Práctica 2:
Planificación de caminos en Robótica

Antonio Manuel Milán Jiménez

Grupo 3

Resumen

Para esta práctica se nos pide implementar una búsqueda A* para ROS utilizando como base el código de “myAstarPlanner.cpp” proporcionado.

Así, primero modificamos el código que actualmente implementa la búsqueda en anchura por el algoritmo A*. Este algoritmo tiene que implementarse con la excepción de que no sólo busque el camino más corto sino también, el más seguro, controlando de que no elija un camino demasiado cerca a los obstáculos y termine chocando.

A continuación, tenemos que modificar la implementación que acabamos de hacer para buscar una implementación más rápida en la búsqueda, aunque no encuentre la solución óptima. Entonces, implementaremos una modificación que consiste en tener un peso que controle la importancia de la heurística, conforme más nos estemos acercando a la solución, menos importante será la heurística.

Finalmente, realizaremos varios experimentos con el algoritmo implementado y su modificación, y compararemos características cómo el tiempo de ejecución, los nodos expandidos y la longitud del camino encontrado.

Implementación de A*

La primera parte de la práctica consiste en la implementación del algoritmo A*. Utilizamos como base el código proporcionado en “myAstarPlanner.cpp” que tiene implementado la búsqueda en anchura y le realizamos las siguientes modificaciones:

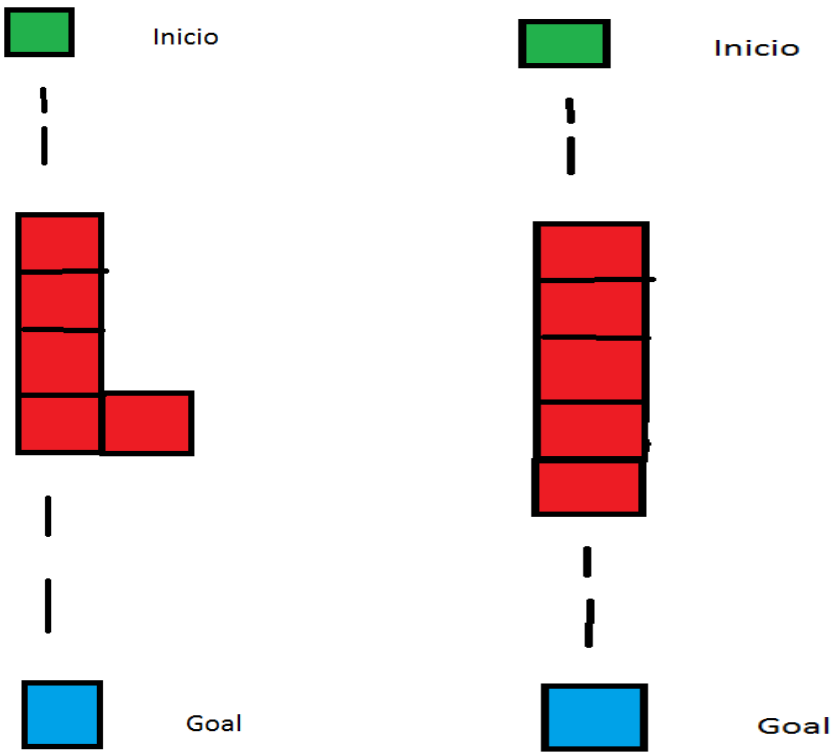
- Anteriormente se escogía el primer nodo de la lista de abiertos; ahora escogemos el nodo con mejor valoración “f” de la lista de abiertos. Ese será el que expandamos.
- Ahora por cada vecino que generamos calculamos su “f” y lo añadimos a la lista de abiertos si no estaba ni en la lista de abiertos ni de cerrados. En el caso de que ya estuviese en una de las lista, comprobamos si su “f” es mejor que la del nodo que ya estaba en la lista, en cuyo caso, actualizamos “g”, “f” y el padre del nodo de la lista.
- Respecto a los vecinos que estamos generando, dado que tenemos que encontrar un camino también seguro, por cada vecino del nodo actual comprobamos si sus vecinos en un radio de [-2,2] no son obstáculos. Si alguno de ellos sí es un obstáculo, no podemos añadir el vecino a la lista de abiertos pues podría no ser un camino seguro. Conforme más aumentemos el radio, más seguro será el camino, aunque esto puede llevar a que haya sitios por los que no puede pasar ya que el robot considera que no son suficientemente seguros.

Mejora del algoritmo A*

Para la segunda tarea, se nos pide que implementemos una mejora en el A* que permite encontrar el camino en menos tiempo, aunque esto pueda llevar a que no siempre encuentre la solución óptima.

La mejora implementada se conoce como “Dynamic weighting”. Consiste en que el valor de la heurística lleva asociado un peso que determinará la importancia de la heurística en el valor “f”. Así, conforme más nos acercamos al nodo objetivo, más se reduce este peso. Esto hace que conforme más nos acerquemos a la solución, le estemos dando más importancia al coste “g” y menos importancia al valor de la heurística.

¿Y cómo influye esto en reducir el tiempo de búsqueda? Como veremos más adelante, con esta mejora se reduce en gran medida el número de nodos que exploramos. Esto consigue porque al acercarnos al objetivo, le estamos dando mucha más importancia al coste del nodo actual al de inicio, que al coste desde el nodo actual al objetivo. Veamos estas dos figuras:



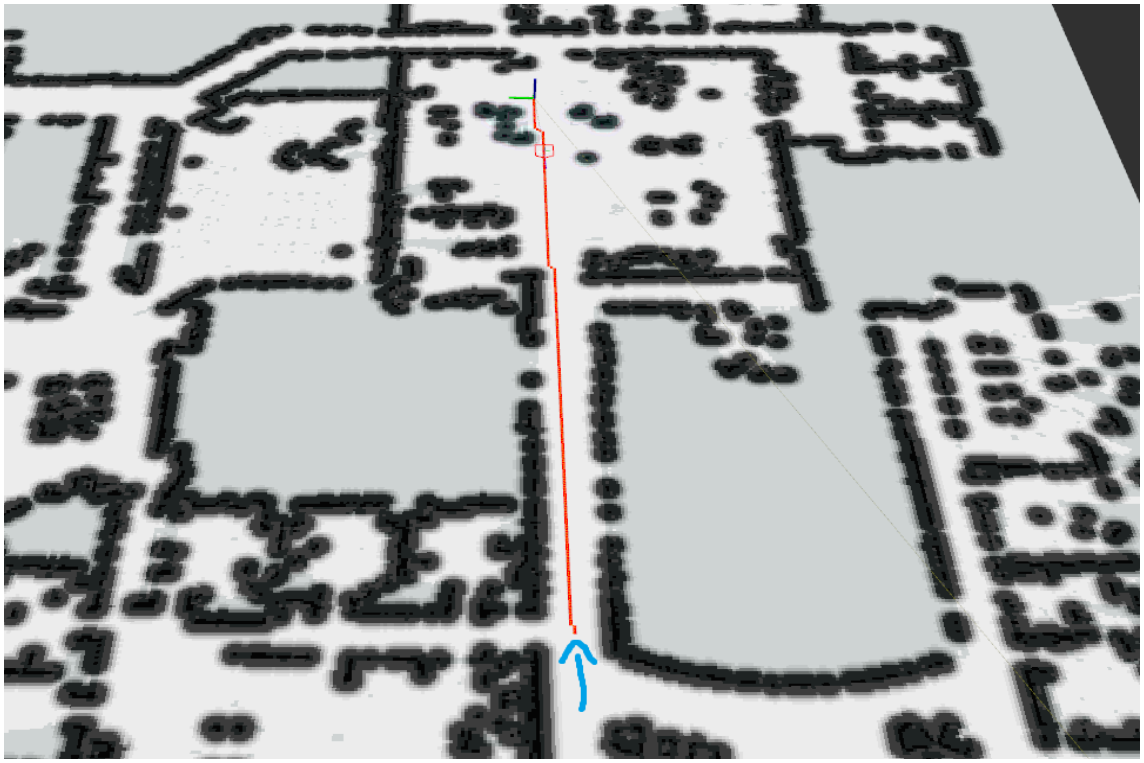
Cómo acabamos de comentar, se le vamos dando mucha más importancia al coste “g”. Esto hace que movimientos como la primera figura se penalicen mucho más de lo normal y tienda el algoritmo mucho más a movimiento cómo el de la segunda figura.

Experimentación

Hemos realizado varios experimentos con el algoritmo A* y su mejora para ver cómo funcionan y compararlos. En las imágenes, señalamos con una flecha azul el nodo objetivo.

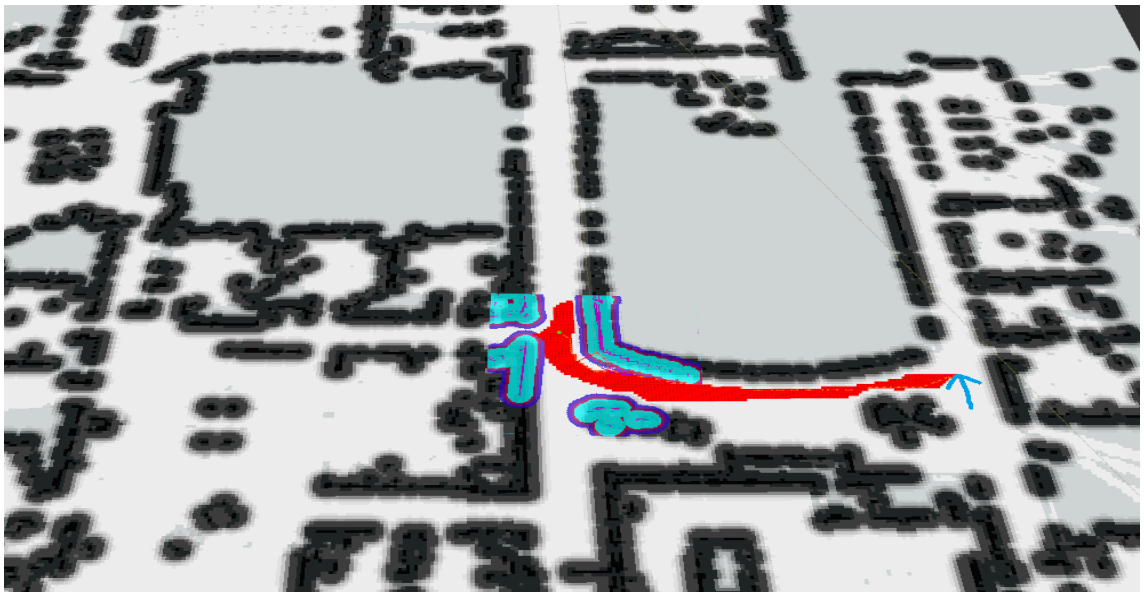
Este es el primer experimento para A*:

Y aquí tenemos el mismo experimento para la mejora:



Mientras que en el A* ha explorado 908 nodos y ha tardado 0.64 segundos en encontrar la solución, en el A* mejorado tan sólo ha tardado 0.17 segundos en dar con la solución, explorando 261 nodos. El camino obtenido es de 261 nodos, 26.34 metros, en ambos algoritmos. Esto significa que el A* mejorado ha explorado exactamente los mismos nodos que ha necesitado para construir su camino, obteniendo una solución igual de óptima que la del A*.

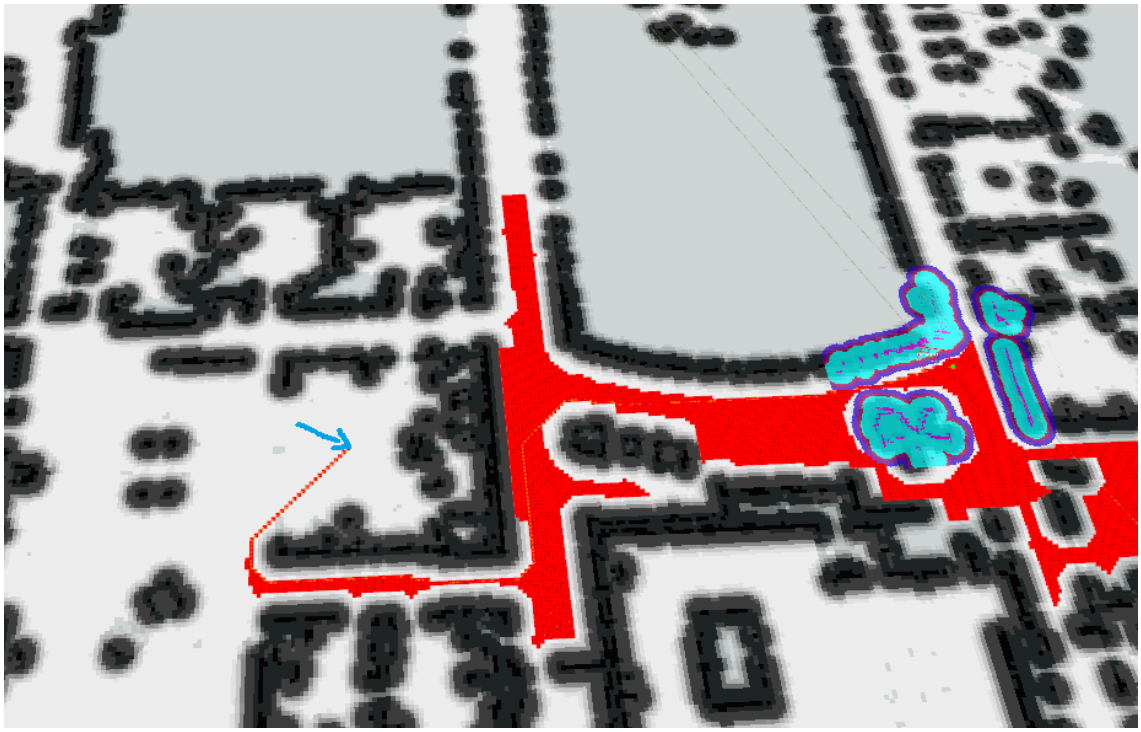
Estudiemos ahora el segundo experimento. Aquí tenemos el resultado del A*:



Y este es el resultado para el A* mejorado:

Al igual que ha pasado en el anterior experimento, el A* ha necesitado explorar más nodos, 841, y más tiempo, 0.5 segundos, en encontrar la solución que el A* mejorado con tan solo 176 nodos explorados y 0.12 segundos. Respecto a la solución, el A* ha encontrado una solución de 118 nodos, 12.75 metros, mientras que el A* mejorado la ha construido con 119 nodos, 12.89 metros. Esta vez, el A* mejorado han encontrado una solución casi óptima.

Veamos ya el tercer experimento. Aquí lo tenemos para el A*:



Y aquí para el A* mejorado:

Este experimento es más especial puesto que, si bien el A* mejorado ha llegado a la solución, luego no ha sido capaz de construir el camino solución (entra en bucle). Al parecer, este A* mejorado tiene más dificultades cuando hay muchos obstáculos entre el origen y el destino. El A* ha necesitado explorar 3946 nodos, en 5.04 segundos. La solución que ha obtenido es de 240 nodos, 26.23 metros. Viendo las imágenes, el A* mejorado ha explorado prácticamente los mismos nodos, aunque el tiempo en hacerlo ha sido mucho menor.

Como conclusión podemos decir que el A* mejorado con Dynamic weighting funciona mucho mejor que el A*, muchos menos nodos que necesita explorar y obtiene una solución prácticamente igual de buena, siempre que no haya demasiados obstáculos, ya que de lo contrario no conseguirá construir la solución. Viendo el primer experimento, funciona realmente bien cuando explora en un pasillo.