumbre según cada criterio. Adicionalmente, con el fin de analizar como crece la complejidad computacional de calcular la incertidumbre según cada criterio al aumentar el tamaño de la matriz de covarianza, se ha repetido el experimento anterior para 10000 valores propios. En la Tabla 11 se muestran los tiempos de cómputo promedio de cada criterio para los dos casos anteriores.

Criterios	Tiempo de cómputo (ns) a partir de 3 valores propios	Tiempo de cómputo (ns) a partir de 10000 valores propios
E-opt	174	4426
E*-opt	171	3479
T-opt	163	2988
A-opt	294	20957
D-opt	438	35268

Tabla 11. Tiempo para calcular los distintos criterios a partir de 3 y 10000 valores propios

Como se puede observar, los criterios E, E* y T-opt son los que menos tiempo tardan frente al A y el D-opt, siendo este último el que más tiempo tarda, lo cual tiene sentido puesto que los tres primeros únicamente recogen, respectivamente, el valor del mínimo y el máximo valor propio, así como la media de los valores propios. Por el contrario, los criterios A y D-opt capturan, respectivamente, la media armónica y el volumen de la (hiper)elipsoide (mediante el determinante de la matriz de covarianza), lo cual es más lento computacionalmente. Para el caso de los 10000 valores propios se puede ver como los tiempos de cómputo asociados a los criterios A-opt y sobre todo al D-opt han aumentado respecto al caso anterior notablemente más que el resto. Esto indica un crecimiento más rápido de la complejidad computacional de estos criterios conforme aumenta el número de valores propios. Por tanto, contra mayor sea el tamaño de las matrices de covarianza o de información (y por tanto el número de valores propios) a evaluar, más tiempo les costara a estos criterios obtener los valores de incertidumbre en proporción al resto.

Se puede concluir, por tanto, que pese a que el criterio D-opt es el que requiere un mayor tiempo de cómputo, es el único capaz de encontrar el 100% de las veces el camino de mínima incertidumbre entre cualquier par de nodos. Sin embargo, puede haber ocasiones en las que la utilización de otro criterio de optimalidad para aproximar la matriz de covarianza puede resultar interesante. Si se utiliza un criterio, como puede ser el T-opt, capaz de capturar bien el comportamiento de la matriz de incertidumbre debido por ejemplo a que esta tiene pocos elementos fuera de la diagonal, su aplicación con respecto al criterio D-opt puede resultar más eficaz debido a la escasa diferencia en la aproximación de la incertidumbre y a la clara mejora en los tiempos de cómputo.

5. Conclusiones

Este Trabajo Fin de Grado se ha desarrollado en el marco de los problemas de SLAM y SLAM Activo a través de la formulación de grafos basados en poses.

Se han estudiado e implementado varios métodos de planificación de rutas clásicos basados en diferentes criterios, y se han comparado entre sí mediante su aplicación en grafos correspondientes a entornos reales. A través de estas comparaciones, se han evidenciado las ventajas y limitaciones de cada uno de ellos, y se ha demostrado que la complejidad de estos algoritmos incrementa cuando el número de nodos y arcos del grafo aumenta. En este contexto, en base a [1], se ha implementado una versión actualizada del algoritmo FAMUS. Las mejoras logradas con respecto a los anteriores algoritmos son dos: garantiza encontrar el camino de mínima incertidumbre, y supone una mejora en el tiempo de cómputo cuando se realizan varias búsquedas sobre un mismo grafo. Este algoritmo, que representa el estado del arte, permite realizar la búsqueda directamente sobre aquellos vértices del grafo en los que el robot puede tomar una decisión entre, al menos, dos trayectorias: los nodos de decisión.

Mediante la aplicación en grafos de entornos reales se ha simulado como se comportaría el algoritmo cuando el robot está re-planificando rutas o ejecutando SLAM activo. Se ha demostrado que, en estos casos, en algunos entornos únicamente es necesario buscar cuatro trayectorias para conseguir que el algoritmo sea más rápido que el resto de métodos. Esto es debido a que la búsqueda de la ruta óptima la puede hacer directamente sobre el grafo simplificado.

Finalmente, se ha hecho un estudio sobre las diferencias de usar distintos criterios basados en TOED. A partir de un amplio experimento en el que se han elaborado grafos aleatorios, se ha demostrado que la caracterización de la incertidumbre a través del criterio D-opt garantiza el 100% de las veces el camino de mínima incertidumbre. Sin embargo, se ha observado que el tiempo necesario para calcular este criterio es muy elevado, por lo que en aquellos casos en los que la aproximación de la incertidumbre con otro criterio fuese próxima a la obtenida con D-opt, podría ser más adecuado la utilización de dicho criterio.

Como tareas de trabajo futuro se plantea:

- Experimentación con un robot real equipado con un sistema sensorial basado en visión que permita ejecutar un algoritmo de SLAM visual activo.

- Integración de la estrategia de planificación basada en el criterio D-opt en un algoritmo de aprendizaje basado en redes neuronales profundas.
- Extensión de los algoritmos de planificación de rutas a grafos para los que se desconozcan parte de sus arcos (*link prediction*).

Bibliografía

[1] H. Carrillo, Y. Latif, J. Neira, and J. A. Castellanos, "Fast minimum uncertainty search on a graph map representation," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 2504–2511, 2012, doi: 10.1109/IROS.2012.6385927.

[2] C. Stachniss, J. J. Leonard, and S. Thrun, "Simultaneous localization and mapping," in *Springer Handbook of Robotics*, 2016, pp. 1153–1175.

[3] M. Pfingsthorn, "Generalized Simultaneous Localization and Mapping (SLAM) on Graphs with Multimodal Probabilities and Hyperedges," no. May, 2013.

[4] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intell. Transp. Syst. Mag.*, vol. 2, no. 4, pp. 31–43, 2010, doi: 10.1109/MITS.2010.939925.

[5] F. Lu and E. Milios, "Globally Consistent Range Scan Alignment for Environment Mapping," *Auton. Robots*, vol. 4, no. 4, pp. 333–349, 1997, doi: 10.1023/A:1008854305733.

[6] S. Thrun, "Probabilistic Robotics," vol. 45, no. 3, pp. 52–57, 2002.

[7] H. Durrant-Whyte and T. Bailey, "Simultaneous Localization and Mapping: Part I," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, 2006.

[8] C. Cadena *et al.*, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, 2016, doi: 10.1109/TRO.2016.2624754.

[9] L. Carlone, R. Aragues, J. A. Castellanos, and B. Bona, "A fast and accurate approximation for planar pose graph optimization," *Int. J. Rob. Res.*, vol. 33, no. 7, pp. 965–987, 2014, doi: 10.1177/0278364914523689.

[10] L. J. Fermin, "Exploiting graph structure in Active SLAM," Universidad de Zaragoza, 2018.

[11] H. Carrillo, I. Reid, and J. A. Castellanos, "On the comparison of uncertainty criteria for active SLAM," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 2080–2087, 2012, doi: 10.1109/ICRA.2012.6224890.

[12] A. A. Makarenko, S. B. Williams, F. Bourgault, and H. F. Durrant-Whyte, "An experiment in integrated exploration," *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 1, pp. 534–539, 2002, doi: 10.1109/irds.2002.1041445.

[13] F. Pukelsheim, "Optimal design of experiments," *Soc. Ind. Appl. Math.*, 2006.

[14] H. Carrillo, P. Dames, V. Kumar, and J. A. Castellanos, "Autonomous robotic exploration using a utility function based on Rényi's general theory of entropy," *Auton. Robots*, vol. 42, no. 2, pp. 235–256, 2018, doi: 10.1007/s10514-017-9662-9.

[15] K. J, "General Equivalence Theory for Optimum Designs (Approximate Theory)," *Ann. Stat.*, vol. 2, no. 5, pp. 849–879, 1974.

[16] J. A. Placed and J. A. Castellanos, "Fast Autonomous Robotic Exploration Using the Underlying Graph Structure," no. Section IV, 2021.

[17] J. Núñez, M. Pérez, S. B. Guillén, R. Diánez, and C. De Elías, "Siete puentes, un camino: Königsberg," *Suma*, vol. 45, pp. 69–78, 2004, [Online]. Available:

http://revistasuma.es/IMG/pdf/45/069-078.pdf.

[18] "RAWSEEDS, Robotics advancement through Webpublishing of sen- sorial and elaborated extensive data sets (project FP6-IST-045144)," 2009, [Online]. Available: http://www.rawseeds.org/rs/datasets.

[19] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman, "The new college vision and laser data set," *Int. J. Rob. Res.*, vol. 28, no. 5, pp. 595–599, 2009, [Online]. Available: http://www.robots.ox.ac.uk/NewCollegeData/.

[20] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general Framework for Graph Optimization," 2011.

[21] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent," *Robot. Sci. Syst.*, vol. 3, pp. 65–72, 2007, doi: 10.15607/rss.2007.iii.009.

[22] "Graph and Network Algorithms." https://es.mathworks.com/help/matlab/graph-and-network-algorithms.html?lang=en.

Lista de figuras

Lista de tablas