Instituto Tecnológico de Estudios Superiores de Monterrey

Campus Puebla

Implementación de robótica inteligente (Gpo 501)

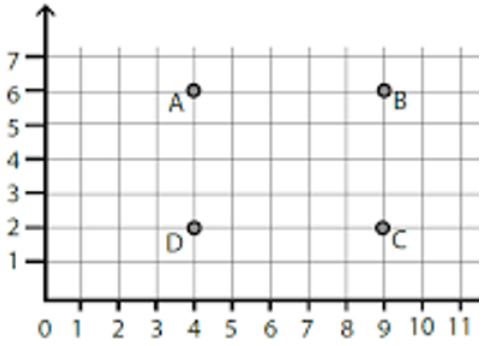
Actividad 1.10 (SLAM de Lidar)

Alumno
José Diego Tomé Guardado A01733345
Victor Manuel Vázquez Morales A01736352
Pamela Hernández Montero A01736368
Fernando Estrada Silva A01736094

Fecha de entrega
Viernes 26 de Abril de 2024

Actividad: Implementar el código requerido para generar el seguimiento de los siguientes waypoints de forma aleatoria, ajustando los parámetros: sampleTime, tVec, initPose, lidar.scanAngles, lidar.maxRange, waypoints, controller.LookaheadDistance, controller.DesiredLinearVelocity y controller.MaxAngularVelocity. Evadiendo los obstáculos del mapa de navegación "exampleMap" y "complexMap"

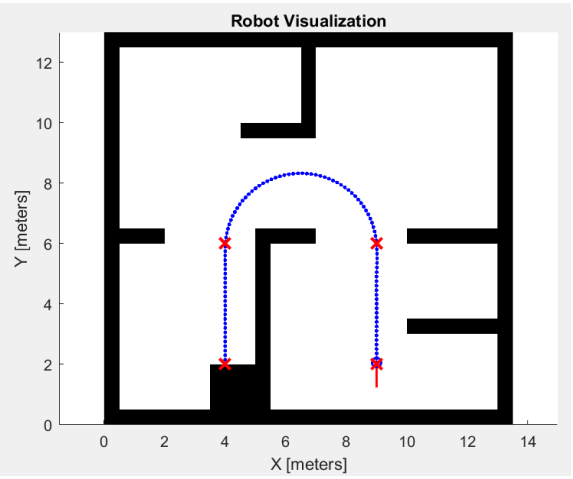
Primera trayectoria




```
% Sample time and time array
sampleTime = 0.1;                % Sample time [s]
tVec = 0:sampleTime:10.3;          % Time array
% Initial conditions
initPose = [4;2;pi/2];              % Initial pose
(x y theta)
pose = zeros(3,numel(tVec));     % Pose matrix
pose(:,1) = initPose;
```

- **exampleMap**

```
lidar.scanAngles = linspace(0,2*pi,300);%51
lidar.maxRange = 0.35;%5
```




```
% Create waypoints
waypoints = [initPose(1:2)';
             4 2;
             4 6;
             9 6;
             9 2];

% Pure Pursuit Controller
controller = controllerPurePursuit;
controller.Waypoints = waypoints;
controller.LookaheadDistance = 0.4;%0.5
controller.DesiredLinearVelocity = 1.5;%0.75
controller.MaxAngularVelocity = 0.6
OVEJA ANONIMA
```

Análisis:

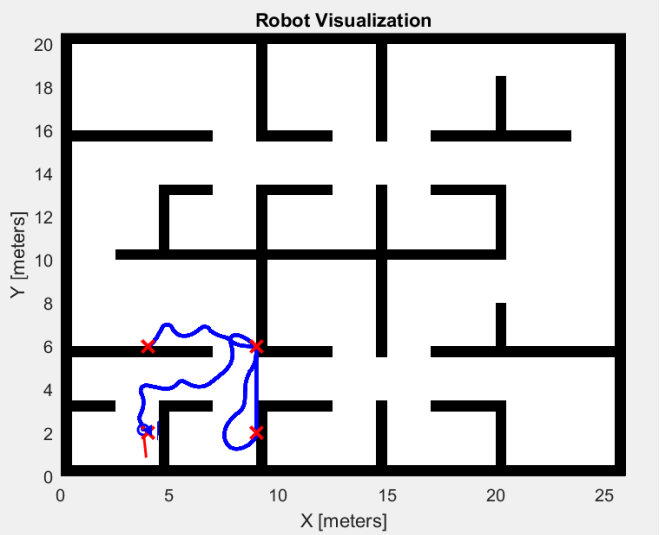
Como podemos observar tendremos una trayectoria bastante sencilla en la que solo tenemos 4 waypoints, de esta manera nosotros decidimos optar por formar una trayectoria mucho más sencilla como podemos observar en la qué comenzará y terminará simplemente dibujando una curva. Al principio al tener un LookaheadDistance más alto tuvimos algunos problemas debido a que detectaba el otro waypoint, también con la ayuda de un buen maxRange del lidar se pudo conseguir una mejor trayectoria para que no chocará con las paredes, además de que la velocidad angular la tuvimos que poner en un valor de 0.6 que es muy bajo para qué hiciera un giro más abierto y tomará enm cuenta el tercer waypoint ya que al tener un valor más alto podia sobre pasar correctamente la pared pero no llegaba correctamente al waypoint.

- **complexMap**

```
% Sample time and time array
sampleTime = 0.1;
tVec = 0:sampleTime:52;
% Initial conditions
initPose = [4;6;pi/6];
pose = zeros(3,numel(tVec));
pose(:,1) = initPose;
% Create lidar sensor
lidar = LidarSensor;
lidar.sensorOffset = [0,0];
lidar.scanAngles = linspace(0,2*pi,500);
lidar.maxRange = 0.845%5

%PRUEBA --- complexMap
waypoints = [initPose(1:2)';
             4 6;
             9 6;
             9 2;
             9 6;
             4,2];

% Pure Pursuit Controller
controller = controllerPurePursuit;
controller.Waypoints = waypoints;
controller.LookaheadDistance = 0.4;%0.5
controller.DesiredLinearVelocity = 0.5; %0.75
controller.MaxAngularVelocity = 2 %1.5
```

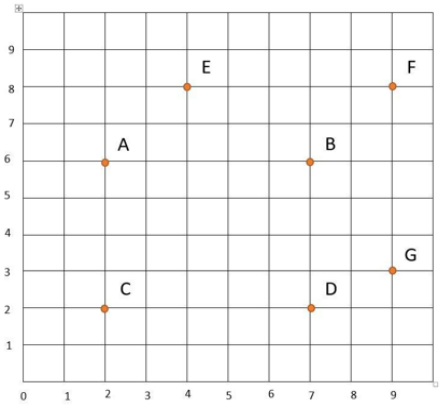


Análisis:

Continuando tendremos el mismo recorrido pero ahora para el complexMap en el que como indica su nombre es más dificil poder hacer el recorrido ya que cuenta con más paredes y un laberinto más complicado, para esta ocasión decidimos poner los waypoints de otra forma de manera a que hiciera un recorrido más optimo sin chocar con las paredes. Tuvimos que volver a pasar por un waypoint debido a que del tercer al cuarto waypoint no tenía otra forma de hacerlo más que de ciclarse o sobrepasar la pared.

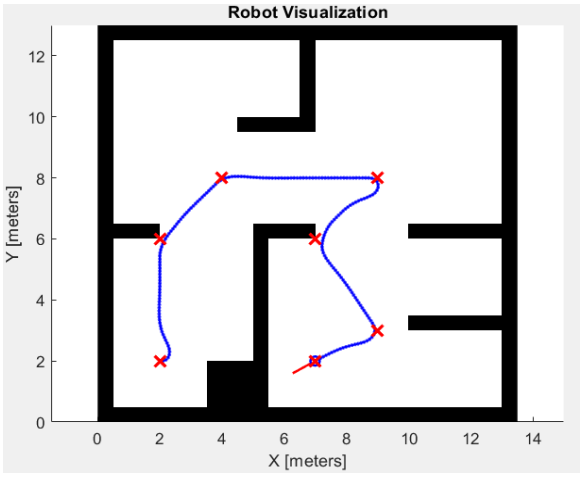
Entonces decidimos regresar al anterior waypoint y de ahi irse al último lo que generó un buen recorrido donde se formó todo de buena forma. Al tener el mismo valor de LookaheadDistance es igual pequeño solo que la velocidad lineal en este caso es menor para que pueda hacer de mejor forma los giros con la combinación de un valor más alto de la velocidad angular para hacer los giros más cerrados para no chocar con las paredes. Finalmente un valor más alto del lidar debido a que necesitamos que detecte todas las paredes y como son más partes donde la ve es por ello.

Segunda trayectoria



- **exampleMap**

```
% Waypoints
waypoints = [initPose(1:2)';
             2 6;
             4 8;
             9 8;
             7 6;
             9 3;
             7 2 ];
% Sample time and time array
sampleTime = 0.1;
tVec = 0:sampleTime:20.5;
% Initial conditions
initPose = [2;2;0];
pose = zeros(3,numel(tVec));
pose(:,1) = initPose;
% Create lidar sensor
lidar = LidarSensor;
lidar.sensorOffset = [0,0];
lidar.scanAngles = linspace(-pi,pi,200);
lidar.maxRange = 0.7;
% Pure Pursuit Controller
controller = controllerPurePursuit;
controller.Waypoints = waypoints;
controller.LookaheadDistance = 0.5;
controller.DesiredLinearVelocity = 1;
controller.MaxAngularVelocity = 100
```

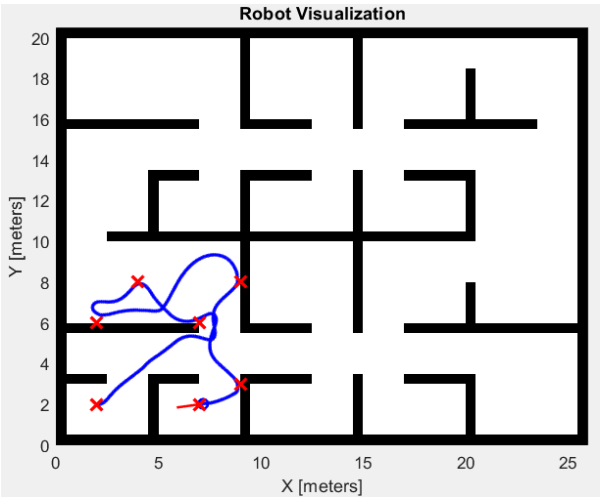


Análisis:

Para el desarrollo de la trayectoria primero determine la mejor forma de recorrer de puntos considerando los que quedaban más lejos con referencia al origen ubicado en las coordenadas del punto C y el mapa. En este caso el recorrido de los puntos fue menor debido a la facilidad de recorrer los puntos, respecto a las variaciones de de las condiciones iniciales así como las del lidar, se observó que no generaban un cambio significativo en el recorrido de la trayectoria. Tenemos un valor considerable para el LookaheadDistace para que no vea otros waypoints y considerando una velocidad angular grande para poder hacer los giros más abiertos y lleguen correctamente a cada waypoints.

- **complexMap**

```
% Create waypoints
waypoints = [initPose(1:2)';
             2 6;
             4 8;
             9 8;
             7 6;
             9 3;
             7 2];
% Sample time and time array
sampleTime = 0.1;
tVec = 0:sampleTime:33;
% Initial conditions
initPose = [2;2;0];
pose = zeros(3,numel(tVec));
pose(:,1) = initPose;
% Create lidar sensor
lidar = LidarSensor;
lidar.sensorOffset = [0,0];
lidar.scanAngles = linspace(-pi,pi,200);
lidar.maxRange = 0.5;
% Pure Pursuit Controller
controller = controllerPurePursuit;
controller.Waypoints = waypoints;
controller.LookaheadDistance = 0.5;
controller.DesiredLinearVelocity = 1;
controller.MaxAngularVelocity = 200
```

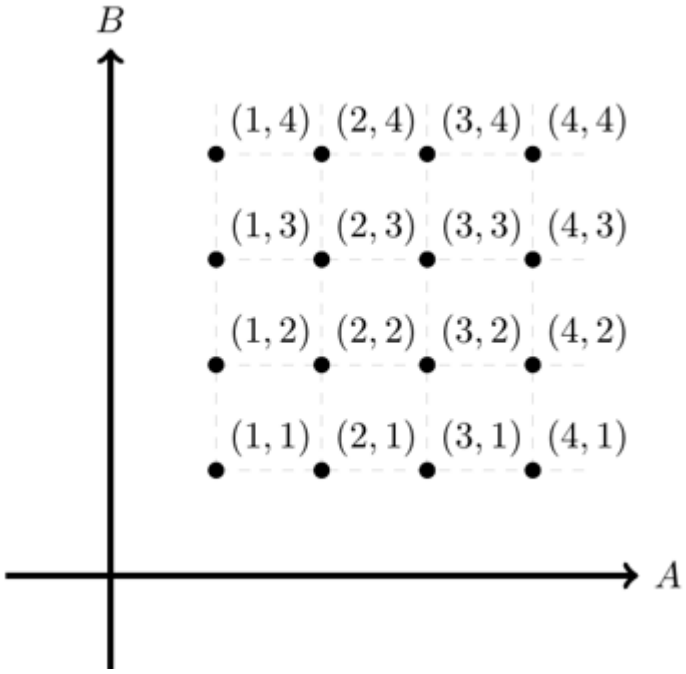


Análisis:

Con respecto al código anterior del exampleMap, en este caso el trazado de la trayectoria se consideró con respecto a la mejor forma de recorrer todos los puntos evitando de traspase la pared debido a que al ir en línea recta el lidar se confunde detectando los puntos y continúa sin detenerse, en este caso al pasar de lado el lidar funciona y evita el obstáculo.

Para este caso tendremos una trayectoria diferente ya que consideramos un mapa más complicado y tendremos que vamos a tener una trayectoria con las mismas variables de LookaheadDistance y la velocidad lineal pero en este caso aumentaremos la velocidad angular para que los giros sean más cerrados y más precisos.

Tercer trayectoria



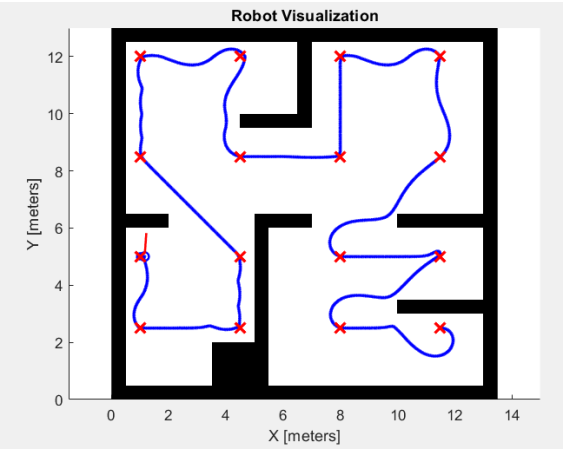
- **exampleMap**

```
% Sample time and time array
sampleTime = 0.1;
tVec = 0:sampleTime:120;

% Initial conditions
initPose = [11.5;2.5;0];
pose = zeros(3,numel(tVec));
pose(:,1) = initPose;

% Create lidar sensor
lidar = LidarSensor;
lidar.sensorOffset = [0,0];
lidar.scanAngles = linspace(0,2*pi,200);%51
lidar.maxRange = 0.55;%5

% Create waypoints
waypoints = [initPose(1:2)';
             11.5 2.5;
             8 2.5;
             11.5 5;
             8 5;
             11.5 8.5;
             11.5 12;
             8 12;
             8 8.5;
             4.5 8.5;
             4.5 12;
             1 12;
             1 8.5;
```



```
                4.5 5;
                4.5 2.5;
                1 2.5;
                1 5;
                ];

        % Pure Pursuit Controller
        controller = controllerPurePursuit;
        controller.Waypoints = waypoints;
        controller.LookaheadDistance = 0.3;%0.5
        controller.DesiredLinearVelocity = 0.5; %0.75
        controller.MaxAngularVelocity = 10;
```

Análisis:

Para este caso, se optó por multiplicar los puntos originales marcados por la figura de tal forma que los landmarks abarcaran la mayor parte del mapa. De igual forma, el seguimiento de los landmarks se siguió de tal manera que no se siguieran de línea recta cuando existiera una pared entre ellos, ya que como se ha mencionado en otros casos, el sensor Lidar suele confundirse y no corrige la trayectoria de manera apropiada.

Respecto a las velocidades, podemos observar que la velocidad angular máxima es mucho mayor que la velocidad lineal, lo que permite al robot priorizar la corrección del ángulo. Por último, se redujo considerablemente la distancia máxima del sensor Lidar, de tal manera que pudiera llegar a ciertos puntos que se encuentran muy cerca de las paredes.

- **complexMap**

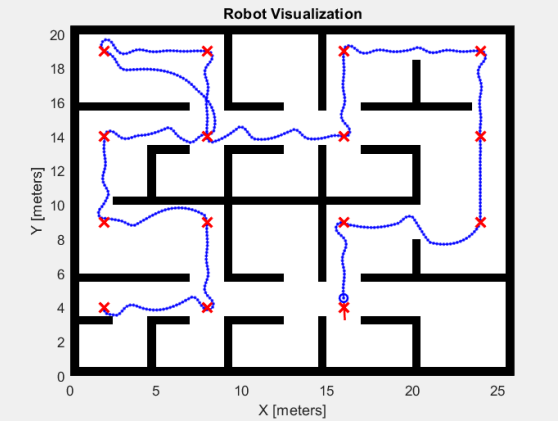
```
% Sample time and time array
sampleTime = 0.1;               % Sample time [s]
tVec = 0:sampleTime:55.5;         % Time array
% Initial conditions
initPose = [2;4;-pi/2];             % Initial pose (x y theta)
pose = zeros(3,numel(tVec));    % Pose matrix
pose(:,1) = initPose;

% Create lidar sensor
lidar = LidarSensor;
lidar.sensorOffset = [0,0];
lidar.scanAngles = linspace(0,2*pi,200);%51
lidar.maxRange = 0.92;%5

% Create waypoints
waypoints = [initPose(1:2)';
             8 4;
             8 9;
             2 9;
             2 14;
             8 14;
```

```
2 19;

8 19;
8 14;
16 14;
16 19;
24 19;
24 14;
24 9;
16 9;
16 4;
24 4;
];
```




```
% Pure Pursuit Controller
controller = controllerPurePursuit;
controller.Waypoints = waypoints;
controller.LookaheadDistance = 0.3;%0.5
controller.DesiredLinearVelocity = 2; %0.75
controller.MaxAngularVelocity = 30;
```

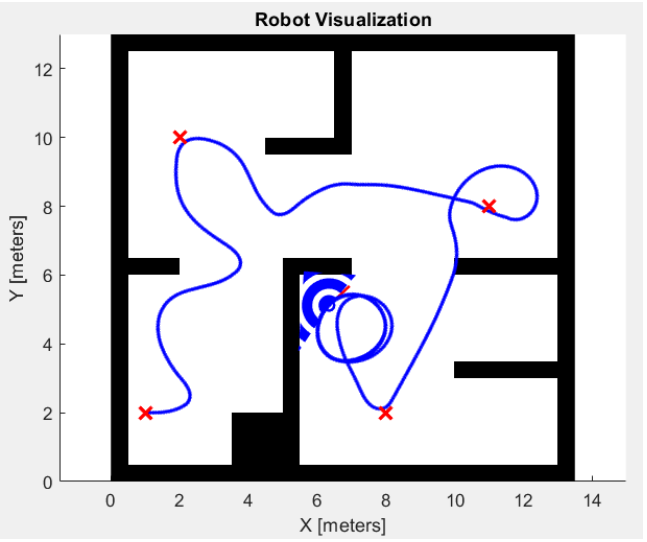
Análisis:

Al igual que en el caso anterior, los landmarks se modificaron de tal forma que abarcaran la mayor parte del mapa, pero manteniendo la figura característica. Ahora bien, al igual que en el caso anterior, notemos que la velocidad lineal sigue siendo mucho menor que la velocidad angular máxima, manteniendo el comportamiento de priorizar la corrección del ángulo. Para este caso, se aumentó considerablemente el rango de alcance del sensor Lidar para lograr que la trayectoria del robot no chocará ni atraviesa paredes.

Ahora bien, lo más destacable de este segundo escenario o mapa es que se forman secciones considerablemente más pequeñas o cerradas, lugares en los que a nuestro robot le cuesta acceder con mayor dificultad. Para dar solución a este problema en específico se optó por repetir algunos landmarks para que el robot pudiera generar una mejor trayectoria.

Actividad 2: Implementar el código requerido para generar el seguimiento de los siguientes waypoints de forma secuencial: (1, 2), (2, 10), (11, 8), (8, 2), y (1, 2) ajustando los parámetros: sampleTime, tVec, initPose, lidar.scanAngles, lidar.maxRange, waypoints,controller.LookaheadDistance, controller.DesiredLinearVelocity y controller.MaxAngularVelocity. Evadiendo los obstáculos del mapa de navegación "exampleMap" y "complexMap"

```
% Create waypoints
waypoints = [initPose(1:2)';
             1 2;
             2 10;
             11 8;
             8 2;
             2  10;
             1  2];
% Sample time and time array
sampleTime = 0.1;
tVec = 0:sampleTime:80;
initPose = [1;2;0];
pose = zeros(3,numel(tVec));
pose(:,1) = initPose;
% Create lidar sensor
lidar = LidarSensor;
lidar.sensorOffset = [0,0];
lidar.scanAngles =
linspace(0,2*pi,500);
lidar.maxRange = 1.6;%5
% Pure Pursuit Controller
controller = controllerPurePursuit;
controller.Waypoints = waypoints;
controller.LookaheadDistance = 0.8;
controller.DesiredLinearVelocity = 0.77;
controller.MaxAngularVelocity = pi;
```



Análisis:

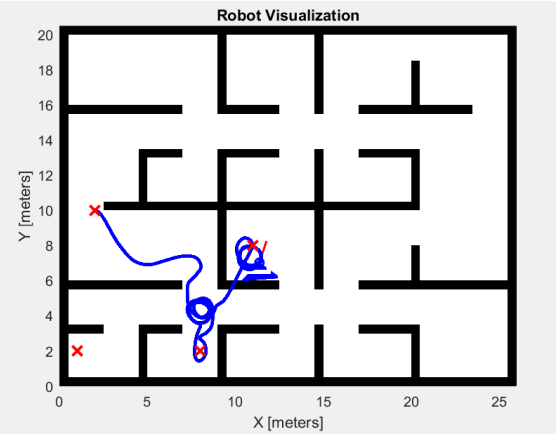
La simulación se realiza de forma exitosa en el trayecto de ida al punto final; sin embargo, al momento de regresar, el robot presenta un "bug 0". Este término hace referencia al momento en que un robot, al encontrarse con un obstáculo en su camino de regreso al punto de inicio, se ve atrapado en un bucle de comportamiento repetitivo, es por ello que al final recorre círculos.

En este caso, el Bug que resolvería el problema es el Bug 1. Esto se debe a que con el Bug 1, el robot rodeará la pared del obstáculo hasta encontrar una referencia directa hacia el objetivo. Este enfoque permite al robot rodear el obstáculo de manera eficiente sin necesidad de recorrer toda la longitud de la pared. Por otro lado, si se utilizara el Bug 2 para resolver el problema, el robot seguiría la pared del obstáculo hasta encontrar el punto más corto para continuar hacia el objetivo. Sin embargo, dado que la pared es muy larga, recorrerla completamente consumiría mucho tiempo y recursos. Por lo tanto, el Bug 1 sería la opción

más adecuada en este escenario, ya que proporciona una solución más directa y eficiente para evitar el obstáculo y llegar al objetivo.

- **complexMap**

```
% Sample time and time array
sampleTime = 0.1;
tVec = 0:sampleTime:100;
% Initial conditions
initPose = [2;10;2*pi];
% Initial pose (x y theta)
pose = zeros(3,numel(tVec));
pose(:,1) = initPose;
```




```
lidar.scanAngles =
linspace(0,2*pi,500);%51
lidar.maxRange = 1.5;%5
```

```
% Create waypoints
waypoints = [initPose(1:2)';
             2 10;
             8 2;
             11 8;
             8 2;
             1  2];
```

```
% Pure Pursuit Controller
controller = controllerPurePursuit;
controller.Waypoints = waypoints;
controller.LookaheadDistance = 0.4;%0.5
controller.DesiredLinearVelocity = 0.5; %0.75
controller.MaxAngularVelocity = 5;
```

Análisis:

En este caso tendremos ahora el complexMap podemos observar qué los waypoints se colocaron en posiciones muy complicadas de acceder sobre todo considerando que en este caso es el bug 0 por lo que al encontrarse con un obstáculo se ve atrapado en un bucle, además de que hicimos una configuración lo más adecuada posible debido a qué es cierto decir que es complicado que el robot pueda tener un gran desempeño con tantas paredes en los waypoints.

Podríamos proponer resolver el problema con posible Bug 1 donde se pueda rodear la pared al obstáculo hasta encontrar una referencia directa y en el Bug 2 encontraría el punto más corto qué sería el wavepoint 2 y posiblemente seguiría teniendo problemas con atravesar las paredes, entonces consideramos la implementaciòn del Bug 1 para poder proporcionar una solución más ajustada.