Control de Robots Móviles Autónomos: Planificación de rutas en mapas geométricos

David García Centelles

Resumen—La implementación de los AGV (Automated Guided Vehicles) en pequeñas empresas puede suponer un coste inicial demasiado elevado, por lo que son necesarios AGVs económicos donde el peso de la implementación recae en el software. En este documento se presenta el desarrollo e implementación de un algoritmo para la planificación de rutas en un mapa 2D a partir de su transformación en un mapa topológico. Se han estudiado las alternativas existentes y se han analizado nuevas hasta encontrar una solución adecuada, que consiste en un método propio basado en uno de expansión. Como resultado se ha obtenido un algoritmo funcional que se ha probado en diferentes escenarios para analizar su comportamiento.

Palabras clave—AGV, mapas geométricos, mapas topológicos, planificación de rutas, método de expansión.

Resum— L'implementació dels AGV (Automated Guided Vehicles) en petites empresas pot suposar un cost inicial massa elevat, per la qual cosa són necessaris AGVs econòmics on el pes de l'implementació recau en el software. En aquest document es presenta el desenvolupament i l'implementació d'un algorisme per a la planificació de rutes en un mapa 2D a partir de la seva transformació en un mapa topològic. S'han estudiat les alternatives existents i s'han analitzat de noves fins a trobar una solució adequada, que consisteix en un mètode propi basat en un d'expansió. Com a resultat s'ha obtingut un algorisme funcional que s'ha provat en diferents escenaris per analitzar el seu comportament.

Paraules clau—AGV, mapes geomètrics, mapes topològics, planificació de rutes, mètodo d'expansió.

Abstract—Implementation of AGV (Automated Guided Vehicles) in small businesses may have too high initial cost, which is why we need economical AGVs where the burden of implementation lies with the software. This document describes the development and implentation of an algorithm for route planning on a 2D map from its transformation into a topological map. We have studied the existing alternatives and analyzed new ones to find an appropriate solution, which consists in a method based on one of expansion. As a result we have obtained a functional algorithm that has been tested in different scenarios to analyze their behavior.

Index Terms—AGV, geometrical maps, topological maps, route planning, expansion method.

1 Introducción

LOS AGV (*Automated Guided Vehicles* [1]) son, cada vez más, una herramienta utilizada por organizaciones y empresas para mejorar la eficiencia de sus procesos y la competividad de su estrategia de negocio. Esto es así debido a las diferentes aplicaciones que se le puede dar a la tecnología, tales como la automatización de las cadenas de montaje, el soporte de operaciones manuales o el transporte de materiales, entre otras.

A día de hoy no es necesario convencer nadie de que el uso de AGVs supone un ahorro económico [2]. Muchas son las empresas u organizaciones que han decidido centrar su actividad en esta solución, como es el caso de la empresa *Seat*, en España, o la empresa *Mondi*, en Austria, que utilizan AGVs desarrollados por *Artisteril* o el hospital universitario de *St.Olav*, en Noruega, que utiliza AGVs desarrollados por la empresa *Swisslog*.

- *E-mail de contacte: David.GarciaC@e-campus.uab.cat*
- *Menció realitzada: Enginyeria de Computadors*
- *Treball tutoritzat per: Lluís Ribas Xirgo (Departament de Microelectrònica i Sistemas)*
- *Curs 2014/15*

Sin embargo, a la hora de asegurar la rentabilidad de esta solución en pequeñas o medianas empresas nos encontramos con que el impacto del coste inicial de los AGVs y su mantenimiento puede resultar un problema. Para solucionar dicho problema se crean AGVs más económicos, que presentan una reducción de la cantidad de sensores que utilizan, la capacidad de los mismos, y la infraestructura necesaria para dirigirlos. Esto provoca que el AGV tenga que trabajar con una información limitada del entorno que lo rodea, haciendo que el peso de la implementación de sus algoritmos de control recaiga en la complejidad del *software*.

Los AGV de bajo coste son cada vez más buscados y actualmente aún se investigan soluciones eficientes para implementar las funcionalidades requeridas para su control. El trabajo que se presenta en este documento consiste en el desarrollo e implementación de una de estas funcionalidades. Con dicha funcionalidad buscamos una forma para lograr la planificación de rutas del AGV dentro de un escenario concreto. Del escenario conocemos un mapa 2D que lo representa y las coordenadas de varios puntos en el mapa que representan lugares de interés a los que el AGV tiene que poder viajar. Es importante conocer que

para llevar a cabo este trabajo ya se dispone de un AGV que cumple con las funciones básicas para el movimiento y la detección de obstáculos, aunque el mismo es utilizado solamente a nivel de simulación.

Los métodos para planificar rutas dentro de mapas 2D son demasiado complejos debido a la cantidad de información que se tiene que tratar. Por ello, existe la posibilidad de que previo a la búsqueda de rutas se realice una transformación sobre el mapa. En nuestro caso se utiliza una transformación del mapa 2D a un mapa topológico debido a que ya formaba parte de los requisitos iniciales del trabajo. Gracias a este cambio nuestros datos iniciales se han transformado en algo tan sencillo como una serie de puntos unidos por relaciones con un cierto coste asociado a cada una, es decir, un grafo ponderado. Por lo que el método para planificar una ruta se transforma en la búsqueda del camino de coste mínimo dentro de un grafo, problema que se ha solucionado ya a través de muchos algoritmos [3]. Pero con los puntos que tenemos al principio es posible que no se pueda generar un grafo completo con relaciones directas entre ellos que no pasen por obstáculos del escenario, motivo por el cuàl ha sido necesario también generar nuevos puntos dentro del mapa topológico como parte del algoritmo.

La implementación de esta funcionalidad y su posterior simulación se han realizado con el lenguaje de programación *Netlogo*, debido a que ya existía la implementación de una simulación de AGVs con este lenguaje. Tras realizar pruebas con diferentes casos de uso se han corregido algunas deficiencias del algoritmo inicial y se ha demostrado que la solución propuesta funciona, aunque presenta ciertas limitaciones de optimización.

El documento presentado a continuación está dividido en las siguientes partes. En la Sección 2 veremos el estado arte del algoritmo para la planificación de rutas a partir de mapas 2D. En la Sección 3 describiremos la funcionalidad que queremos implementar, dividida en varias partes. En la Sección 4 analizaremos diversas alternativas para lograr la funcionalidad requerida. Veremos alternativas existentes y propondremos propias, y elegiremos una. En la Sección 5 hablaremos del proceso que se ha seguido para llevar a cabo la implementación y la simulación de la solución elegida para nuestro algoritmo. Además, se hablará de las diferentes modificaciones que ha sido necesarias para suplir algunas deficiencias existentes en el algoritmo inicial tras la observación de su comportamiento en diferentes casos de uso. En la Sección 6 valoraremos los resultados finales de nuestro algoritmo. En la Sección 7 se presentarán las conclusiones a las que se han llegado tras la realización de este trabajo.

2 ESTADO DEL ARTE

El problema de la planificación de rutas de desplazamiento para los AGV ha sido desde siempre unos de los temas de más interés en este campo. La complejidad de dicho problema se basa principalmente en dos factores fundamentales, localizar al AGV dentro del entorno y conocer como es el entorno que lo rodea. A esta problemática se la conoce como SLAM (Simultaneous Localization And Mapping [4]), y es la base sobre la que se sustentan la mayoría de algoritmos de planificación de rutas. Para no entrar en detalle en SLAM, una explicación que hemos encontrado muy interesante es encontrada en [5].

El problema de la planificación de rutas puede ser simplificado en primer lugar a nivel de *hardware*, ya que cuando mejores son los sensores y mejor preparadas están las instalaciones donde se moverán los AGV más fácil resultará localizarlos dentro del escenario y conocer que ordenes se le tienen que dar para moverse por el mismo. El punto negativo de realizar esta simplificación es el elevado coste de los sensores y la infraestructura y las limitaciones que nos podemos encontrar cuando queremos cambiar el entorno del AGV de lugar. Nosotros no hemos tratado esta opción debido a que supondría la modificación del AGV con el que vamos a trabajar, que, como hemos mencionado la Sección 1, está preparado para ser una solución de bajo coste.

Por otro lado, están las soluciones *software* del problema. Existen tres familias de algoritmos principales para SLAM, de las cuales derivan la mayor parte de las implementaciones que existen para la planificación de rutas.

En primer lugar tenemos los algoritmos basados en el filtro de Kalman. Dichos algoritmos se basan en la estimación de la localización del robot mediante el conocimiento de su estado y de puntos característicos del mapa en el que se encuentra con cierto margen de error. Un ejemplo de la aplicación de este algoritmo es [6].

En segundo lugar tenemos los algoritmos basados en el filtro de partículas. Dichos algoritmos se basan en la estimación de la posición y el movimiento del AGV a partir de conjuntos de partículas. Un ejemplo de la aplicación de este algoritmo es [7].

Por último, y son los que nos resultan de más interés, están los algoritmos basados en grafos. En este tipo de algoritmos tomamos la posición del AGV y de los elementos característicos del mapa como puntos dentro de un grafo, y los caminos entre ellos como el movimiento entre dichos puntos. Como es evidente, nos interesan estos algoritmos debido a que la solución a la que queremos llegar es derivada de los mismos. Un ejemplo de la aplicación de este algoritmo es [8].

Derivados de los últimos algoritmos mencionados, están también los algoritmos que suponen uno de los puntos principales de este trabajo, que son los de la transformación de mapas 2D a mapas topológicos. En la Sección 4.1 se mencionan algunas alternativas existentes para este tipo de algoritmos, razón por la cuàl no son incluídas en esta sección.

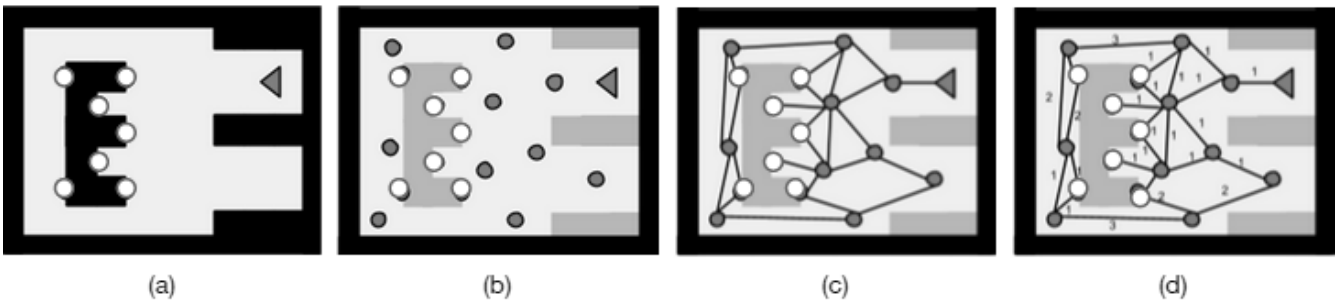


Fig. 1. *(a)* Mapa 2D inicial con nodos de interés y nodo raíz. *(b)* Mapa topológico de nodos con nuevos nodos generados. *(c)* Mapa topológico con las relaciones entre los nodos. *(d)* Mapa topológico con el coste en las relaciones.

3 FUNCIONALIDAD REQUERIDA

La funcionalidad que buscábamos implementar se divide en dos algoritmos bien diferenciados. El primer algoritmo consiste en la generación del mapa topológico a partir del mapa 2D y el segundo consiste en la generación del camino de coste mínimo entre dos nodos del mapa topológico.

3.1 Generación del mapa topológico

Por tal de explicar este algoritmo tomaremos como ejemplo el escenario representado en la Fig. 1. Imaginamos que nuestro mapa 2D, que puede representar por ejemplo el plano de una fábrica, es el que vemos en la Fig. 1*(a)*. Las zonas en negro representan los obstáculos del mapa por los que el AGV no puede moverse. Los puntos blancos representan los puntos de interés a las que queremos que el AGV pueda viajar, aunque por tal de facilitar la comprensión de nuestra solución como un grafo llamaremos a dichos puntos "nodos de interés". El triángulo representa la posición inicial de nuestro AGV, pero debe considerarse como otro nodo de interés al que le damos el apelativo de "nodo raíz". Teniendo esto como datos iniciales, podemos considerar que la transformación del mapa 2D a mapa topológico se divide en dos partes.

La primera parte consiste en la comprobación de que exista algún camino formado por relaciones directas desde cada nodo de interés hacia el nodo raíz. El significado de lo que es una relación directa está incluído en la Sección 4.2.1, por lo que para este ejemplo consideraremos que una relación directa es una línea recta entre dos nodos que no pasa por un obstáculo. Si existe dicho camino entre cada nodo y el nodo raíz querrá decir que tenemos un grafo totalmente interconectado, es decir, el mapa topológico que queríamos y el algoritmo se da por finalizado. En caso de que al menos uno de los nodos esté aislado del resto hará falta pasar a la segunda parte del algoritmo.

La segunda parte consiste en la generación de nuevos nodos. En este ejemplo consideraremos que la generación se hace de forma aleatoria sin tener en cuenta los nodos sin relación encontrados en la primera parte, pero como se analizará en la Sección 4.4.2, existen alternativas que si lo tienen en cuenta. En la Fig. 1*(b)* se ve el resultado de generar los nodos, representados con los puntos negros.

Una vez generados los nodos volveremos a la primera parte del algoritmo. De esta forma se crea un bucle hasta que consigamos el mapa topológico totalmente interconectado, por lo que es necesario que la implementación asegure siempre una solución o el algoritmo se mantendrá en un bucle infinito. El mapa topológico resultante se puede ver en la Fig. 1*(c)*, donde se han generado ya las relaciones entre los nodos, tanto los existentes como los que hemos creado.

Finalmente, en la Fig. 1*(d)* se asignan los costes a cada relación dependiendo de la distancia entre los nodos por tal de conseguir el grafo ponderado.

3.2 Generación del camino de coste mínimo

A partir del grafo ponderado conseguido con el primer algoritmo ya podemos determinar que ruta debe seguir el AGV para llegar de un nodo de interés a otro. Debido a que ya existen muchas implementaciones del algoritmo de búsqueda del camino más corto a partir de un grafo ponderado, nosotros solo nos hemos encargado de elegir entre las alternativas, por lo que el peso real del trabajo recae principalmente en el primer algoritmo.

4 ANÁLISIS DE ALTERNATIVAS

El análisis de alternativas ha consistido en primer lugar en una búsqueda de algoritmos existentes para la generación de mapas topológicos a partir de mapas 2D. Esto nos ha servido para inspirarnos y al mismo tiempo evitar crear algo que ya existiese. De las conclusiones extraídas de dicha búsqueda se han propuesto varias alternativas propias, que se han valorado en conjunto para acabar escogiendo aquella que más se adaptaba a nuestras necesidades. Las alternativas para la generación del camino de coste mínimo se han valorado dentro de las alternativas propuestas debido a que solo hemos valorado varias posibilidades concretas entre los algoritmos existentes.

4.1 Alternativas existentes

Entre las alternativas existentes consultadas hay varias que hemos considerado interesantes de mencionar.

En primer lugar tenemos el método SSH (*Spatial Semantic Hierarchy*) [9], que se ha refinado y reutilizado a lo largo del tiempo, derivando en muchos otros métodos que funcionan de forma híbrida con este. Dicho método se basa en organizar la representación del mapa en una estructura de jerárquica de cuatro niveles, donde cada

cuàl representa el escenario sobre el que se mueve el AGV de una forma de más a menos abstracta. A nosotros solo nos interesan los dos ultimos niveles, conocidos como el nivel *Geométrico* y el nivel *Topológico*. Dentro del método existe una forma de pasar del mapa geométrico al topológico, que consiste principalmente en explorar el mapa 2D y marcar los lugares visitados. Durante la exploración se van registrando los caminos que ha realizado el AGV, si en algún punto dos caminos se cruzan entonces se crea un nuevo nodo en dicha localización. Las relaciones entre son el camino ya registrado para llegar de un nodo a otro.

De este método nos llamó la atención especialmente la creación de nuevos nodos, pero lo consideramos ineficiente debido a que era necesario guardar en memoria todo el recorrido de un nodo a otro y además era un método pensado para escenarios basados principalmente en recorridos, cosa que limitaba su uso en otros escenarios.

En segundo lugar, hemos encontrado otro método que puede ser visto en [10], donde tenemos tambien un mapa 2D con las coordenadas de ciertas puntos de interés conocidas y la posición del robot, como en nuestro caso. Lo que se hace en este método para generar el mapa topológico es que desde cada posición se hace una comprobación de su situacion respecto al resto de posiciónes. Eso quiere decir que comprobamos a que posiciónes podemos acceder desde la que estamos y a que distancia estan. Se elige la posición o posiciónes más cercanas posibles y se generan las relaciones entre ellas.

Este método tambien parecía aplicable a nuestro caso pero se presupone en el mismo que existe un camino para llegar desde cada posición conocida al resto. Esto es debido a que nosotros pensamos solo en relaciones directas y ellos piensan en relaciones que implican un cierto recorrido.

En tercer lugar, tambien podemos encontrar un ejemplo que no es estrictamente de creacion de mapas topológicos, pero que se puede derivar a ello, que consiste en el uso de un *grid*. En [11] podemos ver una comparativa entre el uso de *grid* y el uso de mapas topológicos. Afirmamos esta equivalencia debido a que podemos considerar que todos los cuadros del *grid* corresponden a un punto en el mapa topológico. Usando este método por otro lado tendríamos que definir exactamente cuál es nuestro grid, teniendo que llegar a redimensionar las coordenadas utilizadas en el mapa 2D.

A parte de los comentados, hay muchos más ejemplos como son [12] y [13], que nos demuestran el amplio abanico de soluciones posibles que hay detrás de este problema y la importancia que se le da a encontrar nuevas.

4.2 Alternativas propuestas

La funcionalidad requerida definida en la Sección 3 tiene tres aspectos que podrían variar en su implementación:

1. La comprobación de las relaciones directas entre los nodos, de la que queremos saber realmente como generar dichas relaciones directas.
2. La generación de nuevos nodos en el mapa.
3. La búsqueda del camino de coste mínimo.

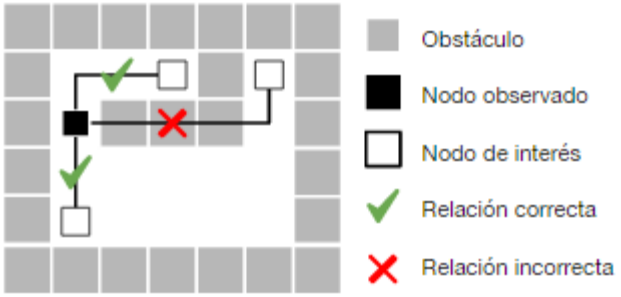


Fig. 2. Criterio a seguir para discernir si existe una relación directa entre dos nodos

4.2.1 Generación de relaciones entre nodos

Lo que buscamos definir a través de las alternativas en cuanto a la generación de relación es que es lo que representa una relación directa entre dos nodos. En este caso solo se ha optado por una alternativa. La misma es que un nodo tiene una relación directa con otro si están unidos por una línea recta horizontal o vertical, o si existe un camino entre ellos formado por dos líneas rectas que forman un angulo de 90º. Este criterio se ve de forma más clara en la Fig. 2. El coste de dicha relación vendría representado por la distancia recorrida por el AGV para llegar de un nodo al otro.

Es importante también tener en cuenta que para comprobar una relación es necesario hacer dos comprobaciones, primero una en la que se compruebe una línea horizontal y luego una vertical, y después otra a la inversa.

El motivo principal por el que se ha escogido esta simplificación es que se ha supuesto que la mayoría de organizaciones o empresas que utilizan AGVs lo hacen en recintos cerrados donde los obstáculos pueden ser representados en el mapa como cuadrados o rectángulos, por lo que para el AGV moverse por el escenario supone mayormente realizar movimientos en línea recta o dar giros de 90º.

4.2.2 Generación de nuevos nodos

Se han analizado diversas alternativas en cuanto a la generación de nodos, mostrando los puntos a favor y en contra que cada uno de ellas presentaba.

4.2.2.1 Aleatorio

Esta quizás es la posibilidad más trivial, que sería la generación de nuevos nodos en el mapa de forma aleatoria. La cantidad de nodos generados dependería del tamaño del mapa. La principal ventaja de esta alternativa es su fácil implementación, pero en contra tiene dos grandes desventajas, 1) Dos nodos pueden encontrarse en una posición muy cercana, creando caminos inútiles, 2) Un nodo creado o existente puede seguir siendo inaccesible por el resto de nodos tras la generación, teniendo que ejecutarla repetidas veces y provocando que se accentue la primera desventaja. Para solucionar ambas desventajas se tendría añadir una complejidad no deseada al algoritmo y siempre dependeriamos del factor de aletoriedad, por lo que no se podría asegurar una solución.

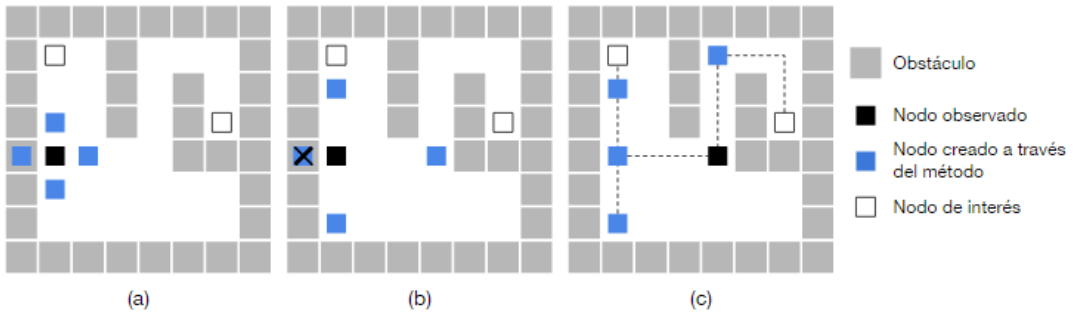


Fig. 3. *(a)* Inicialización del método de Expansión. *(b)* Creación de nodos en cuatro direcciones. *(c)* Resultado final, los nodos creados ya pueden formar un grafo interconectado completamente.

4.2.2.2 Grid

Consiste en usar directamente la propia *grid* que nos aporta el mapa en 2D, como [11]. Es decir, cada coordenada en el mapa se convertiría en un nodo de nuestro grafo. Lógicamente las relaciones entre los nodos serían solo con los nodos adyacentes por lo que los costes serían siempre 1, y por ello el algoritmo de búsqueda del camino más corto no sería utilizado y se tendría que comprobar simplemente si al viajar de un nodo a otro estamos más cerca del nodo destino o no. Esta opción no nos ofrece muchas ventajas debido a que estaríamos trabajando directamente con el mapa 2D, por lo que no conseguimos simplificar la información.

4.2.2.3 Exploración

Consisitiría en la búsqueda de relaciones a través de la exploración del mapa con el AGV y la generación de nuevos nodos cada vez que se tuviera que hacer un segundo giro de 90º. Pese a que es una posibilidad real usada en trabajos como [10], no la hemos considerada adecuada para nuestro trabajo debido a nosotros búscamos crear el mapa topológico a partir del mapa 2D y ciertos puntos en el mismo. Si exploraramos el mapa estaríamos recolectando más datos e incumpliendo el objetivo de nuestro trabajo. Además, se ha considerado que esta opción podría llegar a ser demasiado compleja para que se pudiese tener lista en al finalización de este trabajo.

4.2.2.4 Expansión

Este es el método más original propuesto, aunque está basado en el [9]. El método de expansión se basa en que para cada nodo de interés que no tenga un camino hacia el nodo raíz se crean hasta cuatro nodos en cuatro direcciones distintas (Norte, Este, Sur, Oeste), como se ve en la Fig. 3*(a)*. Los nodos creados se van moviendo coordenada a coordenada en la dirección en la que han sido creados a través del mapa 2D hasta que se cumpla alguna de las siguientes condiciones:

1. En el área que rodea la siguiente coordenada no se puede posicionar el AGV debido a que hay un obstáculo que se lo impide.
2. En el área que rodea la siguiente coordenada hay otro nodo.
3. Desde la coordenada actual del nodo creado existe una relación directa hacia un nodo de interés que tiene un camino hacia el nodo raíz.

Cuando se cumple cualquiera de estas condiciones dejamos de mover el nodo creado y lo añadimos a los nodos de interés, indicando que su posición es la misma en la cuál se encuentra. La tercera condición es especial debido a que si se cumple no hará falta crear el resto de nodos, por lo que es imporante el orden en el que los creamos, que es: Norte, Este, Sur y Oeste. En caso de alguna de las condiciones se cumpla en la posición inicial del nodo, no crearemos dicho nodo. En la Fig. 3*(b)* se ve el resultado de crear un nodo en cada dirección, donde se aplican las dos primeras condiciones. También podemos ver que el nodo tachado no ha sido creado debido a que incumplia la primera condición en su posición inicial.

El método de expansión se ejecutará tantas veces como sea necesario hasta que el grafo esté totalmente interconectado. En la Fig. 3*(c)* podemos ver como hemos encontrado un grafo totalmente interconectado tras dos iteraciones del método, si aplicamos el método sobre uno de los nodos creamos en la primera iteración. Como se ve en la figura, solo se ha creado un nodo en la segunda iteración debido a que al crearlo hemos encontrado una relación directa, es decir, se ha cumplido la tercera condición. Es importante saber que en la Fig. 3 se ha realizado una simplificación del método debido a que solo lo aplicamos sobre un nodo en cada iteración, en realidad el método se aplica para todos los nodos sin un camino hacia el nodo raíz en cada iteración, incluyendo los nodos que han sido creados en la iteración anterior.

Las ventajas que nos aporta esté método es que siempre encontrará una solución, pero puede llegar a crear una cantidad excesiva de nodos para escenarios muy complejos, además de crear también nodos innecesarios.

4.2.3 Búsqueda del camino de coste mínimo

Las diferentes alternativas para este caso son principalmente los diferentes algoritmos ya existentens para la búsqueda del camino de coste mínimo en un grafo ponderado [3]. Las grandes diferencias entre estos algoritmos son la utilización o no de costes negativos, la utilización o no de heurísticas, y en última instancia los datos que son necesarios almacenar durante el proceso de búsqueda del camino. Debido que nosotros solo usamos costes positivos y no tenemos ninguna heurística con la que trabajar los únicos dos algoritmos que se presentan como alternativas son el de Dijkstra y el de Floyd-Marshall.

4.3 Alternativa escogida

En primer lugar, en cuanto a la generación de relaciones, debido a que solo teníamos una opción, y no hemos encontrado otra que no hiciera que complejidad del trabajo aumentará en exceso, nos quedamos con la que hemos comentado en la sección 4.2.1.

En segundo lugar, en cuanto a la generación de nuevos nodos, se ha optado por utilizar el método de Expansion. Se ha elegido esté debido a varios factores:

1. Es el método más original de entre los propuestos.
2. Nos asegura encontrar una solución, cosa que no era fácil de demostar con el método Aleatorio.
3. A pesar de que existe la posibilidad de crear un número excesivo de nodos, en ningún caso es tan alto como en el caso del *Grid*.

En tercer lugar se ha elegido Dijkstra para la generación del camino de coste mínimo. En comparación con Floyd-Marshall no nos ofrecía ventajas muy representativas, pero lo hemos elegido debido a su extensa utilización, que nos ha ayudado a la hora de guiarnos en su implementación.

5 IMPLEMENTACIÓN

5.1 Algoritmos en pseudocódigo

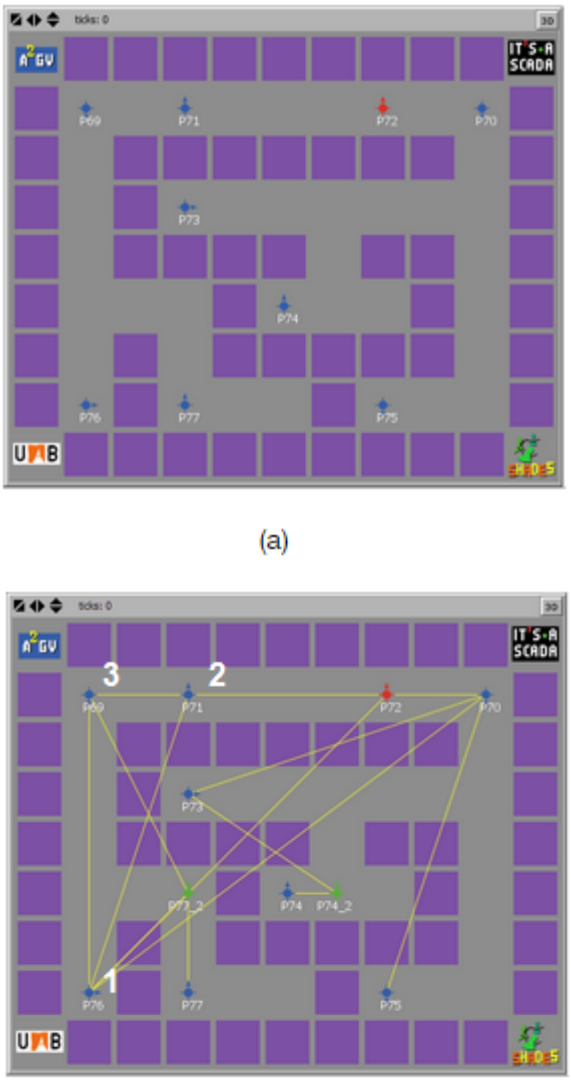
Previa a la implementación de la alternativa escogida se realizó el pseudocódigo para ambos algoritmos, el de la transformación del mapa 2D a topológico y el de la búsqueda del camino de coste mínimo, que corresponde al de Dijkstra. En la Fig.4 se incluye un resumen del pseudocódigo para el primer algoritmo, donde se refleja la utilización de las funciones principales y el orden en el que se ejecutan. No se incluye el pseudocódigo del algoritmo de Dijkstra debido a que puede ser encontrado fácilmente en internet.

```
PART1 //Comprobación de relaciones directas
   Foreach Node in NodeList:
        Look for path to NodeRoot
   If All_connected:
        Generate_Graph
        Jump END
   Else:
        Jump PART2
PART2 //Generación de nuevos nodos
        Generate_New_Nodes_with_Expansion
        Add_New_Nodes_to_NodeList
        Jump PART1
END
```

Fig. 4. Pseudocódigo de la funcionalidad implementada

5.2 Implementación inicial

La implementación se ha realizado usando *Netlogo*, un lenguaje de programación orientada a agentes, que nos ofrece una interfaz gráfica especialmente orientada para realizar simulaciones con los mismos. La razón por la que se escogío este lenguaje es porque ya teníamos un código funcional donde se simulaba el comportamiento de un AGV en dicho lenguaje.



(a)

(b)

Fig. 5. Resultados de ejecutar el algoritmo de transformación del mapa 2D en un mapa topológico sobre un escenario concreto

Dentro del código existente hemos usado funciones de mucha utilidad como son la que nos permite generar el escenario que nosotros queramos, o la que nos permite generar los puntos de interés dentro de dicho escenario. Gracias a ello, el peso de la implementación recayó en los algoritmos definidos anteriormente.

En la Fig. 5*(a)* podemos ver un escenario de ejemplo preparado con Netlogo, donde los nodos de interés están marcados con un color azul y el nodo raíz está marcado de color rojo y los cuadrados lilas representan los obstáculos. Es importante saber que el nodo raíz se escoge de forma aleatoria dentro de la lista de nodos de interés al principio de la función implementada, por lo que el mapa topológico resultante puede variar entre ejecuciones.

En la Fig. 5*(b)* se ve el resultado de ejecutar el algoritmo sobre el escenario preparado, donde los nodos generados se marcan de color verde y las relaciones entre ellos de color amarillo. Además se han numerado algunos nodos para usarlos como ejemplo.

A pesar de que en la Fig. 5*(b)* las relaciones se muestren en línea recta, se tienen que imaginar tal y como han definido en la Sección 4.2.1. Además, no hemos podido mostrar los costes, pero también han sido asignados a cada relación dependiendo del movimiento del AGV.

A primera vista ya se ve un problema en este resultado y es que existen relaciones entre dos nodos que pasan a través de otros nodos, cosa que no tiene sentido para un mapa topológico. Por ejemplo, este es el caso de la relación entre *1* y *2* en la Fig. 5*(b)*, ya que el camino entre ambos pasa por el nodo *3*, y por lo tanto la relación no debería existir. Debido a la gravedad de este problema, se buscó una solución de forma previa al análisis de los casos de uso del algoritmo. Para ello fue necesario realizar un filtro sobre todas las relaciones creadas y comprobar que en el camino de un extremo a otro de la relación no existiera otro nodo. En caso de que exista un nodo, dicha relación es eliminada. Los resultados de aplicar este filtro sobre el caso de la Fig. 5 se ven en la Fig. 6, donde ahora si que se respeta la estructura de un mapa topológico.

También se ha conseguido implementar el segundo algoritmo, el algoritmo de Dijsktra, pero no se ha considerado necesario añadir ejemplos sobre sus resultados. El algoritmo sencillamente devuelve una lista con los nodos por los que tiene que pasar el AGV para ir de un nodo a otro asegurando el coste mínimo dentro mapa topológico creado.

5.3 Casos de uso

Una vez que la implementación era funcional se ha analizado el comportamiento de la misma en diferentes casos de uso, para poder acabar de refinarla.

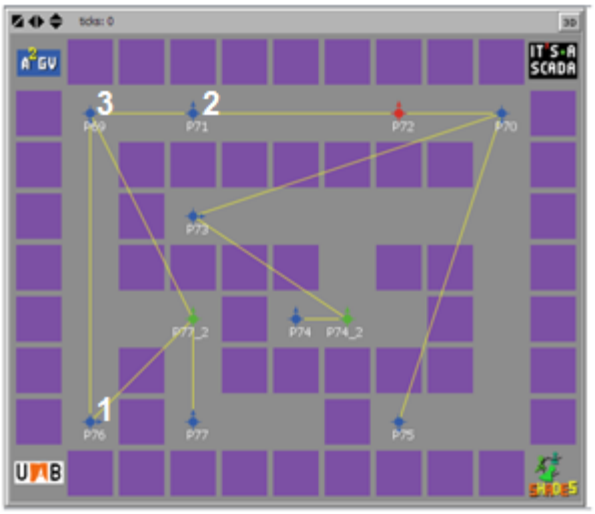


Fig. 6. Resultados de aplicar el filtro para eliminar relaciones inexistentes sobre el escenario de la Fig. 5

Los casos de uso se han preparado usando la función excistente implementada en Netlogo para generar escenarios. Dichos escenarios pueden ser vistos en la Fig. 7, donde el escenario *(a)* corresponde a un mapa abierto donde los AGV tienen mucha movilidad, el escenario *(b)* corresponde a un mapa basado en el primero pero más obstaculizado, el escenario *(c)* corresponde a un mapa formado específicamente por habitaciones, como podría ser la planta de un hospital, y el escenario *(d)* corresponde a un mapa poco realista, pero que ha sido preparado para comprobar el comportamiento de nuestro AGV en el desplazamiento por pasillos.

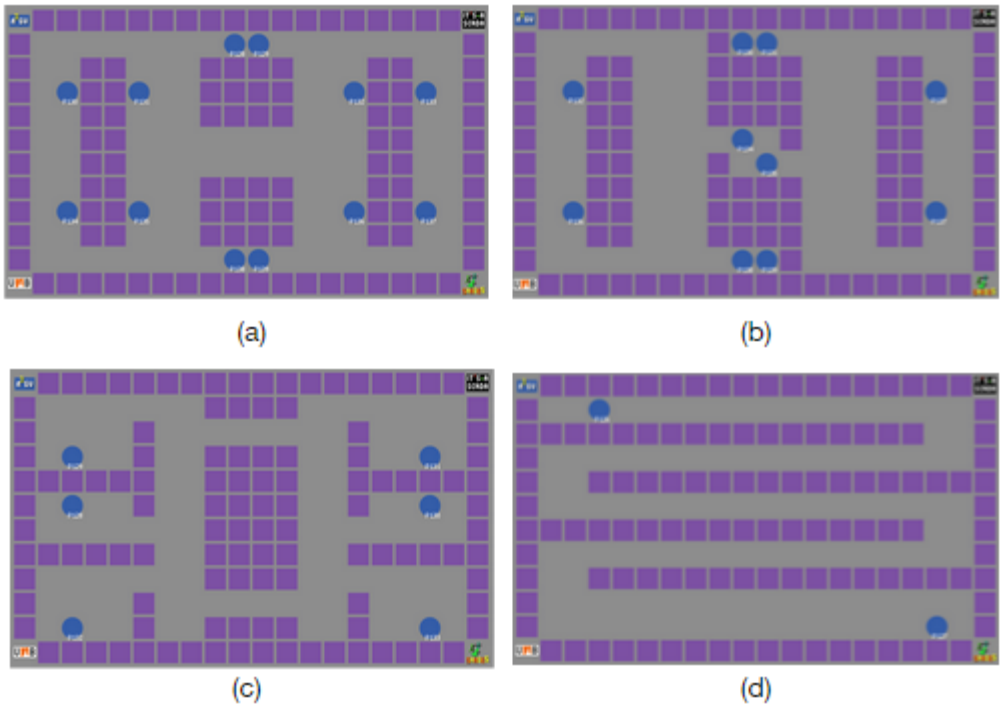


Fig. 7. *(a)* Escenario abierto. *(b)* Escenario obstaculizado. *(c)* Escenario formado por habitaciones. *(d)* Escenario formado por un pasillo.

5.4 Refinamiento de los algoritmos

Tras ver los resultados que daba la implementación inicial sobre cada uno de los casos de uso se han podido ver una serie de problemas en el mismo. Los problemas que no se han podido solucionar están incluídos en la Sección 6, por lo que este apartado es utilizado para definir aquello que si que se ha modificado.

5.4.1 Eliminación de nodos fantasma

Existen casos en los que el método de expansión generaba un nodo con una única relación, por ejemplo cuando el nodo se crea en una dirección en la que no hay ningún otro nodo. Este tipo de nodos no presentan ninguna utilidad dentro del mapa topológico debido a que nunca se viajará a los mismos. Mediante un pequeño filtro se consiguió que todos los nodos creados que tuvieran una única relación fueran eliminados.

5.4.2 Eliminación de nodos redundantes

El problema de los nodos redudantes no provoca que el mapa topológico sea incorrecto, pero si que hace que se conviertiera en algo más complejor e ineficiente de manejar, motivo por el cuál se ha considera necesario eliminarlos. Existen varias formas en las que se ha presentado este problema, cada una con su propia justificación, aunque todas están relacionadas con la forma con la que opera el método de expansión con los nodos más aislados. A continuación presentaremos cada una de ellas, a las que se les ha asignado un nombre, y la forma en la que han sido eliminadas.

5.4.2.1 Acumulación de nodos en línea

Este problema viene derivado de una de las condiciones que se presentan en el método de Expansión (Ver Sección 4.2.2.4), y es que un nodo que estamos moviendo es creado cuando "En el área que rodea la siguiente coordenada hay otro nodo. ". Esto provoca que, si durante varias iteraciones el nodo no encuentra un camino hacia el nodo raíz, se comienzen a acumular nodos en una dirección, es decir, en línea. Por ejemplo, en el resultado de utilizar el algoritmo sobre el escenario de la la Fig. 7*(b)*, que puede verse en la Fig. 8, existen nodos acumulados sin ningún uso más que ser pasos intermedios entre otros nodos. Este es el caso de los nodos de color amarillo en la figura.

Para poder eliminar los mismos lo que se ha hecho es añadir un filtro que elimina los nodos creados que se encuentran en línea recta con otros dos nodos y que solo tengan esas dos relaciones. Tras eso ha sido necesario recalcular las relaciones entre los nodos.

5.4.2.2 Acumulación de expansiones

Este problema viene derivado de que el método de expansión se aplica sobre cada nodo creado en cada iteración hasta que se encuentra que uno de esos nodos creados tiene un camino para llegar al nodo raíz y se termina el algoritmo. Si dichos nodos se encuentran en un espacio cerrado y suceden muchas iteraciones antes de que encuentren dicho camino entonces se empiezan a acumular nodos dentro de dicho espacio.

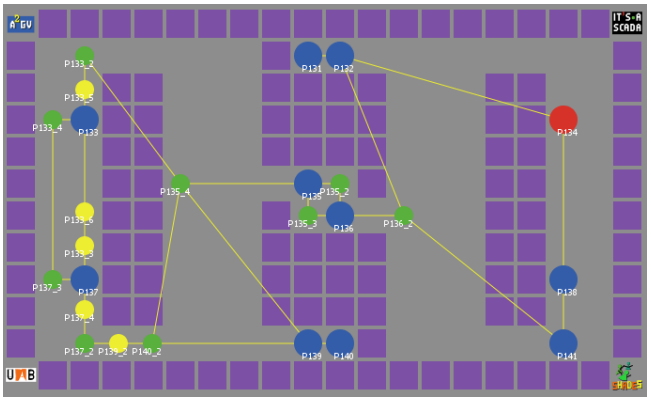


Fig. 8. Resultado de ejecutar el algoritmo sobre el escenario de la Fig. 7*(b)*.

El escenario, de entre los preparados, donde se ve este problema de forma más evidente es el de la Fig. 7*(c)*. Dicho escenario es el que peores resultados nos ha dado, y la cantidad de nodos que llega a crear solo se ha conseguido mitigar, pero no eliminar por completo. Los resultados se ven en la Fig. 9, donde de nuevo los nodos eliminados son los de color amarillo.

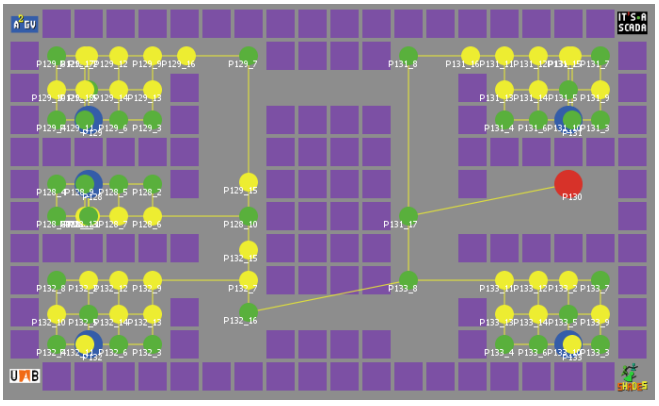


Fig. 9. Resultado de ejecutar el algoritmo sobre el escenario de la Fig. 7*(c)*.

El filtro que se ha utilizado para eliminar dichos nodos ha sido una comprobación de las relaciones de los nodos creados. Si un nodo creado tiene 3 o 4 relaciones con otros nodos creados en línea recta, entonces dicho nodo es eliminado. La razón por la que esté filtro no consigue eliminar una gran cantidad de nodos es porque solo eliminamos los nodos con relaciones en línea recta con otros nodos creados, en cualquier otro caso que se ha comprobado que la eliminación de otros nodos podría destruir el mapa topológico eliminando nodos esenciales para su interconexión.

Gracias al refinimiento que hemos implementado se ha conseguido para un escenario tan conflictivo como el de la Fig. 7*(d)*, lleguen a eliminarse una gran cantidad de nodos redundantes, como se ve en la Fig. 10. En dicha figura también se ve como el nodo que se encontraba en la esquina de arriba-izquierda ha sido eliminado aplicando el filtro definido en la Sección 5.4.1.

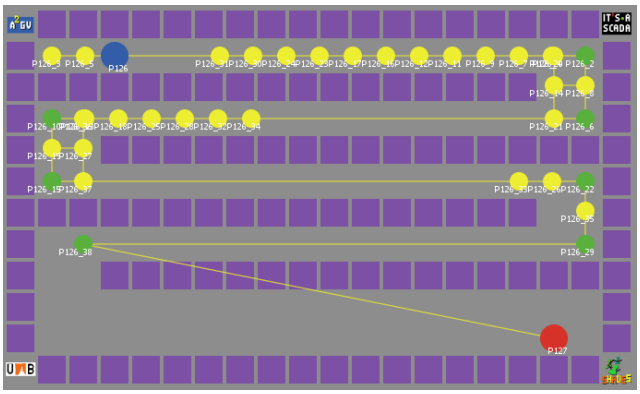


Fig. 10. Resultado de ejecutar el algoritmo sobre el escenario de la Fig. 7*(d)*.

5.4.3. Generación de todos los caminos posibles

A la hora de calcular el camino de coste mínimo entre dos nodos, se puede asegurar que el camino calculado es de coste mínimo dentro del mapa topológico creado, pero no se puede asegurar que esté sea el caso dentro del mapa 2D. Esto es debido a la forma en la que está implementado el algoritmo de transformación a mapa topológico en sí, debido a que se considera que el algoritmo ha terminado cuando se ha encontrado un camino del nodo raíz al resto de nodos, perdiendo con ello algunos caminos posibles entre el resto de nodos. La aplicación del algoritmo sobre todos los nodos, convirtiéndolos en nodo raíz en cada aplicación y generando las relaciones posteriormente, soluciona este problema. A pesar de ello durante la simulación no se ha realizado dicho cambio para facilitar con ello la comprensión de los resultados del algoritmo.

6 Resultados

Los algoritmos implementados han dado buenos resultados en las pruebas que se han realizado, aunque el de transformación de mapas 2D a mapas topológicos sufre algunos problemas leves, mientras que el de Dijkstra funciona sin ningún problema.

Entre los problemas del primer algoritmo nos encontramos en primer lugar con el que ya se comentó en la Sección 5.4.2.2, y es que la presencia de nodos redundantes en el grafo se ha podido minimizar, pero no eliminar.

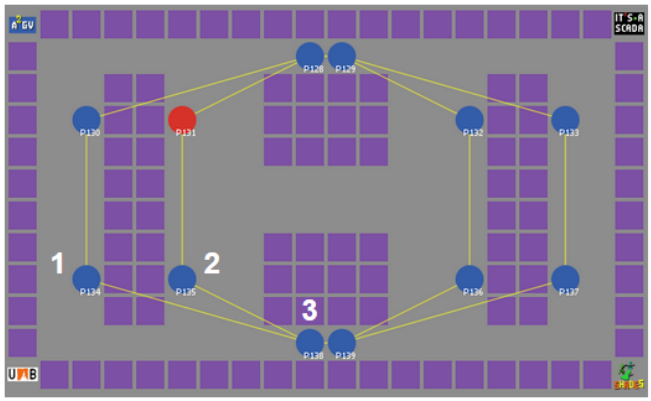


Fig. 11. Resultado de ejecutar el algoritmo sobre el escenario de la Fig. 7*(a)*.

Hay otros dos problemas que no se han podido solucionar que pueden ser entendidos a través de la Fig. 11, que presenta los resultados del algoritmo sobre el escenario de la Fig. 7*(a)*. Como se ve en la figura, en dicho caso no ha sido necesario crear ningún nuevo nodo debido a que los nodos de interés ya tenían una relación directa entre ellos. En la figura se han marcado algunos nodos con números por tal de tomarlos como ejemplo.

El primer problema es que existen relaciones entre nodos que recorren los mismos caminos, cosa que podría dificultar la planificación de rutas cuando dos AGVs operan en el mismo mapa. Este problema por ejemplo se ve en las caminos de *1* a *3* y de *2* a *3*, en la Fig.11.

El segundo problema es que el mapa topológico puede provocar que haya caminos poco eficientes, debido a que primero alejan al AGV de su objetivo, para después acercarlo de nuevo. Este problema por ejemplo se ve en el camino de *1* a *2*, en la Fig. 11.

7 Conclusión

En este documento hemos visto el desarrollo y resultados de la implementación de un algoritmo para la planificación de rutas en mapas geométricos. Dicho algoritmo ha sido dividido en dos independientes. En primer lugar, el algoritmo de transformación del mapa 2D en un mapa topológico, que nos ha ayudado a simplificar los datos hacia un grafo ponderado. En segundo lugar, el algoritmo para generar la ruta de coste mínimo entre los nodos del grafo ponderado. Tras un análisis de alternativas se ha implementado el primer algoritmo mediante un método propio, mientras que la implementación del segundo ha consisitido en la aplicación del algoritmo de Dijkstra.

Los resultados de la implementación han sido correctos funcionalmente para todos los escenarios que se han probado, pero se han encontrado algunos errores de eficiencia, como la creación redudante de nodos, y dos problemas, que son por un lado que las relaciones entre nodos pueden compartir los mismos caminos y que existen caminos entre nodos que provocan un recorrido inproductivo del AGV.

En un futuro sería necesario buscar la forma de solucionar los problemas mencionados del algoritmo, por tal de que sea posible que sea usado en casos reales.

A pesar de los problemas mencionados, el algoritmo podría ser usado actualmente como un paso previo a un proceso manual de introducción de nodos dentro de un mapa topológico. Ahorrando con ello una cantidad de trabajo equivalente al tamaño y la complejidad de los escenarios sobre los que se aplicara el mismo.

Agradecimientos

Agradecimientos a Lluís Ribas Xirgo por toda la ayuda y soporte que me ha dado a lo largo del trabajo.

BIBLIOGRAFIA

[1] Wikipedia, La enciclopedia libre, "Vehículo de guiado automático", 25 abril 2015, http://es.wikipedia.org/wiki/Problema_del_camino_m%C3%A1s_corto (Consultado: 13 de Junio del 2015)

[2] Thomas Davich, "Material Handing Solutions: A look into Automated Robotics", Department of Industrial and Systems Engineering, University of Wisconsin-Madison, 9 January 2010

[3] Wikipedia, La enciclopedia libre, "Problema del camino más corto", 29 mayo 2015, http://es.wikipedia.org/wiki/Problema_del_camino_m%C3%A1s_corto (Consultado: 13 de Junio del 2015)

[4] Wikipedia, The Free Encyclopedia, "Simulataneous Localization and Mapping (SLAM)", 11 junio 2013, https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping (Consultado: 13 de Junio del 2015)

[5] José Alberto Arcos Sánchez, Sistema de navegación y modelado del entorno para un robot móvil, 2009, p.25

[6] Butdee, Suthep and Suebsomran, Anan and Vignot, Frederic and Yarlagadda, Prasad K.D.V. (2008), "Control and path prediction of an automate guided vehicle.", In: 9th Global Congress on Manufacturing and Management, 12-14, November, 2008, Gold Coast, Austrila.

[7] E. Jung, et al., "Implementation of Laser Navigation System using Particle Filter", International Conference on Control, Automation and Systems, pp. 1636-1638, 2011.

[8] Choi, J., Choi, M., Nam, S.Y., Chung, W.K., "Autonomous topological modeling of a home environment and topological localization using a sonar grid map.", Autonomous Robots 30(4) (2011) 351-368

[9] Kuipers, B. and Byun, Y.T., "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations", Robotics and Autonomous System, 8 (1991) 47-63.

[10] Mataric, M. J. (1992). "Integration of representation into goal-driven behavior-based robots." IEEETransactions on Robotics and Automation, 8(3), 304-312.

[11] S. Thrun and A. Biicken, "Integrating grid-based and topological maps for mobile robot navigation.", In Proceedings of the Thirteenth National Conference on Artificial Intelligence, pages 944-950, 1996.

[12] Shatkay, H., & Kaelbling, L. P. (1997). "Learning topological maps with weak local odometric information." Paper presented at the IJCAI International Joint Conference on Artificial Intelligence, 2 920-927.

[13] Duckett, T. ; Dept. of Comput. Sci., Manchester Univ., UK ; Nehmzow, U. ,"Exploration of unknown environments using a compass, topological map and neural network.", Computational Intelligence in Robotics and Automation, 1999. CIRA '99. Proceedings. 1999 IEEE International Symposium on 1999,312-317. doi: 10.1109/CIRA.1999.810067