Memoria práctica 1 de Tecnología de los Sistemas Inteligentes:

UNIVERSIDAD DE GRANADA

Antonio Manuel Fresneda Rodriguez
antoniomfr@correo.ugr.es
77447672-W

Índice

Resumen

En el código proporcionado tenia una implementación de una búsqueda en anchura. Esto provoca que a la hora de trazar un camino hacia un punto se expandan muchos nodos y no sea eficiente.

Para mejorar el comportamiento del algoritmo, hemos sustituido la búsqueda en anchura por el algoritmo A* ya que el código que se nos proporcionaba necesitaba un par de modificaciones para cambiar la búsqueda en anchura por el A*.

Modificaciones

A*

El código proporcionado usaba dos listas para la gestión de abiertos y de cerrados. Ninguna de estas listas seguía ningún orden, así que procedí a ordenar la lista de abiertos en función de f para así poder acceder al elemento con mejor f de una forma eficiente. Para ello modifique la función myAstarPlaner::addNeighborsToOpenList para que se calculase la g de la celda (calculada como la suma del coste de g del padre y la distancia entre el padre y el nodo actual), la h (distancia euclidea desde el punto actual al objetivo) y f (la suma de ambas). Una vez echo esto, ordenamos la lista cada vez que introducimos un nodo según la f.

Otra parte que diferencia el código proporcionado al A* es que si durante la ejecución del algoritmo encontramos una posición del mapa que ya está en abiertos, tenemos que comprobar el valor de g, y en el caso de que la celda encontrada sea mejor que la celda en abiertos sustituirla. Para ello, he cambiado la funcion myAstarPlanner::isContains para que devuelva un iterador al elemento de cerrados y así poder borrarla en caso de que sea necesario.

También he tenido en cuenta el footPrint del robot. Este parámetro es un poco difícil de ajustar, ya que si consideramos un valor del footPrint de una casilla demasiado alto no traza un plan seguro mientras que si lo ponemos demasiado bajo el robot no es capaz de encontrar un camino para ningún sitio. Depende mucho de la posición inicial del robot. Yo la he dejado en 254, ya que si pongo un valor menor en la posición inicial de donde parte mi robot no es capaz de salir.

Mejora en la estructuras de datos

Para mejorar un poco la eficiencia en tiempo de la gestión de abiertos, cambié la lista por un multiset (ya que en un multiset podemos repetir elementos, y como se van a ordenar en función de f, puede haber varios nodos que tengan el mismo valor de f) ordenado en función de f (con la clase Functor). Con esto obtenemos una buena mejora, ya que no ordenamos constantemente la lista (seria de orden NLOGN siendo N el tamaño de la lista) y la inserción en un set es logarítmica respecto al tamaño. Obtener la celda con menor f es de un orden constante.

La siguiente mejora pasa por mejorar cerrados. En cada iteración debemos ver si el nodo actual esta en cerrados. Si en cerrados no la tenemos ordenada este orden es lineal, así que decidí cambiar la lista por un unordered_map, donde la consulta de un elemento es de orden constante. Esto provoco una mejora en tiempo pero no en nodos expandidos. Por defecto, no se usa el compilador de C++11 que es necesario para usar unordered_map, asi que he cambiado el CMakeList.txt para que se use.

Solo he subido a la plataforma el código con la implementación de abiertos con multiset y cerrados con unordered_map, pero en la sección de tiempos reflejaré la diferencia en tiempo usando tanto la primera implementación como la segunda.

Mejoras de A*

He implementado dos mejoras para el algoritmo:

1. Poda a priori: Esta "mejora" consiste en no tener en cuenta los nodos menos prometedores. Para nuestro caso, cada celda tiene como mucho ocho hijos (4 diagonales, 2 verticales y 2 horizontales). Cuando tenemos mas de 6 hijos, eliminamos 3 de ellos. La condición de que el numero de hijos generados sea 6 es debido a que puede existir el caso en el que el robot tenga muchos obstáculos o casillas no validas a su alrededor y en ese caso prefiero expandir esos nodos para intentar salir antes del obstáculo. Esta mejora se traduce en un menor numero de nodos que expandimos.
2. Pesos para la heurística: Esta mejora es muy buena respecto al tiempo, ya que dándole mas peso a la heurística el algoritmo converge hacia la solución mas rápido. Los pesos que he asignado son 0,65 para el coste de h y 0,35 para el coste de g.

Cabe destacar que estas mejoras provocan que el algoritmo pierda su optimalidad, ya que modificamos la función con pesos y expandimos menos nodos. Debido a esto, he decidido eliminar la parte de actualización de los padres, ya que si lo dejara para abiertos tendría también que actualizar cerrados y aquí el algoritmo perdería mucho tiempo, y comparando las distancias entre los algoritmos no óptimos y el óptimo, la diferencia no es muy grande.

Otras mejoras que he implementado pero he decidido eliminarlas han sido la poda a posteriori y la implementación de unos pesos dinámicos. La poda a posteriori, que se basa en eliminar los nodos menos prometedores una vez que la lista de abiertos llega a un tamaño, en mi caso no mejoraba mucho la ejecución del algoritmo y la diferencia entre pesos dinámicos y pesos estáticos es muy baja en los experimentos que yo he realizado.

Las mejoras en el apartado de estructuras de datos siguen estando implementadas en la versión del A* mejorado.

Datos tras los experimentos

A continuación, se muestra una tabla con los datos obtenidos en los distintos puntos de objetivos (estan especificados en el archivo Goals.txt junto con el comando para publicarlo en el nodo move_base_simple/goal).

Tablas

Punto 1

Algoritmo usado	Tiempo de ejecución(s)	Nodos Abiertos	Nodos cerrados	Distancia total del plan	Numero de nodos explorados
A* Listas	5.00	6071	3930	9.688239	3929
A* unordered	3.30	6071	3930	9.688239	3929
A* Mejorado	0.000000	631	164	9.966055	163

Punto 2

Algoritmo usado	Tiempo de ejecución	Nodos Abiertos	Nodos cerrados	Distancia total del plan	Numero de nodos explorados
A* Listas	7.80	7536	4907	25.502503	4906
A* unordered	5.10	7536	4907	25.502503	4906
A* Mejorado	1.90	3418	3001	27,365253	3000

Punto 3

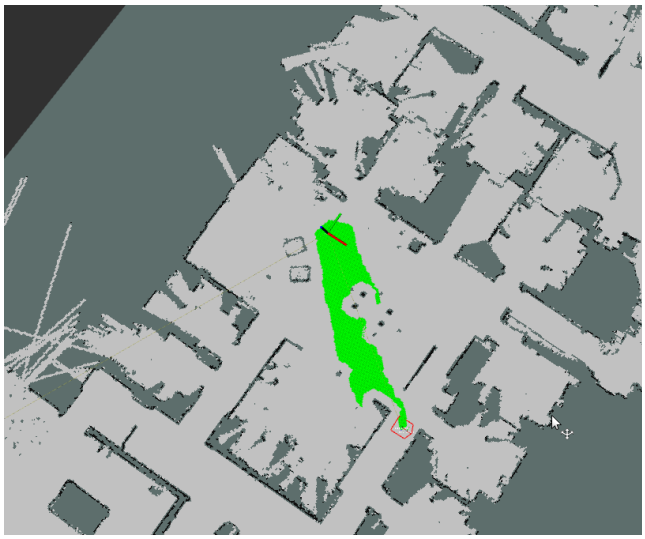
Algoritmo usado	Tiempo de ejecución	Nodos Abiertos	Nodos cerrados	Distancia total del plan	Numero de nodos explorados
A* Listas	18.10	12838	6553	34.455699	6552
A* unordered	11.80	12838	6553	34.455699	6552
A* Mejorado	0.10	960	341	34.455701	340

Coordenadas de los puntos. (No se han cambiado las posiciones de salida).

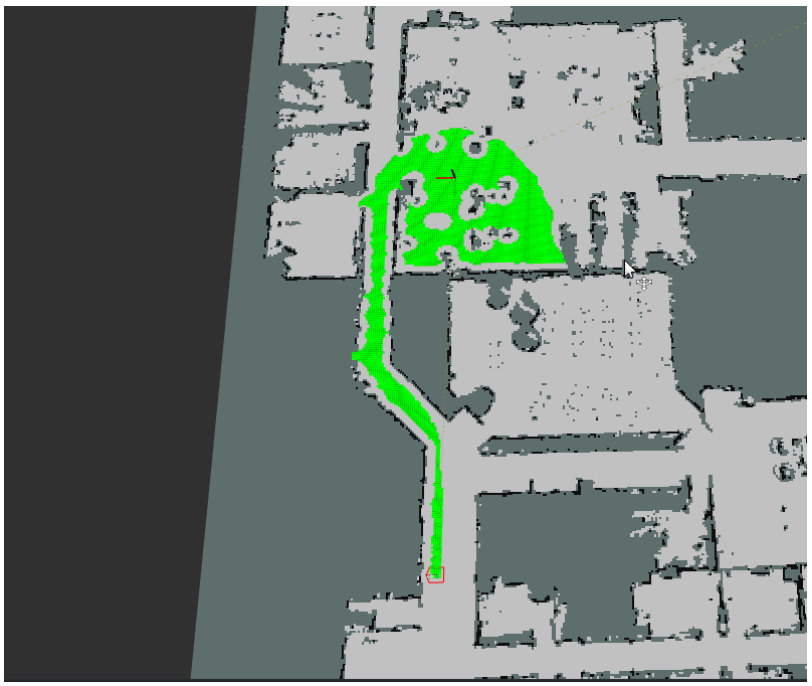
Nombre del punto	Mapa	Coordenada X	Coordenada Y
Punto 1	Willow Garage 5cm	23.4037132263	20.8539066315
Punto 2	Willow Garage 10cm	47.4943313599	41.6638870239
Punto 3	Willow Garage 10cm	13.9772024155	22.9696788788

Capturas de pantalla

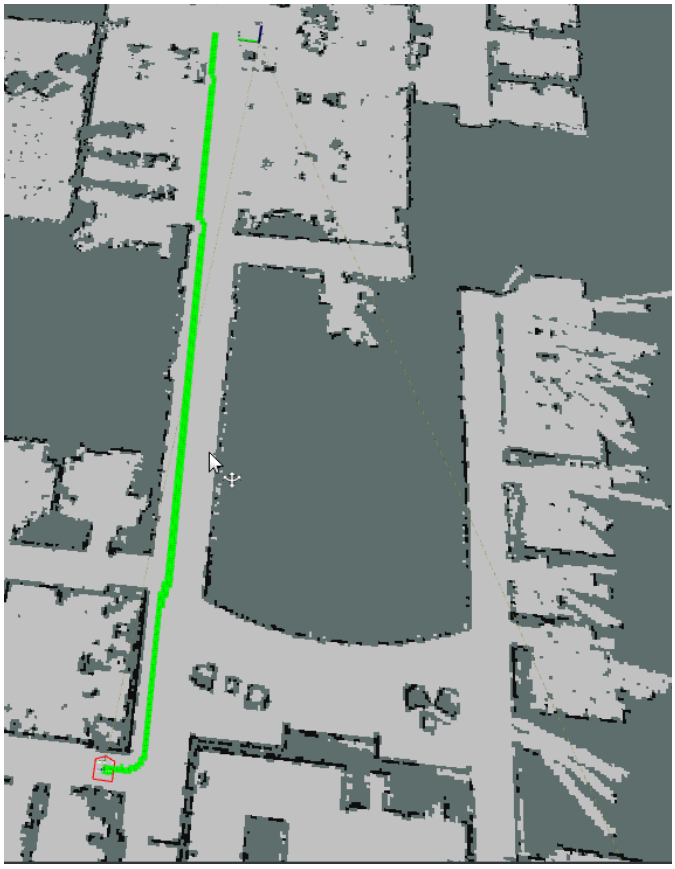
Punto1

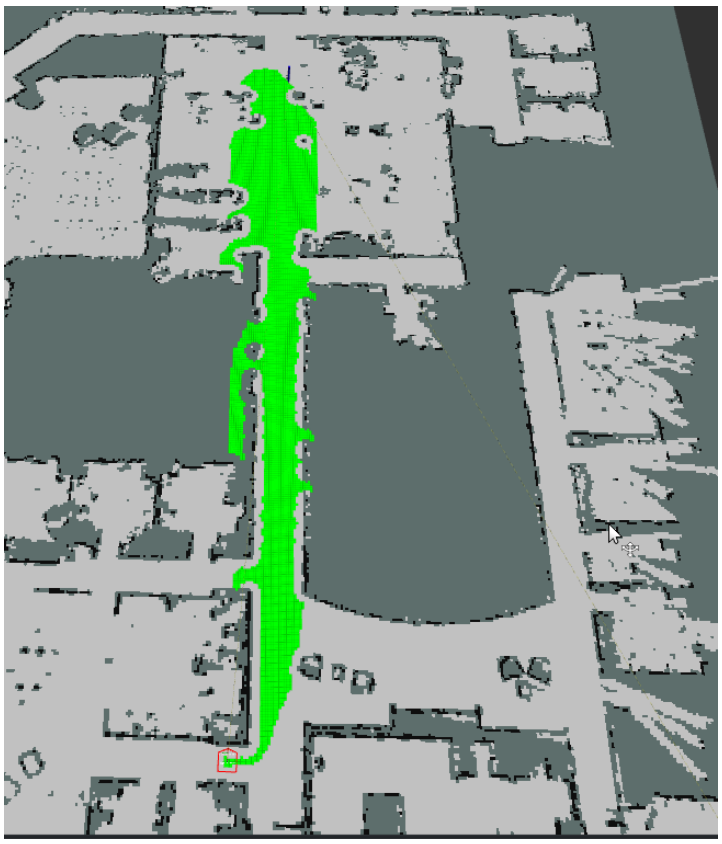


Punto2



Punto3





En esta imagen vemos como el mapa se mete por un sitio que no debería meterse. Esto se debe a un fallo que hay en el mapa en ese punto