



Sistemas de Control para Robots

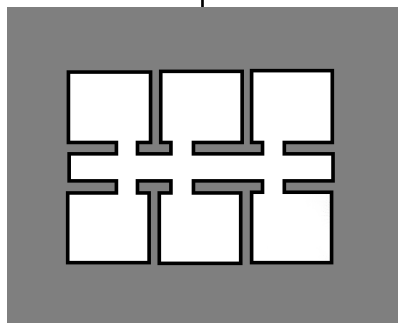
P2: Navegación local y global

Rafael Barea/Elena López

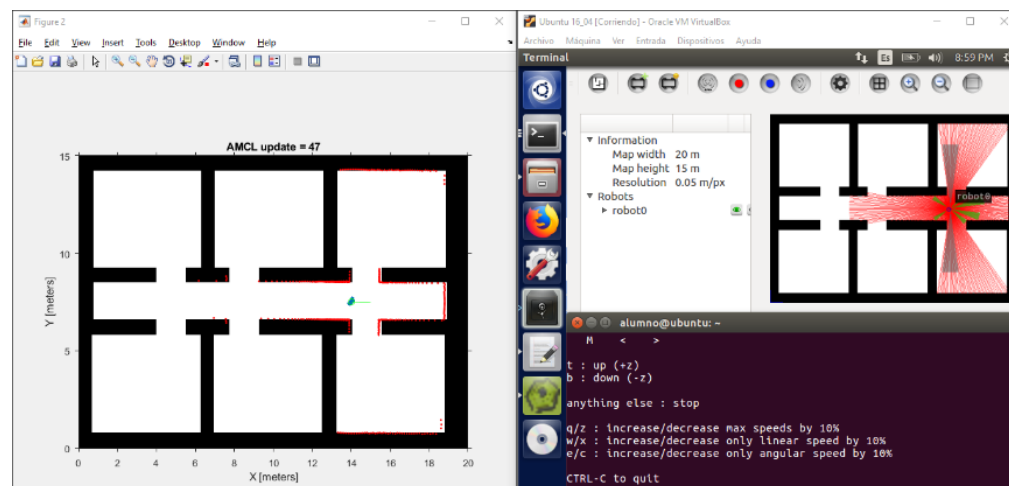
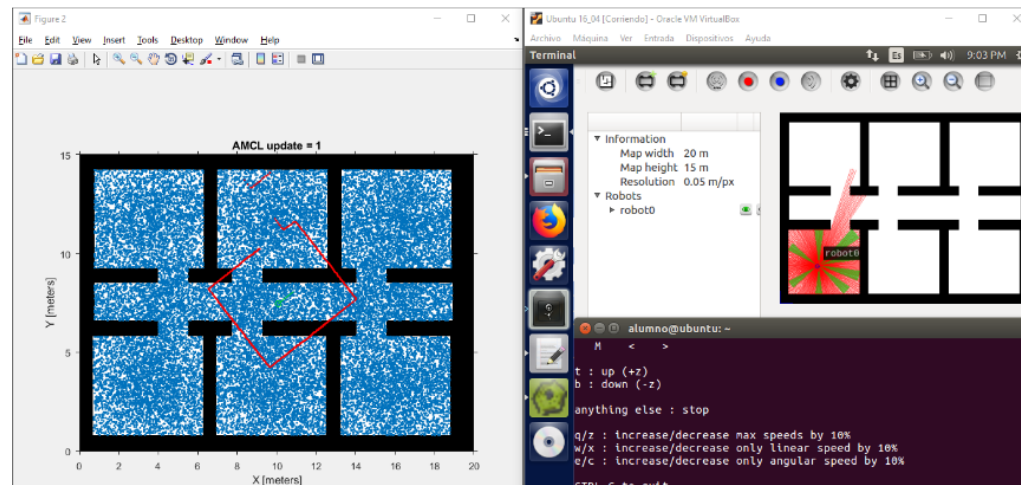
1. Introducción

PRÁCTICA 1

Generación Mapa
(SLAM)



Localización
Localización (AMCL)
Control (teleoperado)



PRÁCTICA 2 → Navegación local y global

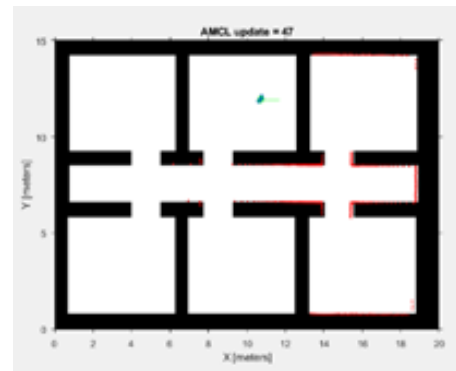
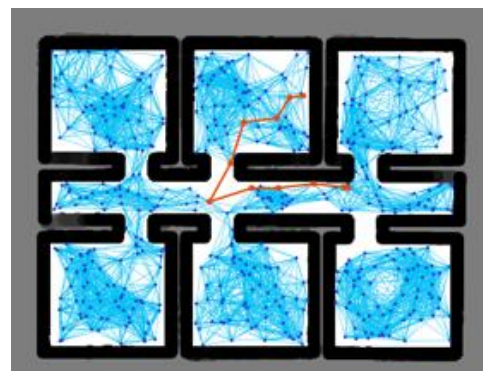
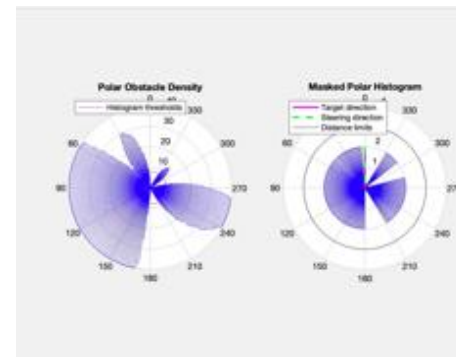
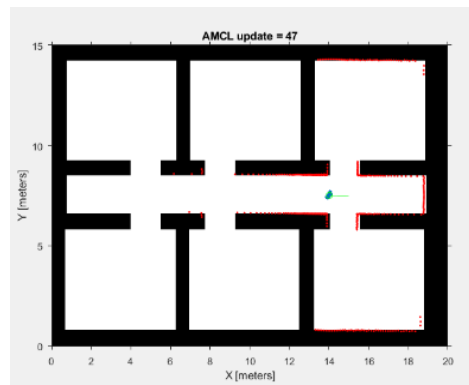
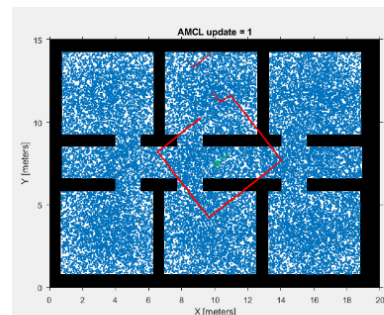
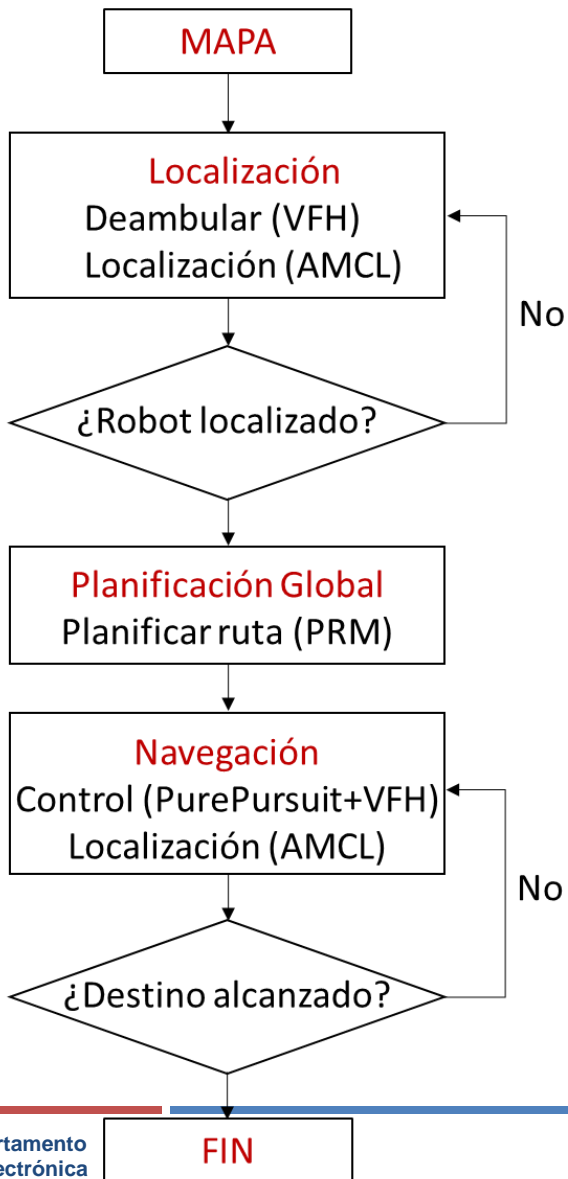
Planificador local

- Evitador de obstáculos
- VFH: Vector Field Histogram

Planificador Global

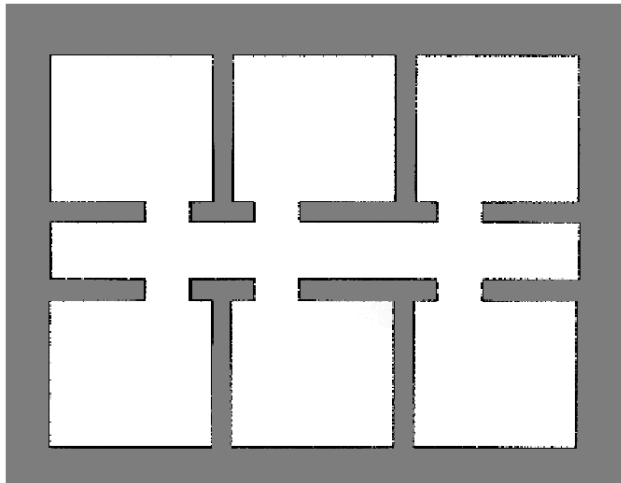
- Navegar desde un origen a un destino
- Generación de trayectorias
- Complementado con un planificador local para evitar obstáculos
- PRM: Probabilistic Road Map

1. Introducción

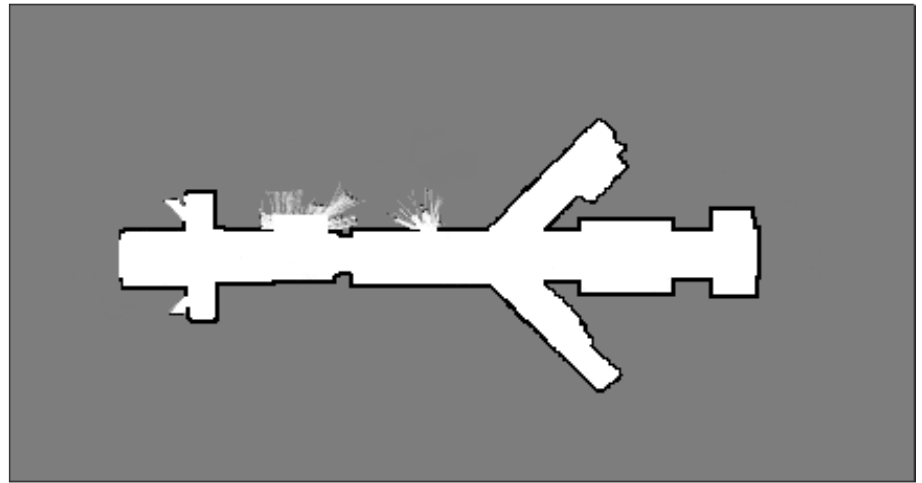


Objetivo Práctica 2

- Navegar de forma segura desde un punto origen a un punto destino de forma segura y evitando obstáculos
- Pruebas en simulación y en real



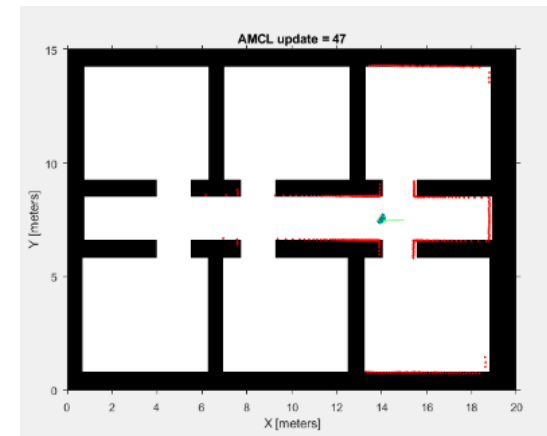
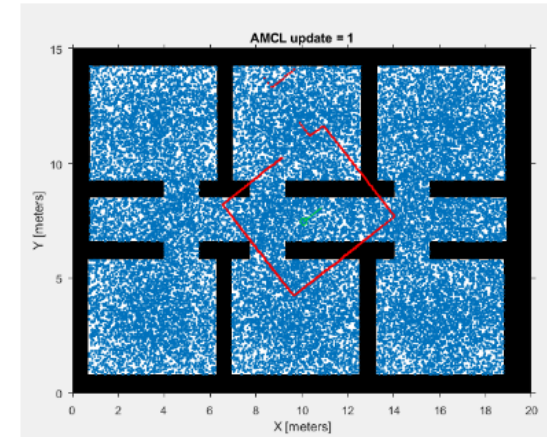
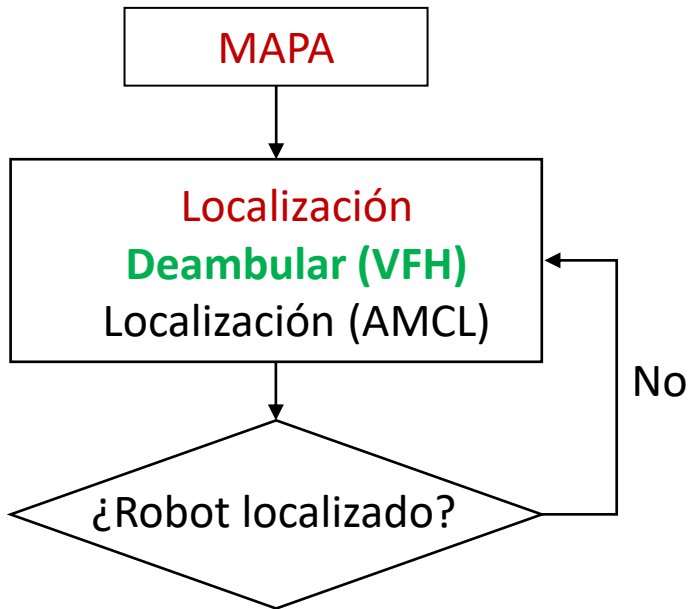
Mapa “simple_rooms”



Mapa “pasillo”

2. Planificación LOCAL

Desarrollo práctica



Simulación

- Comportamiento **wander_vfh**
 - Avance en línea recta evitando obstáculos
- Comportamiento **wander_localiza**
 - Comportamiento wander incorporando localización AMCL

Robot Real

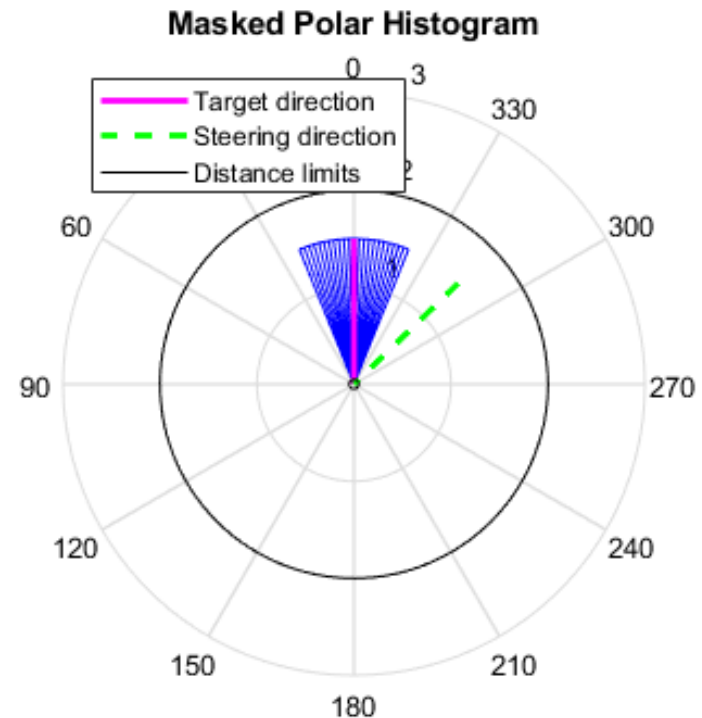
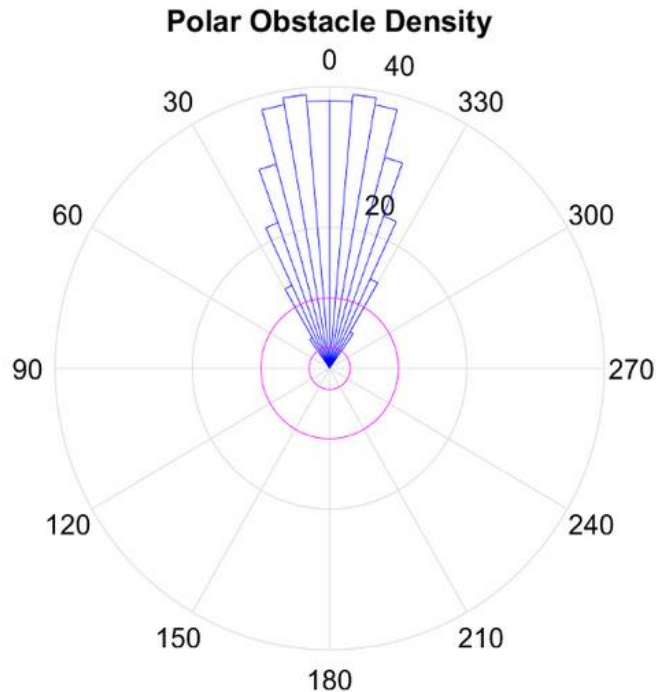
- Comprobar comportamientos anteriores con el robot real.

2. Planificación LOCAL (VFH)

Estudio Algoritmo VFH

Algoritmo VFH (Vector Field Histogram)

- Las casillas de ocupación bidimensionales se convierten en un histograma polar de 1D, que hace más fácil seleccionar la dirección óptima a seguir ante la detección de un obstáculo que no estaba contemplado en el mapa.



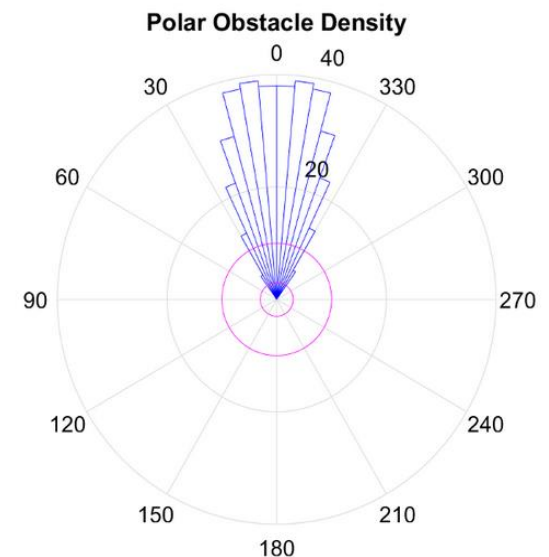
2. Planificación LOCAL (VFH)

Estudio Algoritmo VFH

- Describir el funcionamiento del algoritmo VFH disponible en la robotics toolbox de Matlab.
 - Clase "*robotics.VectorFieldHistogram*" (robotic system toolbox)
 - A partir de 2019b → **clase controllerVFH** (navigation toolbox)

Objeto **robotics.VectorFieldHistogram**

- Syntax **VFH** = **robotics.VectorFieldHistogram**
- Syntax **VFH** = **controllerVFH** (>2019b)
- Enables your robot to avoid obstacles based on range sensor data using vector field histograms (VFH).
- Given laser scan readings and a target direction to drive toward, the object computes an obstacle-free steering direction.



Objeto robotics.VectorFieldHistogram - Propiedades

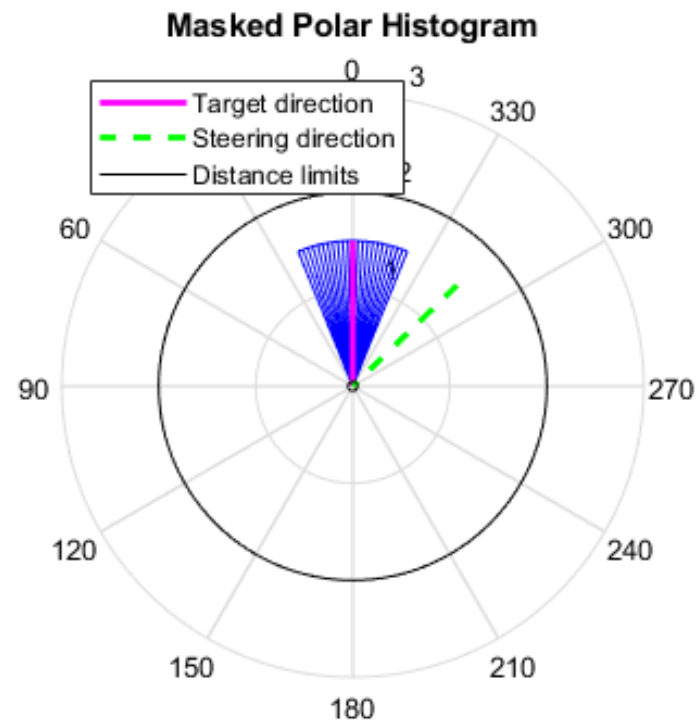
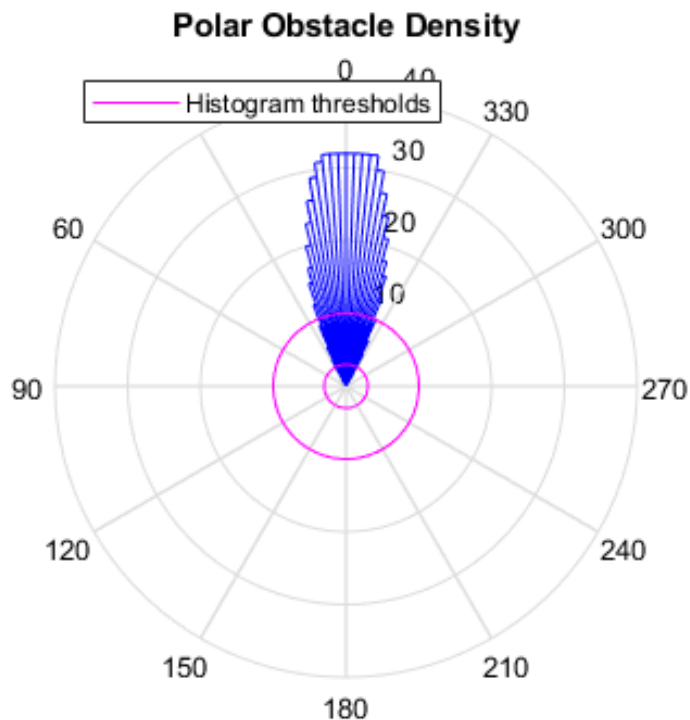
- VFH.NumAngularSectors=; Number of angular sectors in histogram 180 (default)
- VFH.DistanceLimits=; Limits for range readings [0.05 2] (default)
- VFH.RobotRadius=; Radius of robot 0.1 (default)
- VFH.SafetyDistance=; Safety distance around robot 0.1 (default)
- VFH.MinTurningRadius=; Minimum turning radius at current speed 0.1 (default)
- VFH.TargetDirectionWeight=; Cost function weight for target direction 5 (default)
- VFH.CurrentDirectionWeight=; Cost function weight for current direction 2 (default)
- VFH.PreviousDirectionWeight=; Cost function weight for previous direction 2 (default)
- VFH.HistogramThresholds=; Thresholds for binary histogram computation [3 10] (default)
- VFH.UseLidarScan=true; % Use lidarScan object as scan input

2. Planificación LOCAL (VFH)

Estudio Algoritmo VFH

Objeto robotics.VectorFieldHistogram - Usage

- `steeringDir = VFH(scan,targetDir)`
- Finds an obstacle-free steering direction using the VFH+ algorithm for the input lidarScan object, scan. A target direction is given based on the target location.



2. Planificación LOCAL (VFH)

Script wander_vfh.m

Objetivo: Implementar un comportamiento tipo wander (deambulación) detectando obstáculo mediante el algoritmo VFH

Organigrama

- Crear el objeto VFH
- Ajustar propiedades objeto VFH
- Comportamiento wander (avanzar)
 - Avanzar en línea recta mientras sea posible, y en caso de detectar algún obstáculo en dicha dirección, girar hacia una zona libre.
 - Controlador VFH con dirección de destino *targetDir=0*
 - **$\text{steeringDir} = \text{VFH}(\text{scan}, \text{targetDir})$**
 - Control de la velocidad angular proporcional a la dirección seleccionada por el algoritmo.
 - **$V_{\text{ang}} = K * \text{steeringDir}$**

DECLARACIÓN DE PUBLISHER Velocidad (Modo simulación)

- `pub_vel=rospublisher('/robot0/cmd_vel','geometry_msgs/Twist');`
- `pub_vel=rospublisher('/cmd_vel', 'geometry_msgs/Twist');` Robot Real

GENERACION DE MENSAJES

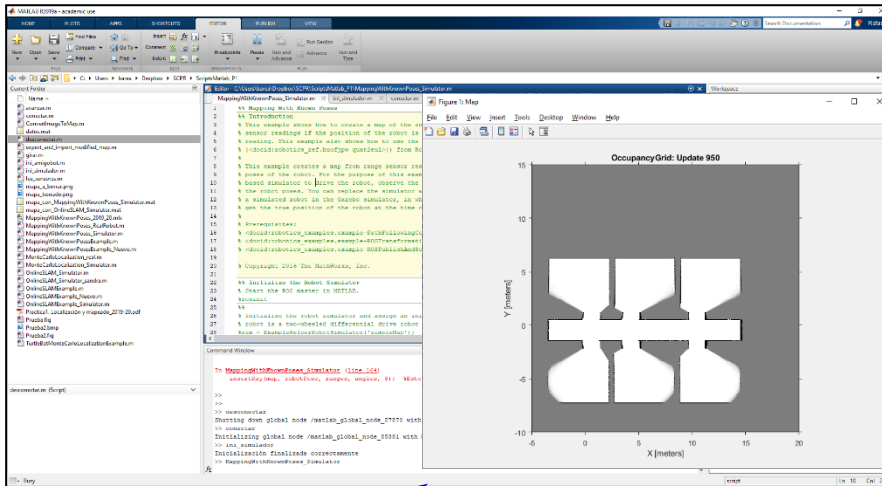
- `msg_vel=rosmessage(pub_vel);`

CAMPOS DEL MENSAJE

- `msg_vel.Linear.X=XXX;`
- `msg_vel.Linear.Y=0;`
- `msg_vel.Linear.Z=0;`
- `msg_vel.Angular.X=0;`
- `msg_vel.Angular.Y=0;`
- `msg_vel.Angular.Z=XXX;`
- `send(pub_vel,msg_vel);`

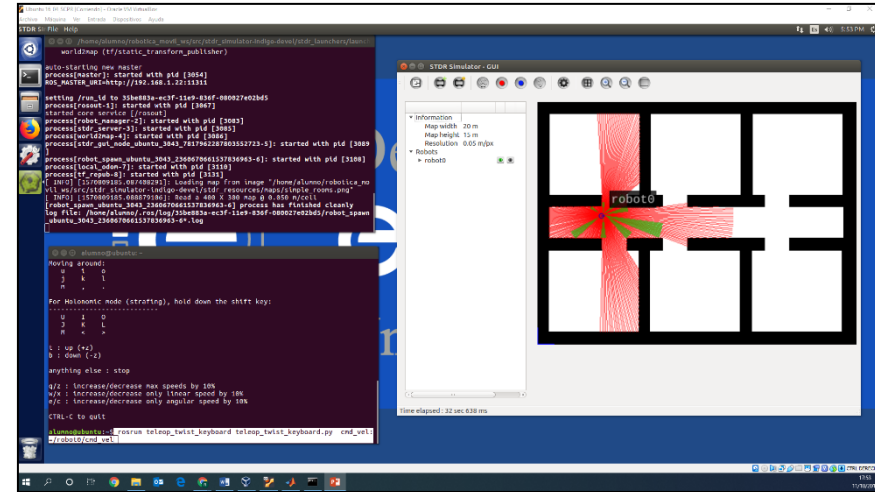
2.2.1 Planificación LOCAL (VFH) wander_vfh

PC1 Scripts matlab



1. Conectar.m
2. Ini_simulador.m
3. wander_vfh.m

PC2 Máquina virtual (MASTER)



T1: roslaunch stdr_launchers amigobot.launch

2. Planificación LOCAL (VFH)

Script wander_localiza.m

Objetivo: Incorporar el localizador AMCL al bucle de control del wander_vfh.m, de manera que el robot pueda irse localizando a medida que deambula por el entorno.

Organigrama

- Crear el objeto VFH y ajustar propiedades
- Inicializar el localizador AMCL
- Bucle (mientras robot no localizado)
 - Leer sensores
 - Ejecutar AMCL para obtener la posición estimada
 - `[isUpdated, estimatedPose, estimatedCovariance] = amcl(pose, scans);`
 - Si robot localizado → FIN (**localizado si covarianza < umbral**)
 - Controlador VFH. Evitación de obstáculos
 - Comportamiento wander. Publicar velocidad

2. Planificación LOCAL (VFH)

Cómo saber si el robot está bien localizado

AMCL → is the variant of MCL implemented in monteCarloLocalization. AMCL dynamically adjusts the number of particles based on KL-distance to ensure that the particle distribution converge to the true distribution of robot state

`[isUpdated, estimatedPose, estimatedCovariance] = amcl(pose, scans);`

- **estimatedPose**: Current pose estimate, returned as a three-element vector, [x y theta]. Computed as the mean of the highest-weighted cluster of particles.
- **estimatedCovariance**: Covariance estimate for current pose, returned as a matrix. This matrix gives an estimate of the uncertainty of the current pose. The covariance is computed as the covariance of the highest-weighted cluster of particles.

➤ Ejemplo

estimatedPose = 1×3

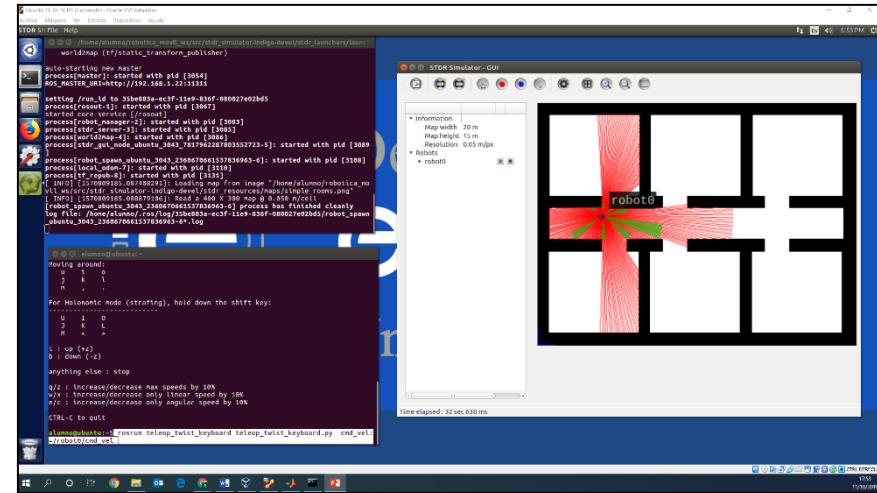
0.0350	-0.0126	0.0280
x	y	theta

covariance = 3×3

x	0.9946	-0.0012	0
	-0.0012	0.9677	y
	0	0	0.9548
			theta

- Robot localizado si `cov.x < UmbralX && cov.y < UmbralY && cov.tetha < UmbralT`

PC2 Máquina virtual (MASTER)

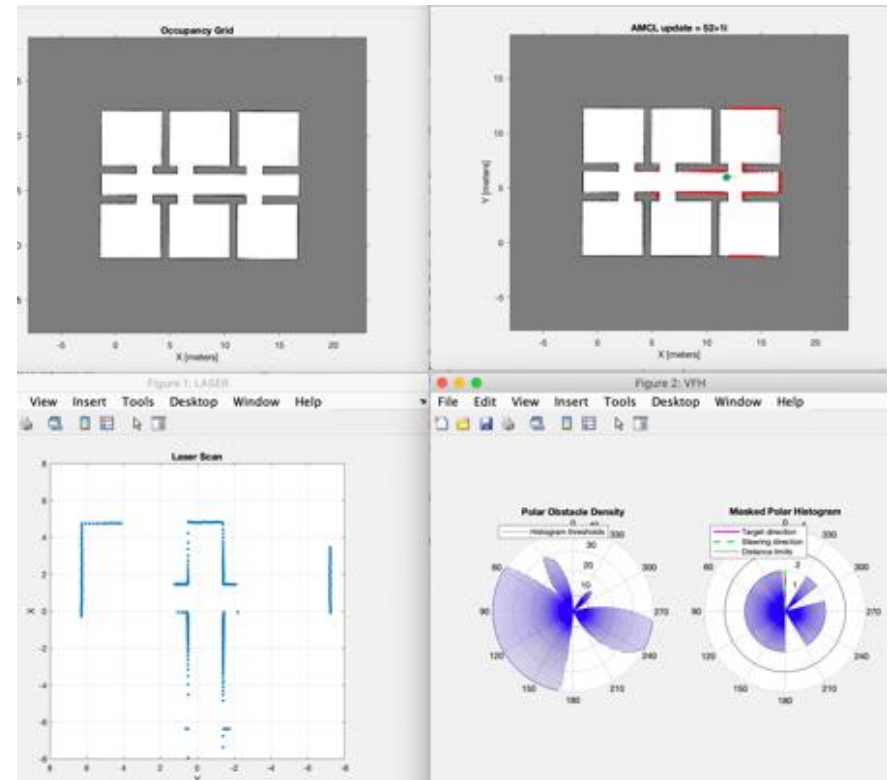
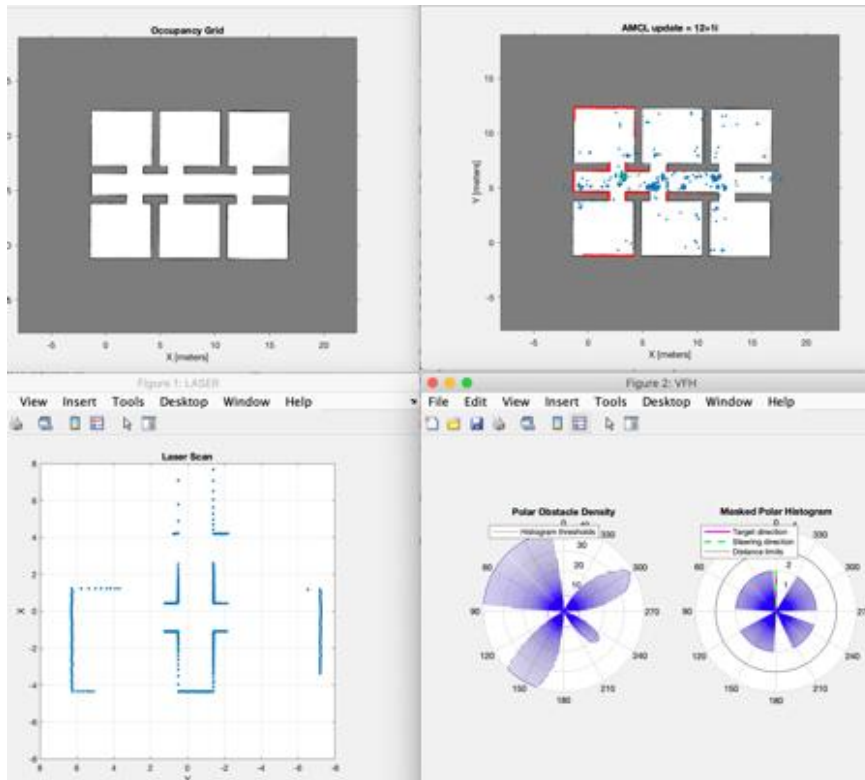


1. Conectar.m
2. Ini_simulador.m
3. wander_localiza.m

T1: `roslaunch stdr_launchers amigobot.launch`

2.2.1 Planificación LOCAL (VFH) wander_localiza

Simple_rooms (Pasillo)

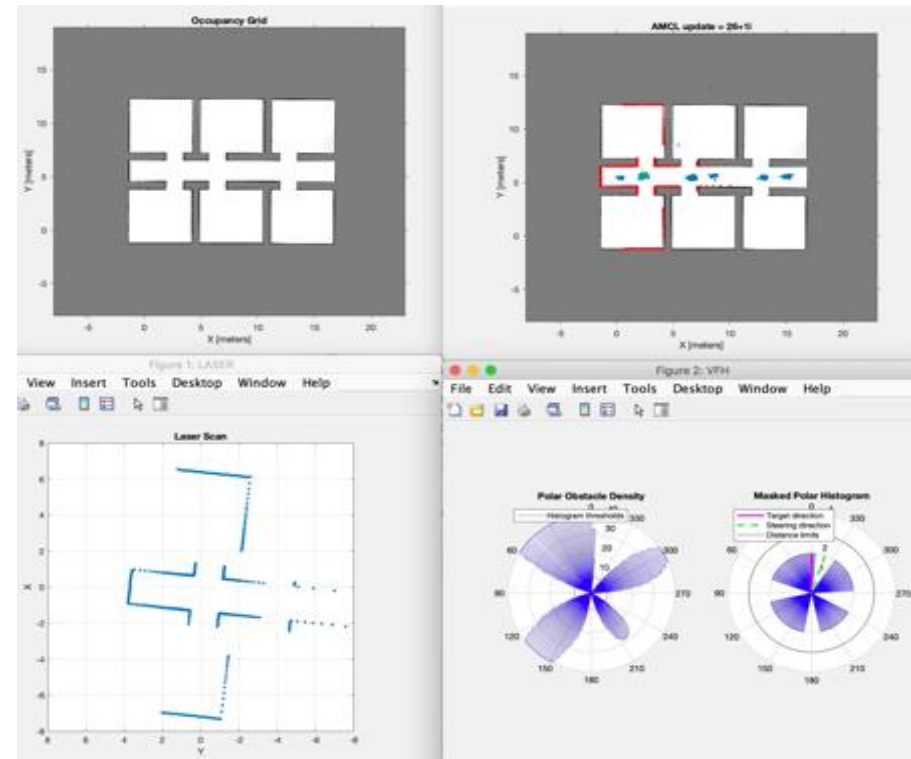
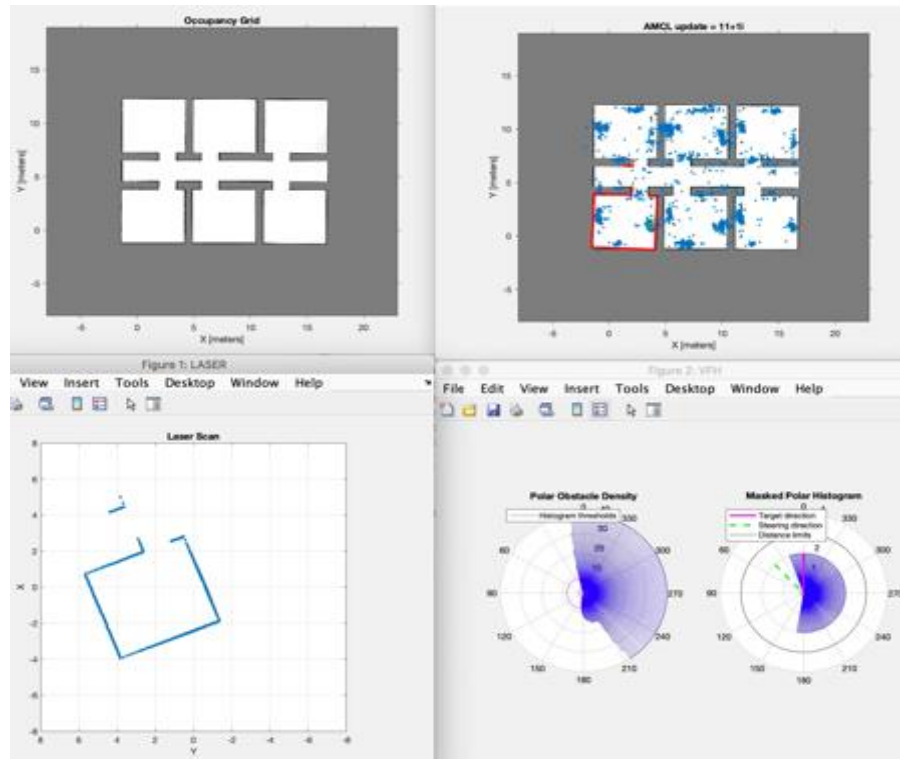


```
estimatedCovariance =
    0.0014    0.0005         0
    0.0005    0.0012         0
         0         0    1.0877

ROBOT LOCALIZADO GLOBALMENTE
fx >>
```

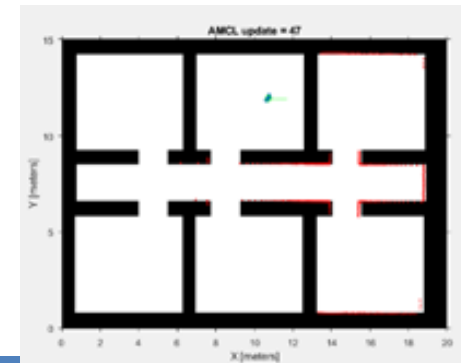
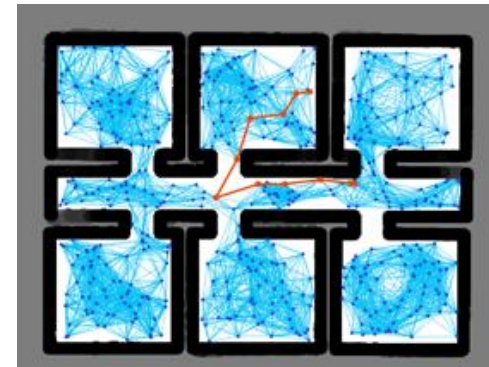
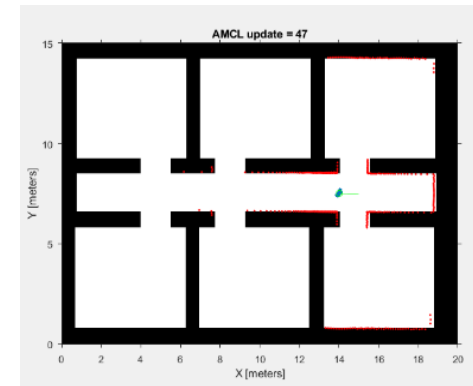
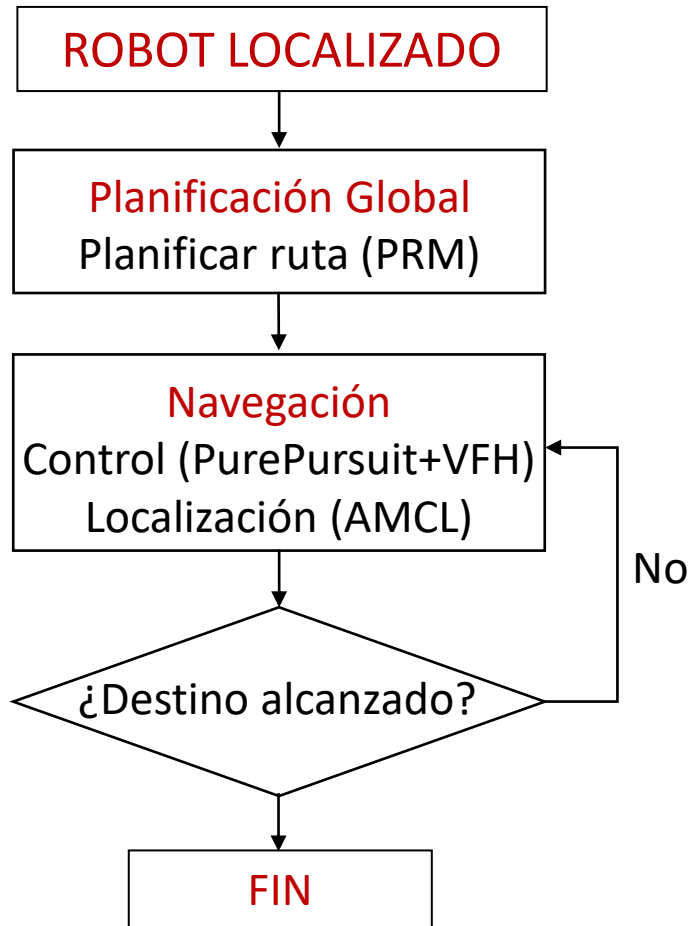
2.2.1 Planificación LOCAL (VFH) wander_localiza

Simple_rooms (Habitación)



3. Planificación GLOBAL

Desarrollo práctica



Simulación

- Comportamiento **wander_localiza_planifica**
 - Una vez el robot se haya localizado (**wander_localiza**), obtener la ruta para llegar al destino (endLocation) utilizando un planificador PRM
- Comportamiento **wander_localiza_planifica_controla**
 - Una vez planificada la ruta, añadir un controlador **PurePursuit**
- Comportamiento **navegacion_total**
 - Añadir el evitador de obstáculos VFH en el bucle de control del **PurePursuit**

Robot Real

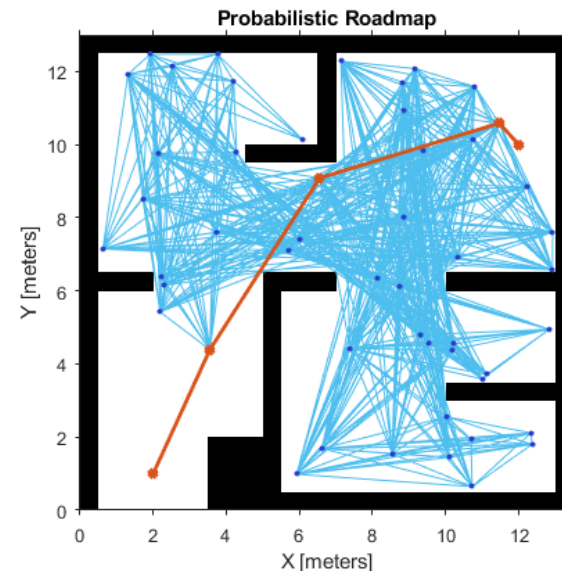
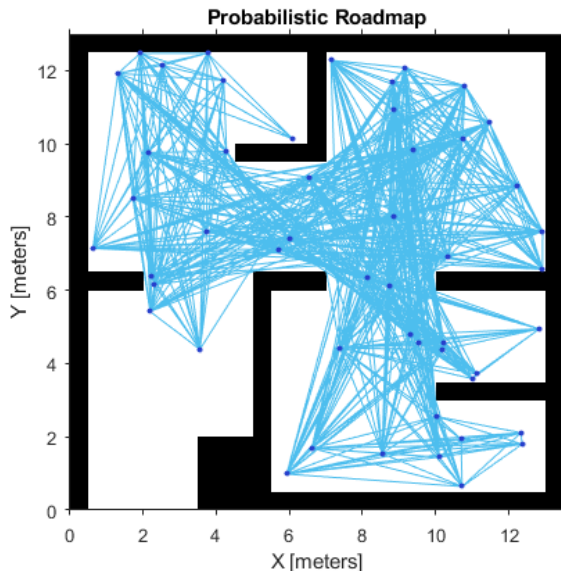
- Comprobar el comportamiento **navegacion_total** con el robot real

3. Planificación GLOBAL (PRM)

Estudio Algoritmo PRM

Algoritmo PRM (Probabilistic RoadMap)

- Se basa en generar de forma aleatoria un conjunto de “nodos” desde el origen al destino y comprobar la conectividad (cuando no son bloqueados por un obstáculo) con los demás nodos para saber si se establece la conexión o no.
- Se selecciona el camino óptimo entre el origen y destino.



3. Planificación GLOBAL (PRM)

Script wander_localiza_planifica.m

Objetivo: Planificar la ruta para alcanzar el destino una vez se haya localizado el robot

Organigrama (punto de partida script wander_localiza.m)

- Definir la posición de destino `endLocation = [x y];`
- Navegar por el entorno hasta localizarse (wander_localiza)
- Crear el **objeto PRM** y ajustar sus parámetros
 - `planner = mobileRobotPRM;`
- Calcular la ruta al punto destino desde la posición actual del robot y mostrarla en una figura

3. Planificación GLOBAL (PRM)

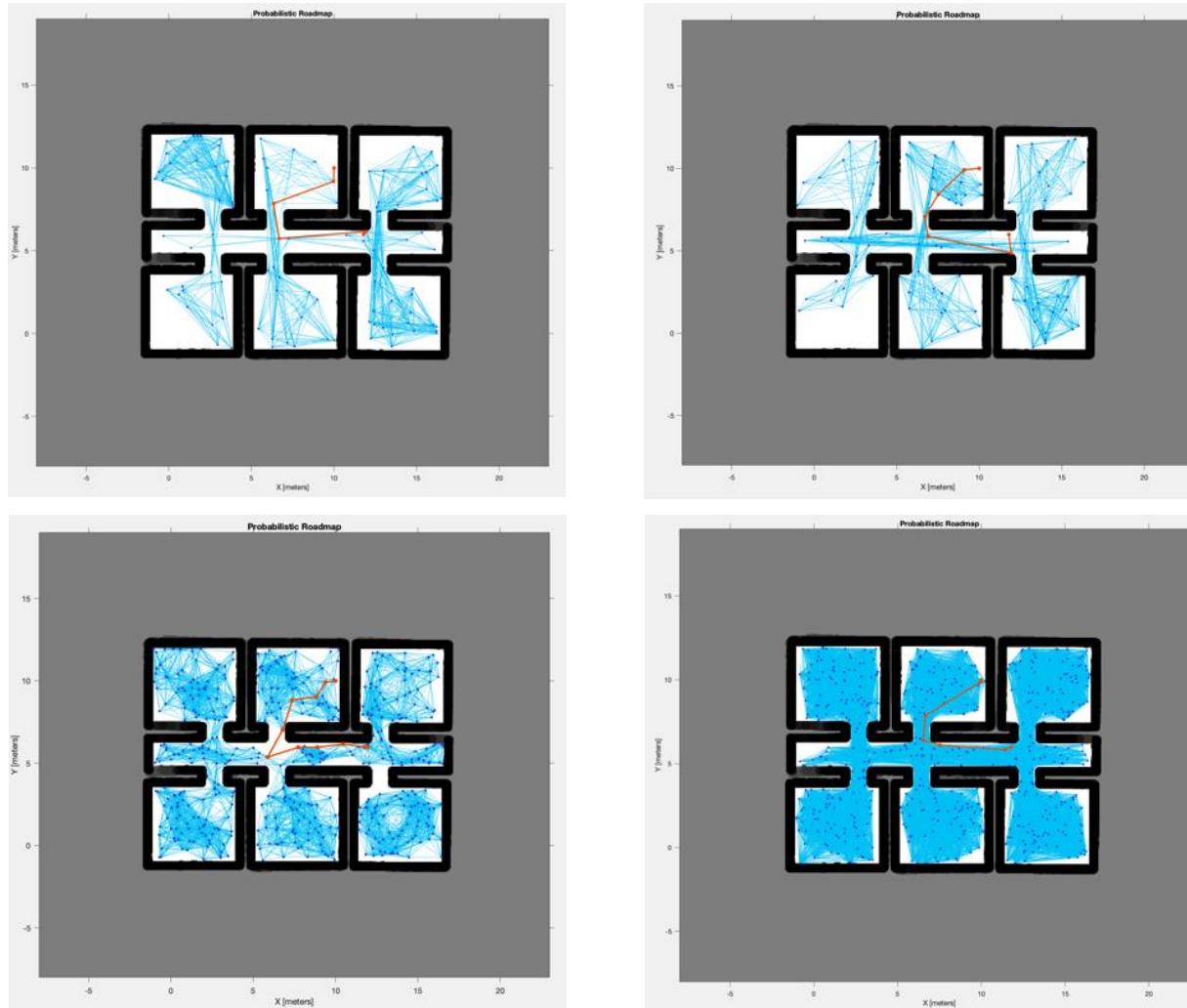
Script wander_localiza_planifica.m

PRM – PROBILISTIC ROAD MAP

- Cargar mapa y aumentar el tamaño para corregir “efecto tamaño del robot”
- Para localizar hay que seguir usando el mismo hacemos una copia
 - `cpMap = copy(map);`
 - `inflate(cpMap,0.25);`
- Crear el **objeto PRM** y ajustar sus parámetros
 - **`planner=mobileRobotPRM;`**
 - `planner.Map=cpMap;`
 - `planner.NumNodes=XnumNOdes;`
 - `planner.ConnectionDistance=Xmetros;`
- Calcular la ruta al punto destino desde la posición actual y mostrarla en figura
 - `ruta=findpath(planner,startLocation,endLocation);`
 - `figure; show(planner);`

3. Planificación GLOBAL (PRM) Script wander_localiza_planifica.m

PRM – PROBABILISTIC ROAD MAP



3. Planificación GLOBAL

Path Tracking Algorithm

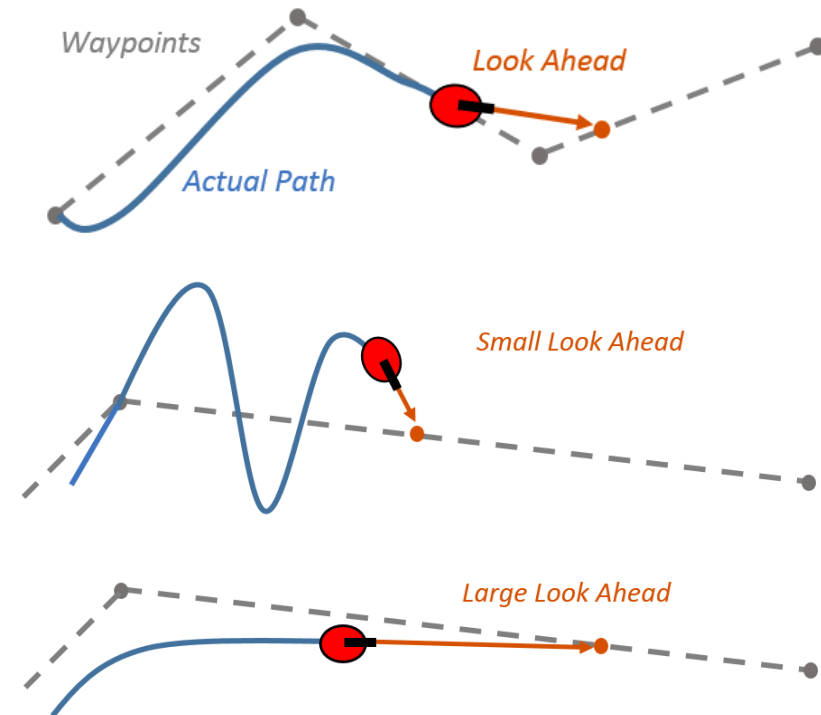
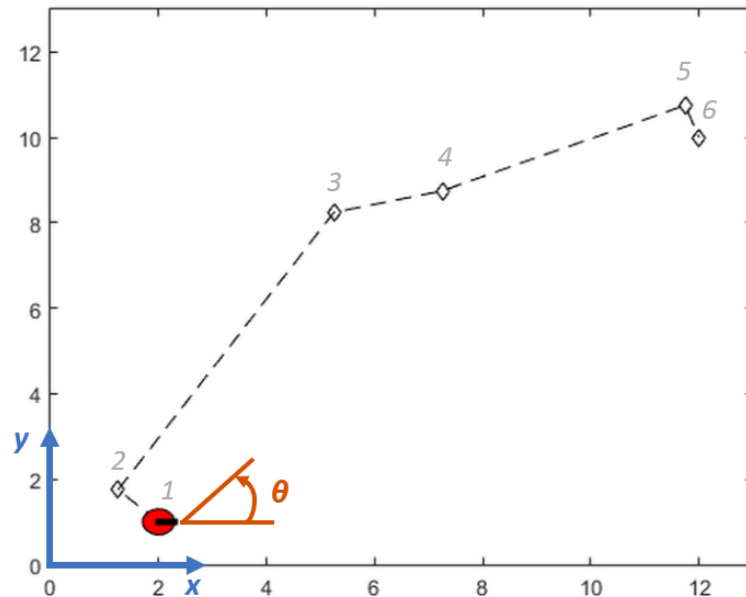
CONTROLADOR PURE PURSUIT

<https://es.mathworks.com/help/nav/ug/pure-pursuit-controller.html>

<https://es.mathworks.com/help/robotics/ug/pure-pursuit-controller.html>

- Pure Pursuit: algoritmo de seguimiento de rutas.
- Calcula la velocidad angular que desplaza al robot desde su posición actual para alcanzar un punto look-ahead situado delante de él.
- La velocidad lineal se asume constante,

Distancia look-ahead



Objetivo: Una vez planificada la ruta, el robot debe recorrerla utilizando un controlador *PurePursuit* (Controlador de persecución pura).

Organigrama (punto de partida script wander_localiza_planifica.m)

- Planificar la ruta (wander_localiza_planifica)
- Crear el **objeto PurePursuit** y ajustar sus propiedades
 - `CONTROLLER=controllerPurePursuit;`
- Pasar al controlador la lista de waypoints a recorrer (ruta)
- Bucle
 - Ejecutar el controlador PurePursuit para obtener las velocidades lineal y angular
 - Navegar hasta alcanzar el destino

CONTROLADOR PURE PURSUIT

<https://es.mathworks.com/help/robotics/ug/pure-pursuit-controller.html>

<https://es.mathworks.com/help/robotics/ref/controllerpurepursuit-system-object.html>

Crear el controlador

- `CONTROLLER=controllerPurePursuit;`

Ajustar propiedades

- `CONTROLLER.DesiredLinearVelocity= ...0.1;`
- `CONTROLLER.LookaheadDistance=...3;`
- `CONTROLLER.MaxAngularVelocity= ...0.5;`

CONTROLADOR PURE PURSUIT

Pasamos los Waypoints al controlador PurePursuit

- Ejecutar PRM y obtener ruta
- `CONTROLLER.Waypoints=ruta;`

Ejecutar Controlador PURE PURSUIT

- Calcular posición estimada con AMCL → `estimatedPose`
- `[lin_vel,ang_vel]=CONTROLLER(estimatedPose);`
- Publicar velocidad `send(pub_vel,msg_vel);`

¿¿¿Destino alcanzado??? → Distancia euclídea entre goal y pose actual

$$\text{sqrt}((\text{Goal.x-Pose.x})^2+(\text{Goal.y-Pose.y})^2) < \text{UMBRAL}$$

3. Planificación GLOBAL

navegacion_total.m

Objetivo: Incorporar el en bucle del controlador *PurePursuit* el evitador de obstáculos *VFH*.

Organigrama (punto de partida wander_localiza_planifica_control.m)

- Bucle control
 - Ejecutar el controlador PurePursuit para obtener las velocidades lineal y angular
 - Llamar al VFH pasándole como "targetDir" un valor proporcional a la velocidad angular calculada por el PurePursuit
 - Calcular la velocidad angular final como una combinación lineal de la generada por el controlador PurePursuit y la generada por VFH
 - Navegar hasta alcanzar el destino

VELOCIDAD LINEAL Y ANGULAR

Ejecutar el controlador PurePursuit

```
[lin_vel,ang_vel]=CONTROLLER(estimatedPose);
```

Corregir velocidad angular con el VFH

```
targetdir=K1*ang_vel;
```

```
direccion=VFH(scan,targetdir);
```

```
ang_vel_vfh=K2*direccion;
```

Combinar velocidades angulares

```
msg_vel.Linear.X=lin_vel;
```

```
msg_vel.Angular.Z=ang_vel_vfh; ///msg_vel.Angular.Z=ang_vel+ang_vel_vfh;
```

Publicar velocidad

```
send(pub_vel,msg_vel);
```

3. Planificación GLOBAL

navegacion_total.m

Vídeo navegación total

3. Planificación GLOBAL OPCIONAL

Planificadores en Matlab 2022a → Motion Planning

Use motion planning to plan a path through an environment.

https://es.mathworks.com/help/nav/motion-planning.html?s_tid=CRUX_topnav

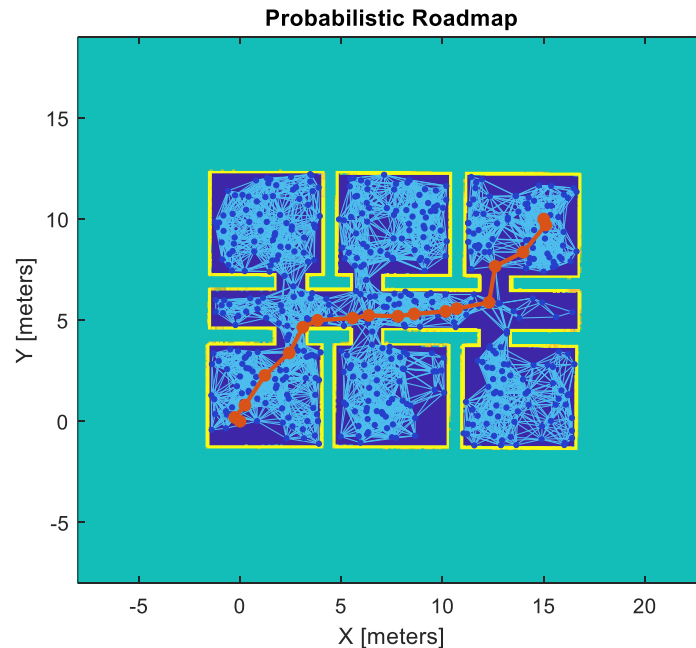
plannerRRT	Create an RRT planner for geometric planning
plannerRRTStar	Create an optimal RRT path planner (RRT*)
plannerBiRRT	Create bidirectional RRT planner for geometric planning
plannerControlRRT	Control-based RRT planner
plannerAStarGrid	A* path planner for grid map
plannerHybridAStar	Hybrid A* path planner
plannerPRM	Create probabilistic roadmap path planner
plannerBenchmark	Benchmark path planners using generated metrics

3. Planificación GLOBAL

Estudio Algoritmo PRM

- **Planner PRM:** Create probabilistic roadmap path planner
- The probabilistic roadmap path planner constructs a roadmap without start and goal states. Use the plan function to find an obstacle-free path between the specified start and goal states.

```
startPose=[0 0];  
goalPose=[15 10];
```



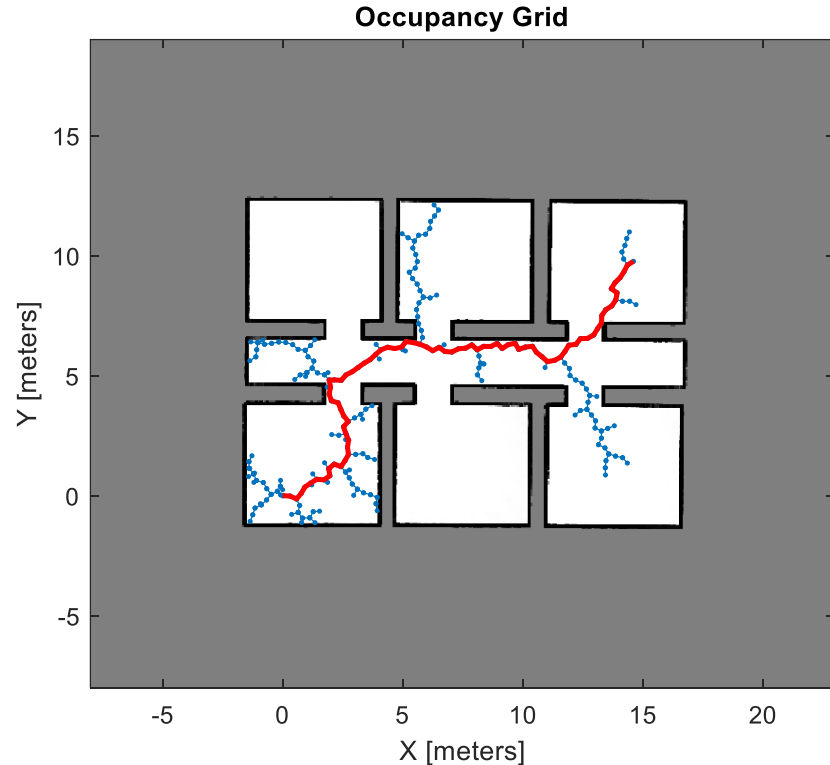
3. Planificación GLOBAL

Estudio Algoritmo RRT

Planner RRT: Create an RRT planner for geometric planning

Description

startPose=[0 0];
goalPose=[15 10];



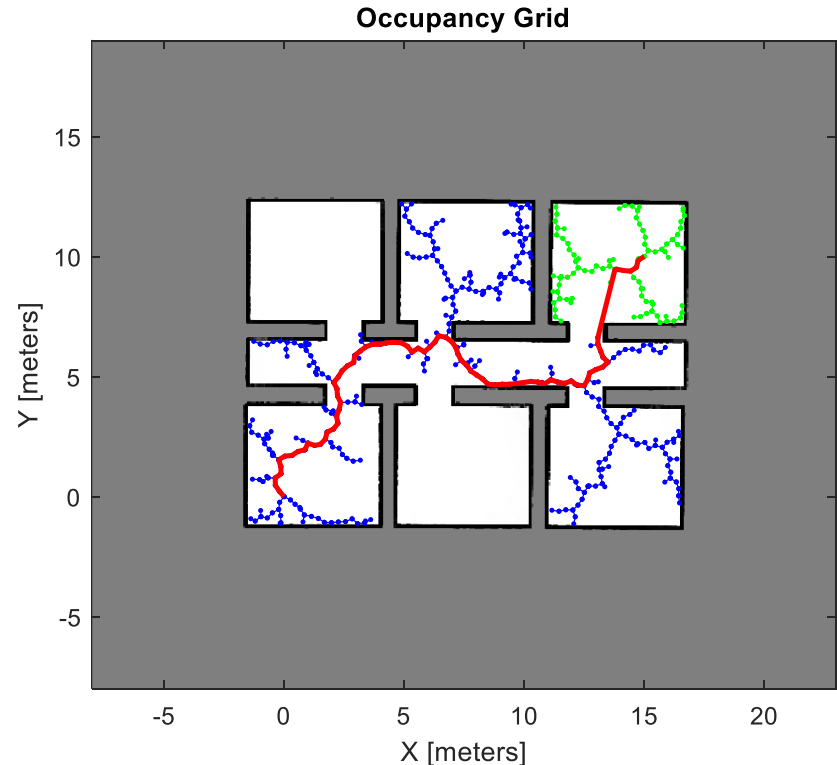
3. Planificación GLOBAL

Estudio Algoritmo BiRRT

Planner BiRRT: Create bidirectional RRT planner for geometric planning

Description

```
startPose=[0 0];  
goalPose=[15 10];
```



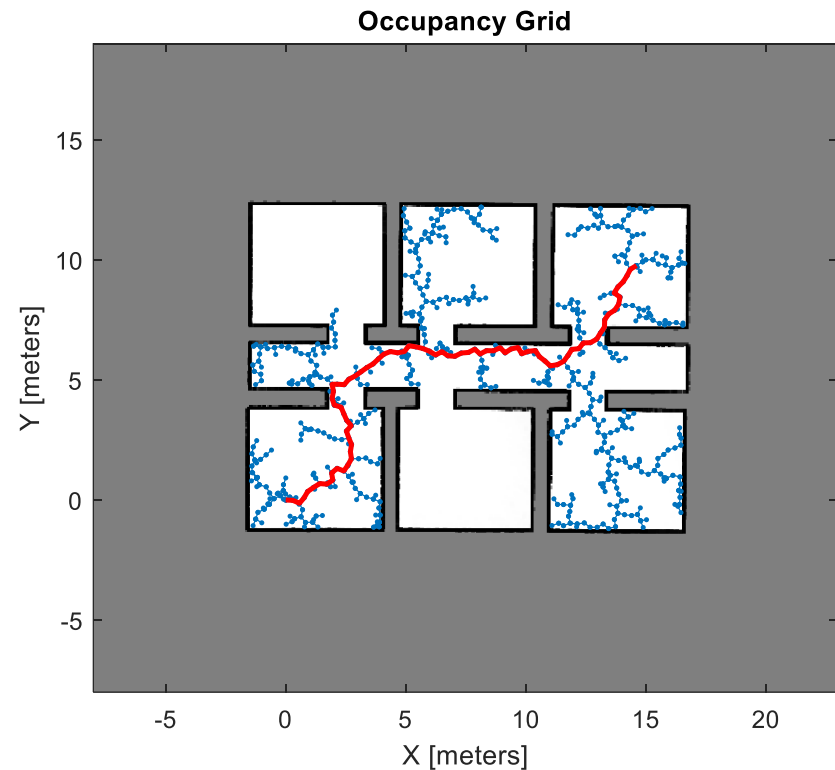
3. Planificación GLOBAL

Estudio Algoritmo RRTStar

Planner RRTStar: Create an optimal RRT path planner (RRT*)

Description

startPose=[0 0];
goalPose=[15 10];



3. Planificación GLOBAL

Estudio Algoritmo A* (A-star)

Planner AStar: Create an A* path planner

Description

- The planner AStarGrid object creates an A* path planner.
- The planner performs an A* search on an occupancy map and finds shortest obstacle-free path between the specified start and goal grid locations as determined **by heuristic cost**.

```
startPose=[0 0];  
goalPose=[15 10];
```

