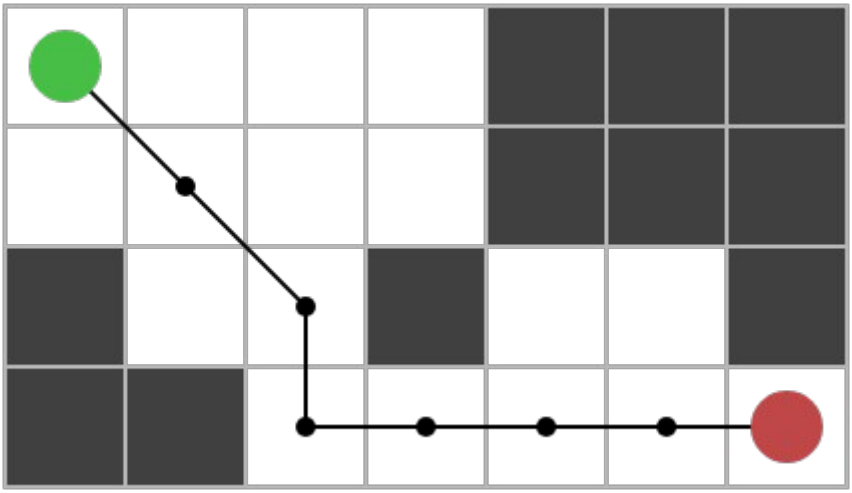
Entrega 2

Algoritmo A*



Jorge Soler Padial

Introducción

En esta practica se pide la implementación de A*, un algoritmo para encontrar “path's” o rutas entre dos puntos de forma óptima. Implementar el algoritmo como un plugin para ROS que hara la función global planner del paquete navigation. La función de global planner es la de calcular rutas entre dos posiciones, normalmente la posción del robot y una posición de destino.

Para ello he implementado el algoritmo siguiendo el pseudo-codigo que lo describe:

```
function A*(start, goal)
  open_list = set containing start
  closed_list = empty set
  start.g = 0
  start.f = start.g + heuristic(start, goal)
  while open_list is not empty
    current = open_list element with lowest f cost
    if current = goal
      return construct_path(goal) // path found
    remove current from open_list
    add current to closed_list
    for each neighbor in neighbors(current)
      if neighbor not in closed_list
        neighbor.f = neighbor.g + heuristic(neighbor, goal)
        if neighbor is not in open_list
          add neighbor to open_list
        else
          openneighbor = neighbor in open_list
          if neighbor.g < openneighbor.g
            openneighbor.g = neighbor.g
            openneighbor.parent = neighbor.parent
  return false // no path exists
```

Para ello usamos dos listas, una de nodos abiertos, por los que continuamos nuestra búsqueda, y otra de nodos cerrados, nodos que ya hemos explorado.

La función heurística que usamos es la distancia euclídea entre los puntos, mejor que la de manhattan ya que aunque estemos trabajando sobre un grid este representa un mapa real a cierta resolución.

A la hora de explorar los vecinos de un nodo, solo consideramos que un nodo es accesible por el robot si su valoración en costmap es menor de 127.

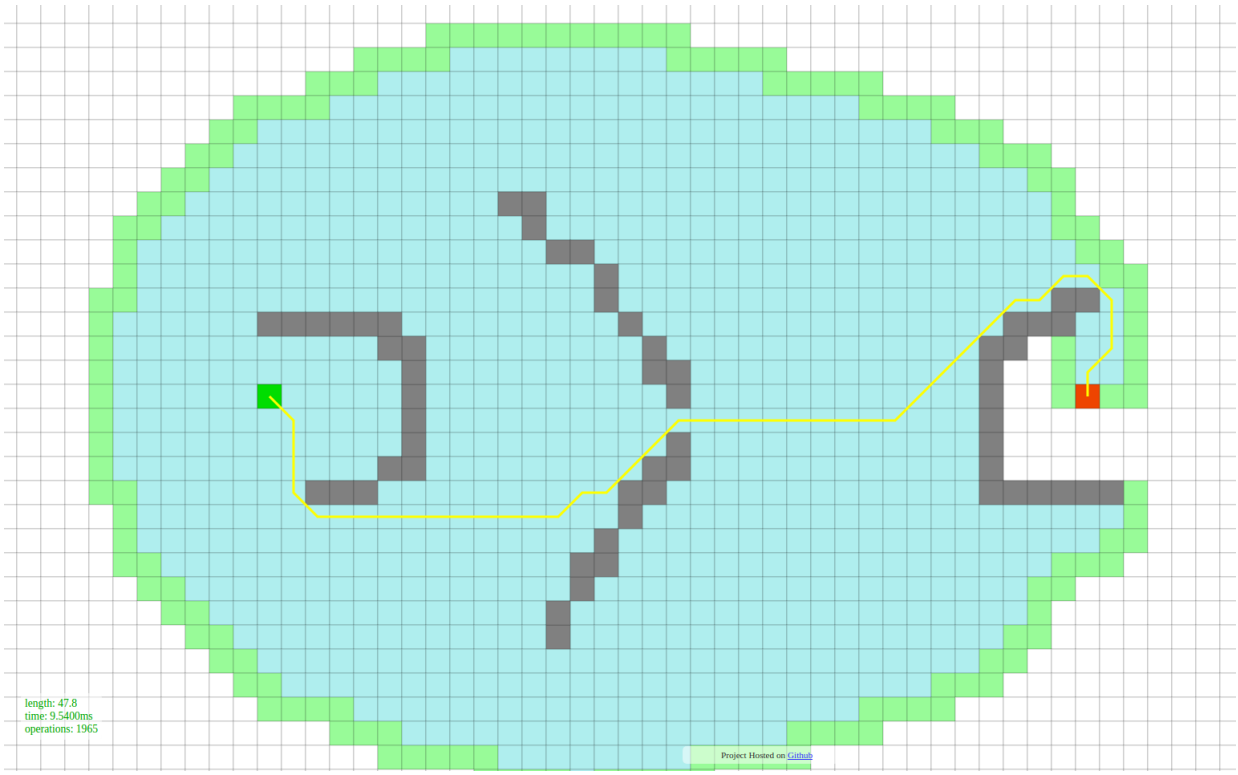
Mejora JPS

Para intentar mejorar los resultados del algoritmo A* he usado un algoritmo llamado JPS.

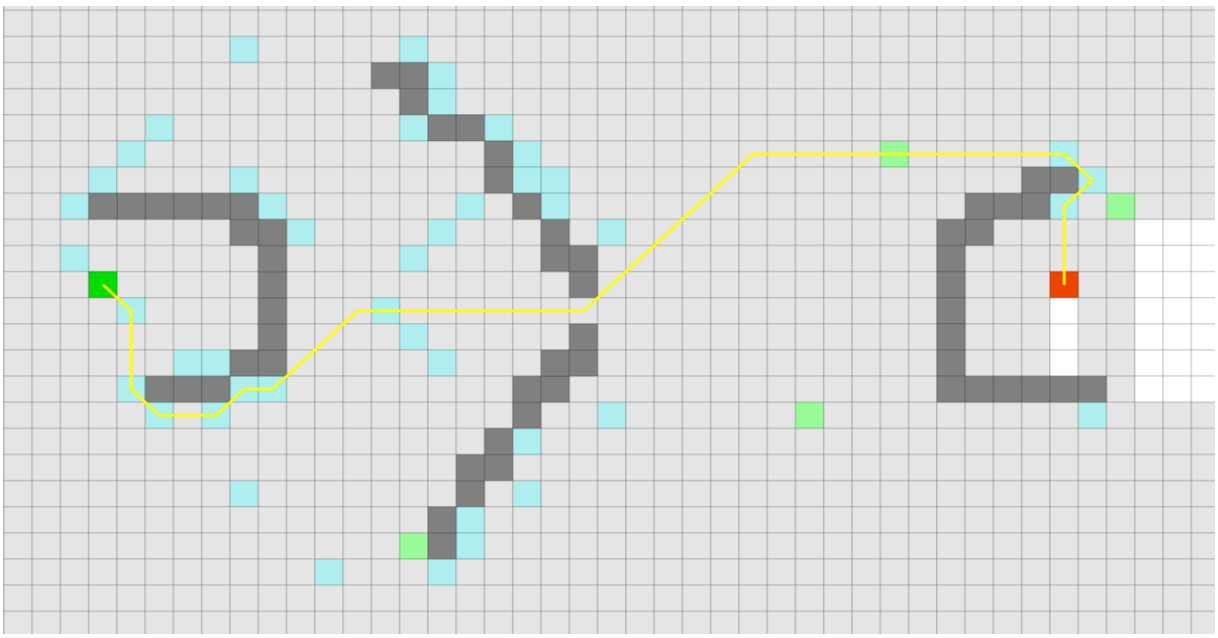
JPS es una extensión de A*. La gran diferencia radica a la hora de encontrar los nodos vecinos de un nodo.

Mientras A* considera todos los nodos vecinos y los añade a la lista de abiertos JPS se detiene en encontrar nodos relevantes. Test: http://qiao.github.io/PathFinding.js/visual/

A*



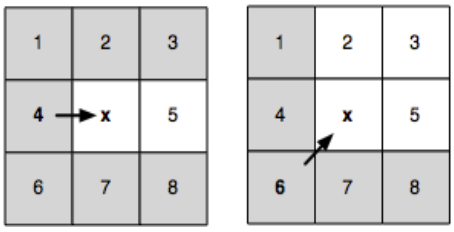
JPS



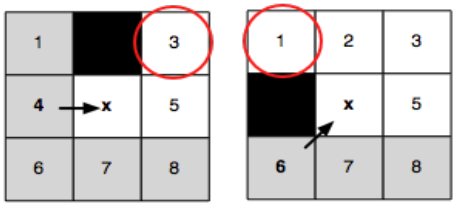
Como vemos JPS explora muchísimos menos nodos que A* y encuentra la misma ruta.

JPS se puede resumir en dos operaciones:

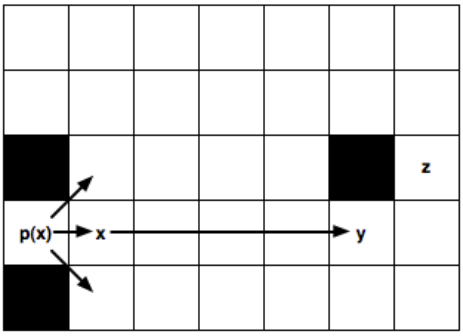
- Poda de vecinos: A la hora de expandir el nodo X, en la dirección de la flecha, podemos podar los nodos de color gris, ya que pueden ser alcanzados óptimamente desde el padre de X sin pasar por X.

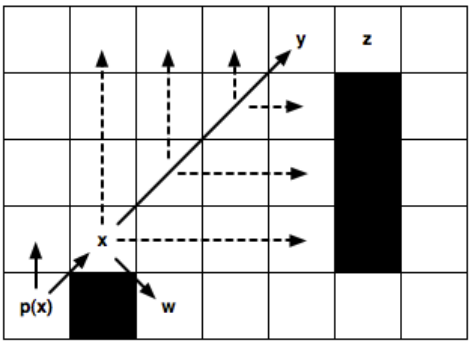


- Vecinos forzosos: Expandiendo el nodo X en la dirección de la flecha, si tenemos un obstáculo en ciertas posiciones, no podemos podar los nodos marcados, ya que el camino óptimo hasta ese nodo desde el padre de X esta bloqueado.



Estas operaciones se realizan de forma recursiva hasta encontrar lo que llamamos nodos sucesores. Los nodos sucesores son nodos que tiene vecinos que no pueden ser alcanzados por una ruta simétrica alternativa, la ruta óptima debe pasar por ellos.





Y representa nodos sucesores en las imágenes.

Al explorar los vecinos de un nodo, se busca de forma recursiva hasta encontrar estos nodos sucesores. Que serán los que se añadan a la lista de abiertos de A*.

Enlaces:

https://harablog.wordpress.com/2011/09/07/jump-point-search/

https://gamedevelopment.tutsplus.com/tutorials/how-to-speed-up-a-pathfinding-with-the-jump-point-search-algorithm--gamedev-5818

http://aigamedev.com/open/tutorial/symmetry-in-pathfinding/

http://zerowidth.com/2013/05/05/jump-point-search-explained.html

https://github.com/qiao/PathFinding.js

http://qiao.github.io/PathFinding.js/visual/

Experimentos

Para las pruebas he comparado los resultados obtenidos, con el algoritmo A* y con JPS.

He lanzado las búsquedas con el launch de 10cm, desde la posición de inicio a 4 puntos:

Punto 1

A*

```
00000]: Inserta en Plan: 47.250001, 21.650000
00000]: Sale del bucle de generaci??n del plan.
00000]: Num nodos en abiertos: 361
00000]: Num nodos en cerrados: 6013
00000]: Num nodos en PATH: 294
00000]: Distancia del PATH: 31.395418
00000]: Time: =  12.600000
00000]: Map update loop missed its desired rate of 5.0000Hz... the loop actually took 12.5000 seconds
```

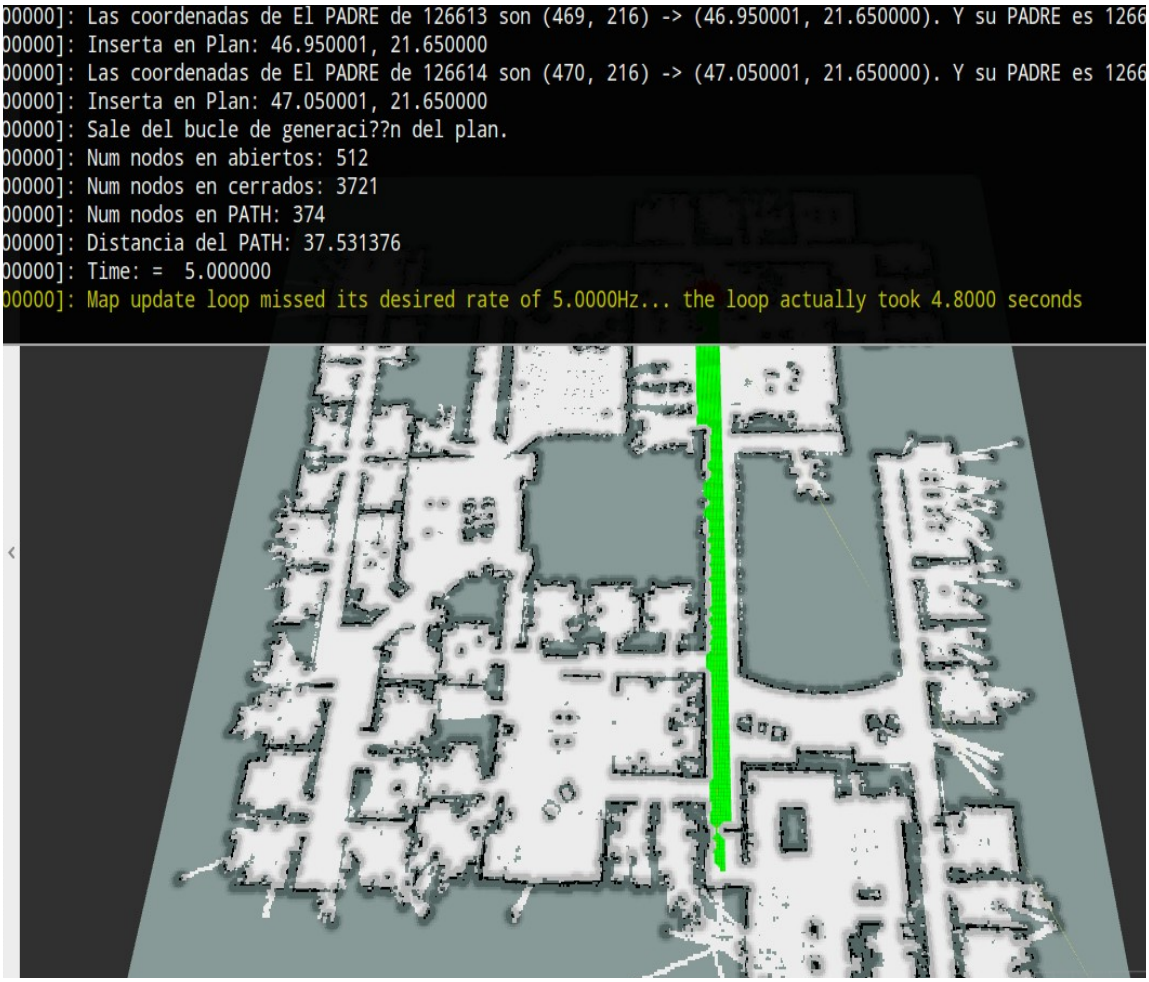
JPS

```
.261574177, 38.600000000]: Las coordenadas de El PADRE de 123718 son (494, 211) -> (49.450001, 21.150000). Y su PADRE es 123700.
.261599455, 38.600000000]: Inserta en Plan: 49.450001, 21.150000
.261633881, 38.600000000]: Las coordenadas de El PADRE de 123700 son (476, 211) -> (47.650001, 21.150000). Y su PADRE es 126615.
.261660635, 38.600000000]: Inserta en Plan: 47.650001, 21.150000
.261741701, 38.600000000]: Sale del bucle de generaci??n del plan.
.261833804, 38.600000000]: Num nodos en abiertos: 380
.261921713, 38.600000000]: Num nodos en cerrados: 2234
.261946385, 38.600000000]: Num nodos en PATH: 42
.261972859, 38.600000000]: Distancia del PATH: 52.195530
.262000615, 38.600000000]: Time: =  2.700000
.264936668, 38.600000000]: Map update loop missed its desired rate of 5.0000Hz... the loop actually took 2.6000 seconds
```

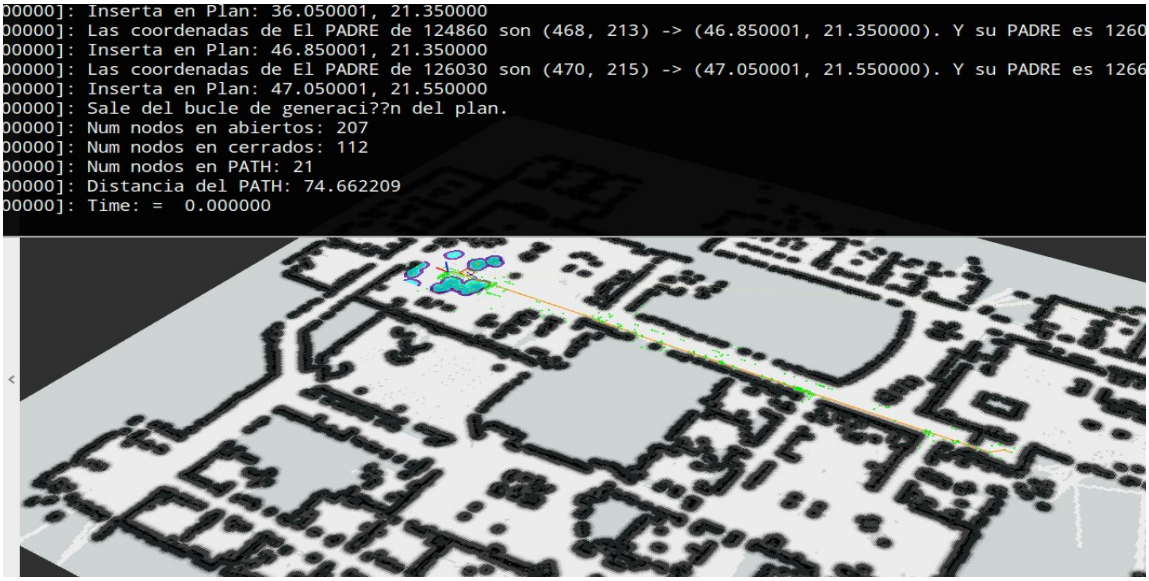
	A*	JPS	Mejora %
Abiertos	361	380	-5
Cerrados	6013	2234	169,16
PATH	294	42	600
Time	12,6	2,7	366,67

Punto 2

A*



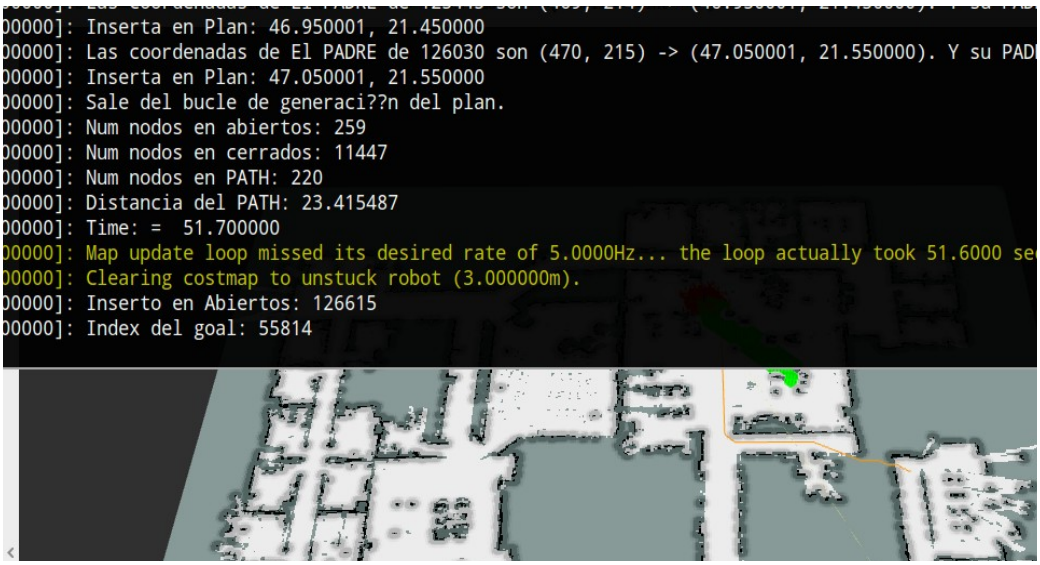
JPS



	A*	JPS	Mejora %
Abiertos	512	207	147,34
Cerrados	3721	112	3222,32
PATH	374	21	1680,95
Time	5	0,0001	4999900

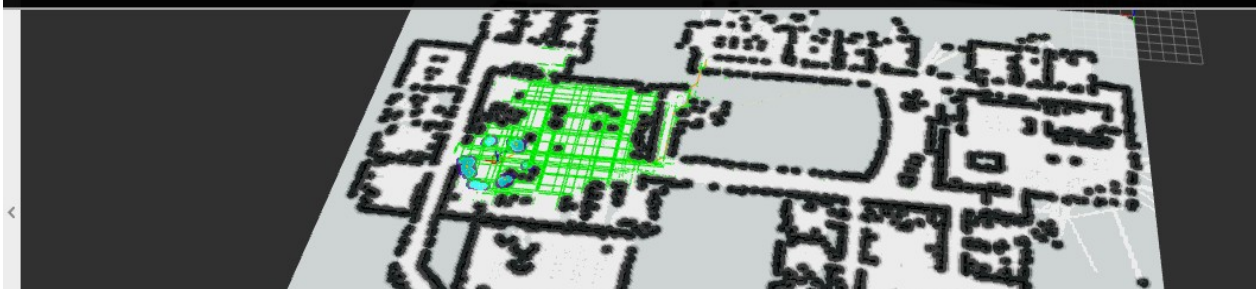
Punto 3

A*



JPS

```
00000]: Las coordenadas de El PADRE de 124860 son (468, 213) -> (46.850001, 21.350000). Y su PADRE es 1260
00000]: Inserta en Plan: 46.850001, 21.350000
00000]: Las coordenadas de El PADRE de 126030 son (470, 215) -> (47.050001, 21.550000). Y su PADRE es 1266
00000]: Inserta en Plan: 47.050001, 21.550000
00000]: Sale del bucle de generaci??n del plan.
00000]: Num nodos en abiertos: 470
00000]: Num nodos en cerrados: 4610
00000]: Num nodos en PATH: 28
00000]: Distancia del PATH: 41.421745
00000]: Time: =  9.700000
00000]: Map update loop missed its desired rate of 5.0000Hz... the loop actually took 9.5000 seconds
```

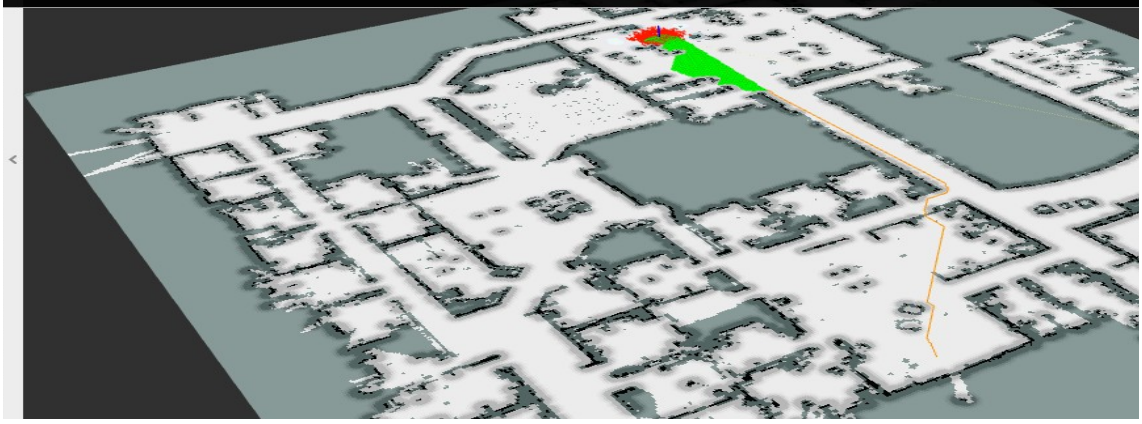


	A*	JPS	Mejora %
Abiertos	259	470	-44,89
Cerrados	11447	4610	148,31
PATH	220	28	685,71
Time	51,7	9,7	432,99

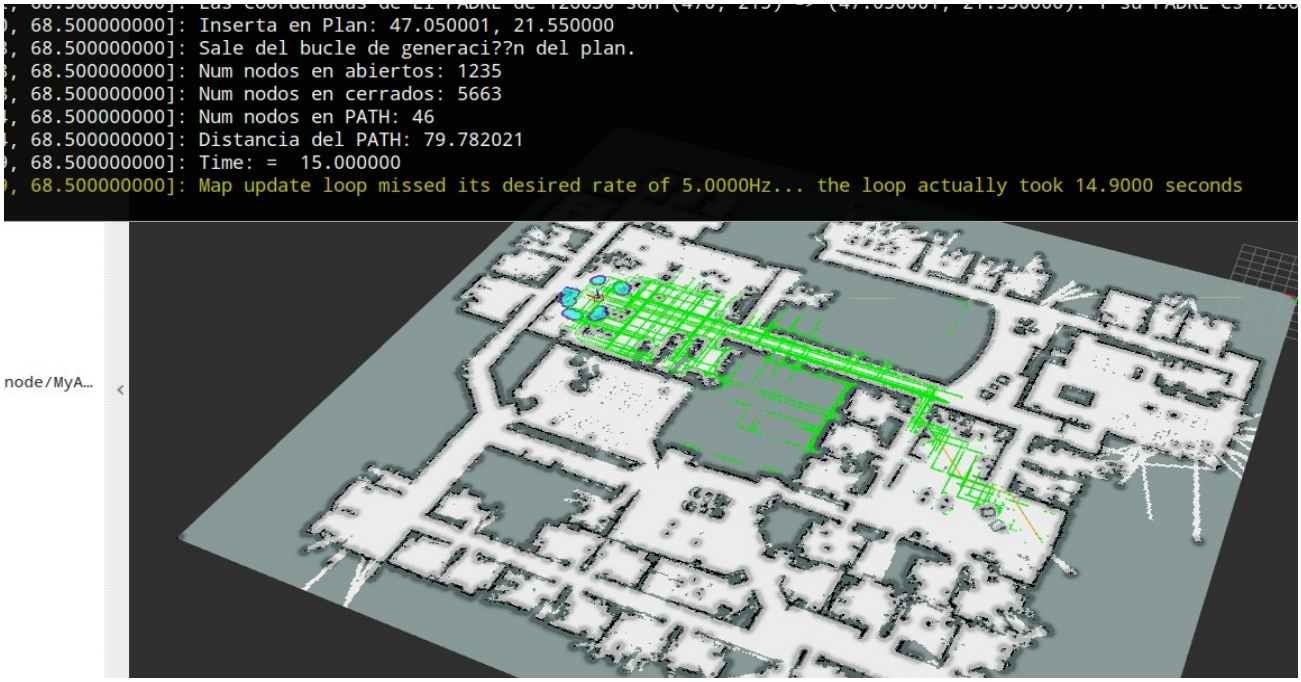
Punto 4

A*

```
000000]: Inserta en Plan: 47.050001, 21.650000
000000]: Sale del bucle de generaci??n del plan.
000000]: Num nodos en abiertos: 734
000000]: Num nodos en cerrados: 19965
000000]: Num nodos en PATH: 379
000000]: Distancia del PATH: 41.552116
000000]: Time: =  169.100000
000000]: Map update loop missed its desired rate of 5.0000Hz... the loop actually took
000000]: Clearing costmap to unstuck robot (3.000000m).
000000]: Inserto en Abiertos: 126615
000000]: Index del goal: 184064
```



JPS



	A*	JPS	Mejora %
Abiertos	734	1235	-40,57
Cerrados	19965	5663	252,55
PATH	379	46	723,91
Time	169,9	15	1032,67