Universidad de Granada

Grado en Ingeniería Informática.

Algoritmo A*.

Nombre de la asignatura:
Técnicas de los Sistemas Inteligentes.

Realizado por:
Néstor Rodríguez Vico

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS
INFORMÁTICA Y DE TELECOMUNICACIÓN.

Granada, 8 de mayo de 2017.

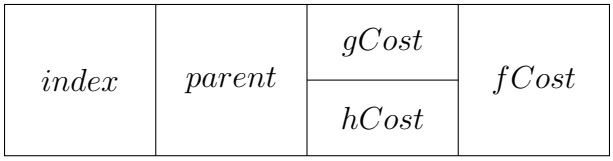
1. Resumen.

El código que se nos proporcionó tenía implementada una búsqueda de caminos para nuestro robot que no era muy eficiente, ya que expandía muchos nodos. En esta práctica he desarrollado un algoritmo A* para encontrar un camino corto y seguro de una manera más eficiente. Mi algoritmo está basado en el código que se nos proporcionó, ya que no le faltaba mucho para ser un A* funcional. Una vez implementado el algoritmo, he realizado una mejora para que se expandan menos nodos y por lo tanto sea más eficiente todavía. Los resultados (tanto de la versión estándar del algoritmo como la versión mejorada) han sido obtenido en el mapa willow_garage. A la hora de comprar como de eficiente es cada versión del algoritmo se va a considerar una versión mejor que otra si dicha versión expande menos nodos y por lo tanto emplea menos tiempo en encontrar un camino para nuestro robot.

2. Primeros pasos.

Como bien se ha comentado en el resumen, el código proporcionado ya tenía implementado un algoritmo de búsqueda de caminos. Viendo gráficamente como se expanden los nodoso en RViz podemos ver que no se hace de manera eficiente. Aunque no sea un algoritmo eficiente, tiene más o menos todos los pasos que debería tener nuestro A*, por lo tanto lo he usado como base. Dicho algoritmo de búsqueda está implementado en la función *makePlan*.

Antes de pasar a los detalles de implementación debemos tener clara la estructura que tienen los nodos procesados por nuestro algoritmo. La estructura es la siguiente:



Debemos recordar que *fCost* se calcula como *gCost* + *hCost*.

3. Diferencias.

La principal diferencia entre el algoritmo proporcionado y el algoritmo A* es la manera en la que se eligen los nodos. El primero de ellos elige el primer nodo de la lista de abiertos, mientras que el segundo elige el mejor nodo de la lista de abiertos. Para facilitar esta tarea he cambiado la estructura que se usa para almacenar los nodos, cambiando la lista (*list*) por una estructura que esté ordenada como es un *set* con un functor para ordenar los nodos en función del coste *f*, de tal manera que el nodo de menor *f* sea siempre el primero y por lo tanto, sea el seleccionado. Nuestro *set* tiene la siguiente estructura:

$openList =$

n_1	n_2	...	n_{i-1}	n_i

Y cumple la siguiente condición:

$$\forall i \;\; fCost \text{ de } n_i \; < \; fCost \text{ de } n_{i+1}$$

La siguiente diferencia más notable entre los dos algoritmos son los costes de los nodos. El algoritmo original no asignaba ningún valor a la g del nodo, y por lo tanto la f es igual a la h. Nuestro algoritmo A* cambia esto. Tal y como hemos visto en teoría asigna a la g de cada nodo el coste de ir desde el nodo inicial al nodo que estamos procesando. De esta manera tenemos una f distinta a h y formada por la suma de g más h.

La siguiente diferencia que podemos ver es que el algoritmo original no realiza ningún tratamiento en el caso de que se encuentre un mejor padre para un nodo. Como vimos en teoría, el algoritmo A* si realiza esta tarea, así que se ha añadido una función llamada *propagarInformacion* que se encarga de dicha tarea.

Finalmente, debemos asegurarnos que el camino que va a seguir nuestro robot sea seguro. Para ello he modificado la función llamada *footPrintCost* para que si algún nodo dentro de la *footprint* del robot sea un obstáculo, el coste sea elevado y no se añada a la lista de abiertos como posible vecino a explorar, ya que no nos interesan los nodos que sean obstáculos o cercanos a ellos por no ser seguros para nuestro robot.

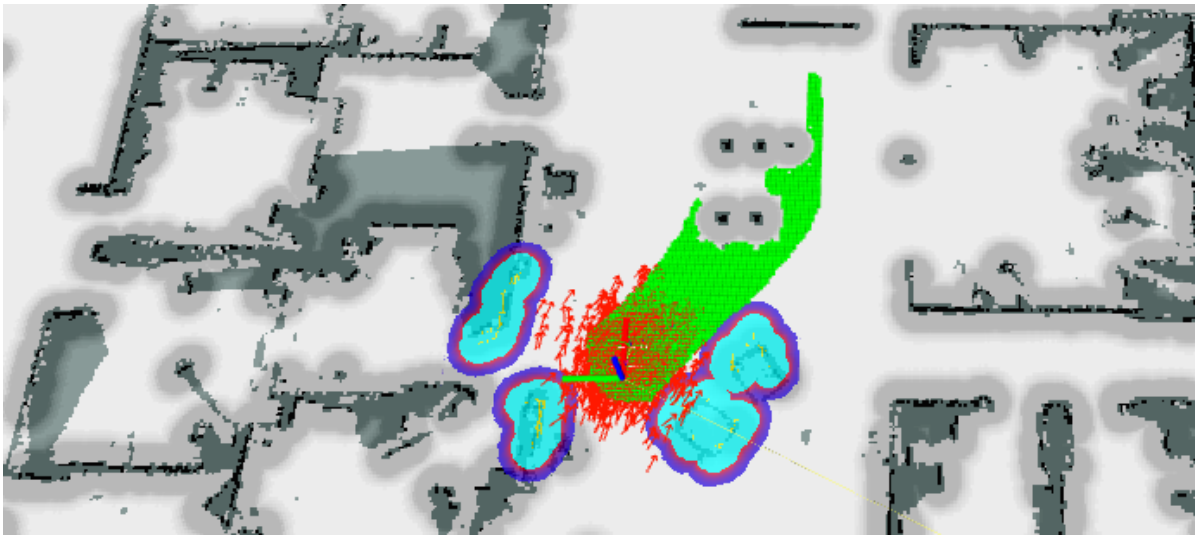
A parte de las diferencias ya comentadas, no se ha modificado nada más del código, exceptuando pequeños detalles (y que no afectan al algoritmo) como puede ser el mostrar información por pantalla o el calculo de tiempos de ejecución.

4. Mejora del algoritmo y experimentación.

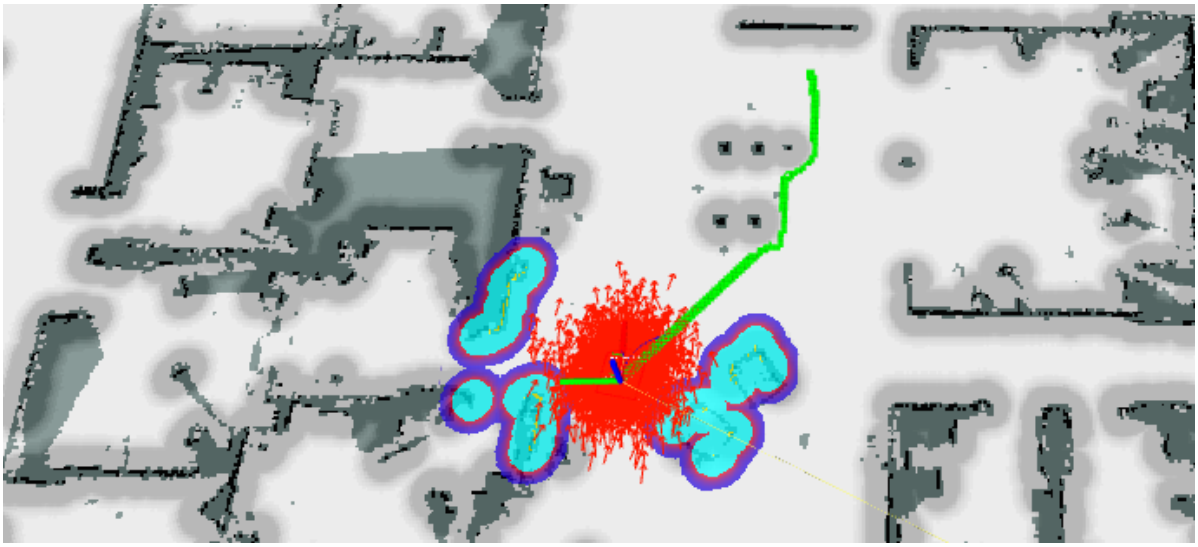
Anteriormente hemos comentado que el coste f de un nodo se calcula como la suma del coste g más el coste h, pero en teoría vimos que se pueden ponderar estos dos pesos para darle más importancia a uno de los dos. Por lo tanto, la nueva manera de calcular $fCost$ sería $fCost = \alpha gCost + (1 - \alpha)hCost$. En función del valor de α obtenemos un algoritmo u otro:

- Para $\alpha = 0 \rightarrow fCost = hCost \rightarrow$ Algoritmo Greedy con primero el mejor.
- Para $\alpha = 1 \rightarrow fCost = gCost \rightarrow$ Algoritmo con costo uniforme.
- Para $\alpha \neq 1$ y $\alpha \neq 1 \rightarrow fCost = \alpha gCost + (1 - \alpha)hCost \rightarrow$ Algoritmo A*.

Pero, ¿que comportamiento obtiene el algoritmo A* en función del valor de α? Ejecutamos nuestro algoritmo sobre el mapa *willow_garage* con $\alpha = 1$ y obtenemos el siguiente resultado:



Ejecutamos nuestro algoritmo con $\alpha = 0{,}3$ y obtenemos el siguiente resultado:

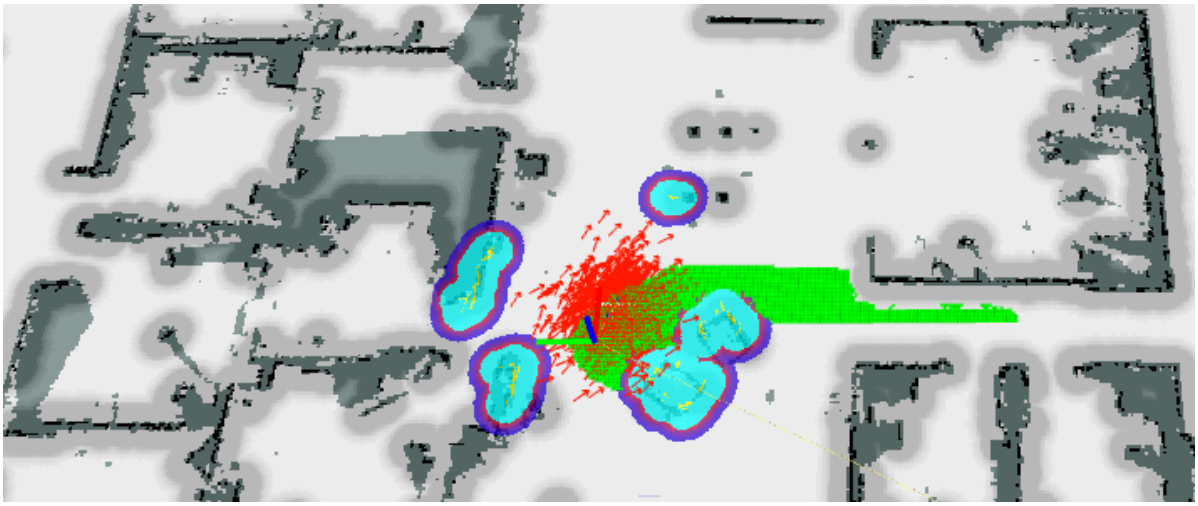


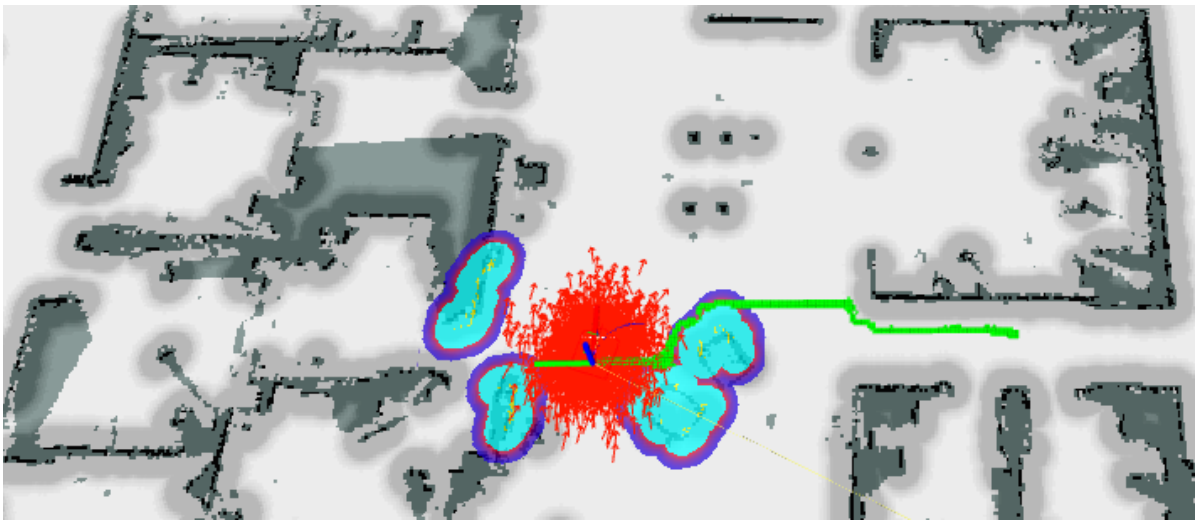
Si usamos $\alpha = 1$ se exploran 2476 nodos en 2.1 segundos, obteniendo un plan con 115 nodos y 6.264982 metros, mientras que si usamos $\alpha = 0{,}3$ solamente se exploran 121 nodos en 0 segundos, obteniendo un plan con 118 nodos y 6.352952 metros. [1]

Como podemos ver, con un valor de α cercano a 0 aumentamos la importancia de *hCost*, lo que nos lleva a obtener un camino más rápido, ya que el algoritmo le da más importancia a la distancia al objetivo.

[1] La diferencia del tamaño del plan se debe a que, a la hora de repetir el experimento con un valor de α distinto, no tengo una precisión perfecta como para enviar al robot al mismo sitio exactamente que en la vez anterior, sino que lo mando a un sitio aproximado.

He realizado un par de pruebas más. He reiniciado el robot y lo he llevado al mismo punto pero en dos pasos. Veamos los resultados. Primer paso: la primera imagen es para $\alpha = 1$ y la segunda para $\alpha = 0{,}3$:





Si usamos $\alpha = 1$ se exploran 2170 nodos en 1.7 segundos, obteniendo un plan con 148 nodos y 7.360876 metros, mientras que si usamos $\alpha = 0{,}3$ solamente se exploran 197 nodos en 0.1 segundos, obteniendo un plan con 163 nodos y 7.363763 metros. Veamos que sucede en el segundo paso. La primera imagen es para $\alpha = 1$ y la segunda para $\alpha = 0{,}3$:

Si usamos $\alpha = 1$ se exploran 6395 nodos en 14.3 segundos, obteniendo un plan con 236 nodos y 9.973089 metros, mientras que si usamos $\alpha = 0{,}3$ solamente se exploran 2412 nodos en 2 segundos, obteniendo un plan con 248 nodos y 10.417773 metros.

5. Experimentación.

Como parte final de la practica he subido un pequeño video a YouTube donde se muestra el comportamiento del robot. El enlace es el siguiente: A* - TSI