Sistemas de Control para Robots

P1: Mapeado y Localización

Rafael Barea/Elena López

Índice

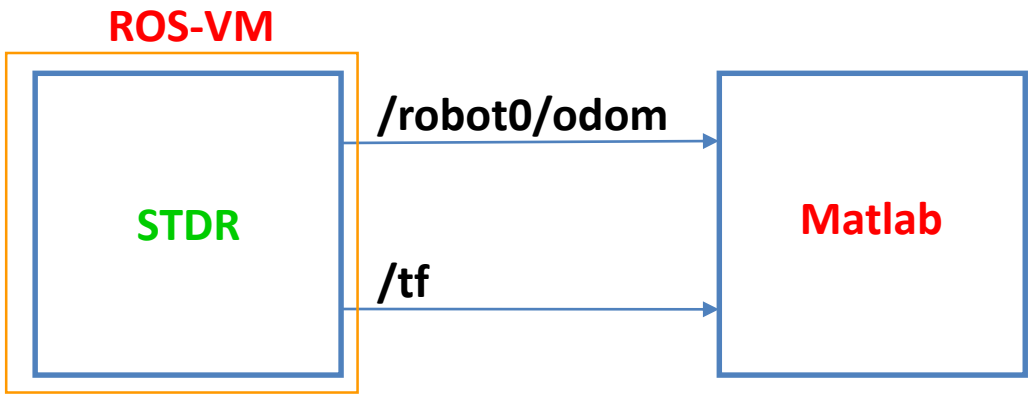
1. Instalación de paquetes
2. Estudio odometría del simulador STDR
   - Generación de odometría local
   - Incorporación ruido odometría STDR
3. Mapeado
   - Mapeado con posiciones conocidas
   - Localización y mapeado simultáneos (SLAM)
4. Localización AMCL

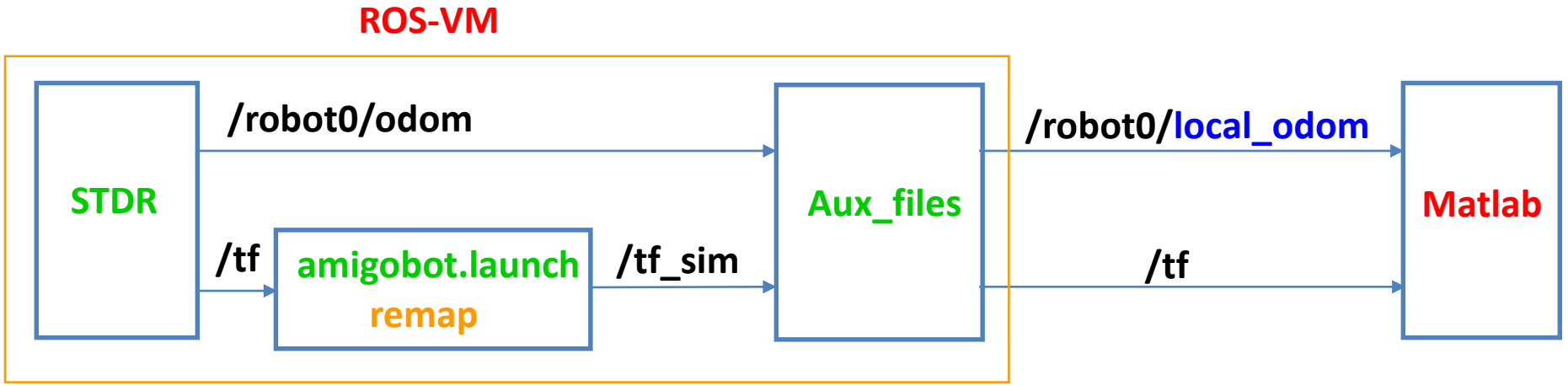
Instalación de paquetes ROS

- Paquetes necesarios para los filtros bayesianos
  - sudo apt-get install ros-kinetic-bfl

- Teleoperar el robot con un teclado
  - sudo apt-get install ros-kinetic-teleop-twist-keyboard

2. Estudio odometría del simulador STDR
Arquitectura del sistema





2. Estudio odometría del simulador STDR
Problemática simulador STDR

- El simulador STDR no es realista en cuanto a la odometría del robot:
  - El valor de la posición está referido al marco de referencia del mapa (valor absoluto). No es realista respecto al funcionamiento real de la odometría, que siempre comienza en el valor (0,0,0).
  - No incorpora ruido, cuando la presencia de un ruido acumulativo es una de las características más destacables del funcionamiento real de un sistema odométrico.

- Problema:
  - El topic /robot0/odom muestra la posición absoluta del robot respecto al mapa sin ningún tipo de error (como si se tratara de un sensor GPS ideal).

2. Estudio odometría del simulador STDR
Acciones a realizar

Acciones a realizar

1. Redirigir el árbol de transformadas del topic /tf a /tf_sim.

- Editar fichero *.launch* del simulador (amigobot.launch) y en todos los nodos haremos un *<remap from="tf" to="tf_sim"/>* del siguiente modo:

```
<launch>

        <include file="$(find stdr_robot)/launch/robot_manager.launch" />

        <node type="stdr_server_node" pkg="stdr_server" name="stdr_server" output="screen" args="$(find
stdr_resources)/maps/simple_rooms.yaml">
                <remap from="tf" to="tf_sim"/>
        </node>

        <node pkg="tf" type="static_transform_publisher" name="world2map" args="0 0 0 0 0 0  world map 100" >
                <remap from="tf" to="tf_sim"/>
        </node>

        <include file="$(find stdr_gui)/launch/stdr_gui.launch"/>

        <node pkg="stdr_robot" type="robot_handler" name="$(anon robot_spawn)" args="add $(find stdr_resources)/
resources/robots/amigobot.xml 2 2 0" >
                <remap from="tf" to="tf_sim"/>
        </node>

</launch>
```

- Editar los ficheros que están incluidos en este launch (mediante la instrucción include) y realizar la misma operación

2. Estudio odometría del simulador STDR
Acciones a realizar

Acciones a realizar

- **2. Instalar el paquete aux_files**
  - Modifica el valor de la odometría para que sea local
  - Añade ruido acumulativo
  - Publica el resultado en dos topics:
    - /robot0/local_odom -> odometría del robot
    - /tf -> sistemas de referencia de manera que coincidan con los del robot real
  - Instalación
    - Descargarlo de la página de la asignatura, descomprimirlo dentro de la carpeta */src*, y recompilar el espacio de trabajo ejecutando *catkin_make*.
  - Ejecución
    - Editar aux_files.launch y ajustar/comentar parámetros error_x, error_y y error_a.
    - **roslaunch aux_files aux_files.launch** (con el STDR ya abierto)
    - **Meterlo como include en amigobot.launch (mejor opción)**

2. Estudio odometría del simulador STDR
Acciones a realizar

Fichero amigobot.launch

*amigobot.launch: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
<launch>

        <include file="$(find stdr_robot)/launch/robot_manager.launch" />

        <node type="stdr_server_node" pkg="stdr_server" name="stdr_server" output="screen" args="$(find stdr_resources)/maps/simple_rooms.yaml">
                <remap from="tf" to="tf_sim"/>
        </node>

        <node pkg="tf" type="static_transform_publisher" name="world2map" args="0 0 0 0 0 0  world map 100" >
                <remap from="tf" to="tf_sim"/>
        </node>

        <include file="$(find stdr_gui)/launch/stdr_gui.launch"/>

        <node pkg="stdr_robot" type="robot_handler" name="$(anon robot_spawn)" args="add $(find stdr_resources)/resources/robots/amigobot.xml 4 8 0" >
                <remap from="tf" to="tf_sim"/>
        </node>

        <include file="/home/alumno/robotica_movil_ws/src/aux_files/launch/aux_files.launch"/>

</launch>
```

Añadir <remap from="tf" to="tf_sim"/>

Línea 21, columna 10 | 100% | UNIX (LF) | UTF-8

2. Estudio odometría del simulador STDR
Acciones a realizar

Fichero robot_manager.launch

*robot_manager.launch: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
<launch>
  <node pkg="nodelet" type="nodelet" name="robot_manager"  args="manager">
      <remap from="tf" to="tf_sim"/>
  </node>
</launch>
```

Fichero stdr_gui.launch

*stdr_gui.launch: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
<launch>
    <node name="$(anon stdr_gui_node)" pkg="stdr_gui" type="stdr_gui_node">
        <remap from="tf" to="tf_sim"/>
    </node>
</launch>
```

2. Estudio odometría del simulador STDR
Acciones a realizar

Fichero aux_files.launch (.../src/aux_files/launch/aux_files.launch)

- **Comportamiento con odometría ideal = ruido nulo**

```
<node pkg="aux_files" type="local_odom" name="local_odom" >
        <!--remap from="tf" to="tf_new"/> -->
        <param name="error_x" value="0.0"/>
        <param name="error_y" value="0.0"/>
        <param name="error_a" value="0.0"/>
        </node>
```

- **Odometría real = ruido**

```
<node pkg="aux_files" type="local_odom" name="local_odom" >
        <!--remap from="tf" to="tf_new"/> -->
        <!--param name="error_x" value="0.0"/> -->
        <!--param name="error_y" value="0.0"/> -->
        <!--param name="error_a" value="0.0"/> -->
        </node>
```

2. Estudio odometría del simulador STDR
Acciones a realizar

Configuración del mapa del simulador

- Utilizar mapa *simple_rooms.yaml*
  - ➢ Modificar el mapa que carga el fichero amigobot.launch

*amigobot.launch: Bloc de notas

Archivo Edición Formato Ver Ayuda

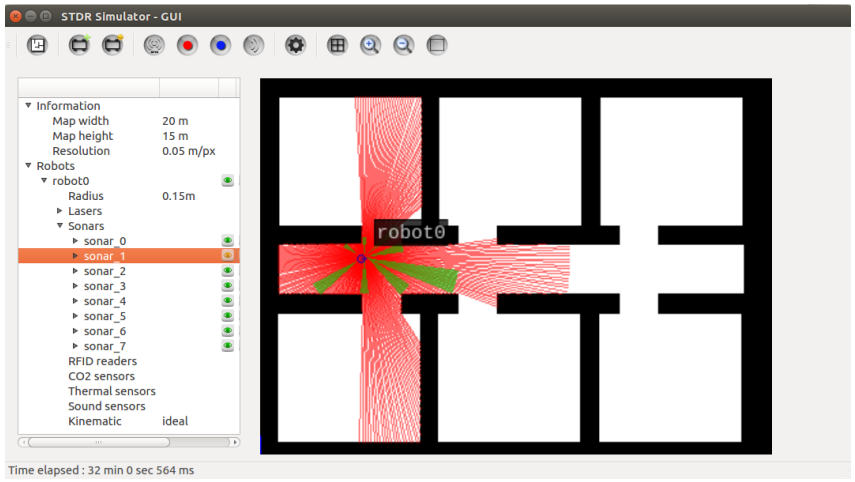
```
<launch>

        <include file="$(find stdr_robot)/launch/robot_manager.launch" />

        <node type="stdr_server_node" pkg="stdr_server" name="stdr_server" output="screen" args="$(find stdr_resources)/maps/simple_rooms.yaml">
        <!--<node type="stdr_server_node" pkg="stdr_server" name="stdr_server" output="screen" args="$(find stdr_resources)/maps/robocup.yaml">  -->
        <!--<node type="stdr_server_node" pkg="stdr_server" name="stdr_server" output="screen" args="$(find stdr_resources)/maps/robocup_ver2.yaml"> -->
                <remap from="tf" to="tf_sim"/>
        </node>
```

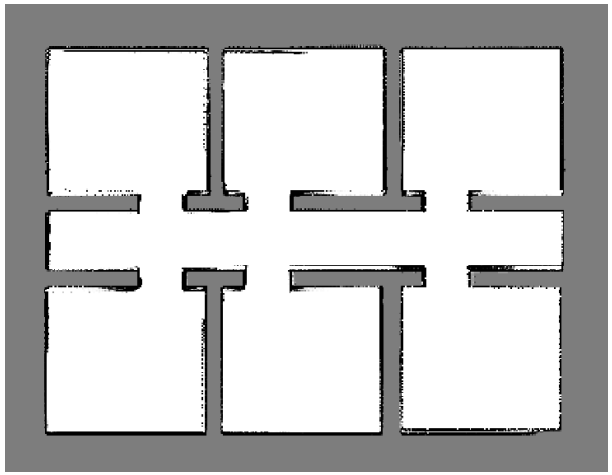
Línea 5, columna 1 | 100% | UNIX (LF) | UTF-8

3. Mapeado

Objetivo

- Obtener una mapa de ocupación del entorno del robot utilizando la información del láser y de la odometría.





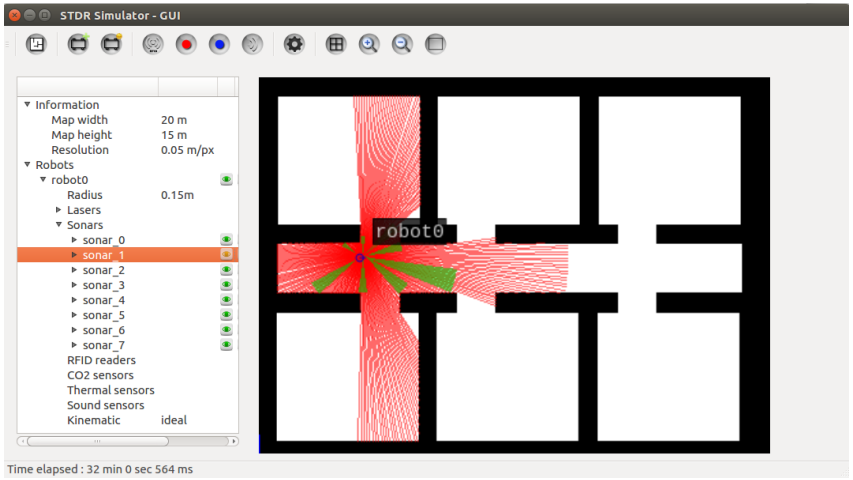
2 Técnicas

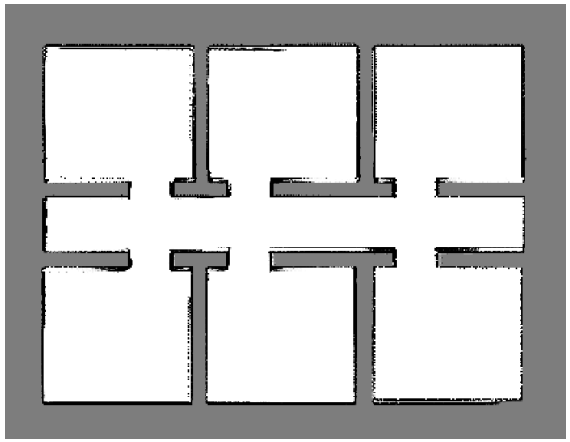
- Mapeado con posiciones conocidas
- Localización y mapeado simultáneos (SLAM)

3.1. Mapeado posiciones conocidas

Objetivo

- Obtener una mapa de ocupación del entorno del robot utilizando la información del láser y de la odometría con y sin errores en las medidas.





3.1. Mapeado posiciones conocidas

Pasos a seguir

Punto de partida → MappingWithKnownPoses_2022a.mlx

- Objetivos

  - Generar Script: MappingWithKnownPoses_Simulator.m
    - Mapeado del entorno en simulación configurando un error nulo para la odometría en el simulador STDR.
    - Mapeado del entorno en simulación CON errores en la odometría del robot.
      - ✓ Ajustar/comentar errores en aux_files.launch
    - Comparar resultados y obtener conclusiones

  - Generar script: MappingWithKnownPoses_RealRobot.m
    - Mapeado del entorno con el robot real (pasillo exterior al laboratorio)
    - Obtener conclusiones

3.1. Mapeado posiciones conocidas

Pasos a seguir - Código script

Punto de partida → MappingWithKnownPoses_2022a.mlx

1. Lectura del árbol de transformadas de ROS
   - tftree = rostf;

2. Definición de un mapa (rejilla) vacío.
   - Utilizar la clase occupancyMap

3. Obtener la posición del robot en el momento de la lectura del laser.
   - getTransform(tftree, 'source_frame', 'target_frame', msg_laser.Header.Stamp, 'Timeout', 2);
   - Consultar propiedades del 'tftree' o 'rqt' en ros para conocer "source_frame" y "target_frame"

3.1. Mapeado posiciones conocidas

Pasos a seguir - Código script

4. Extraer los rangos y los ángulos del mensaje del láser.
   - Consultar **LaserScan** (propiedades y funciones) en Matlab
     - ranges = ...
     - angles = ...

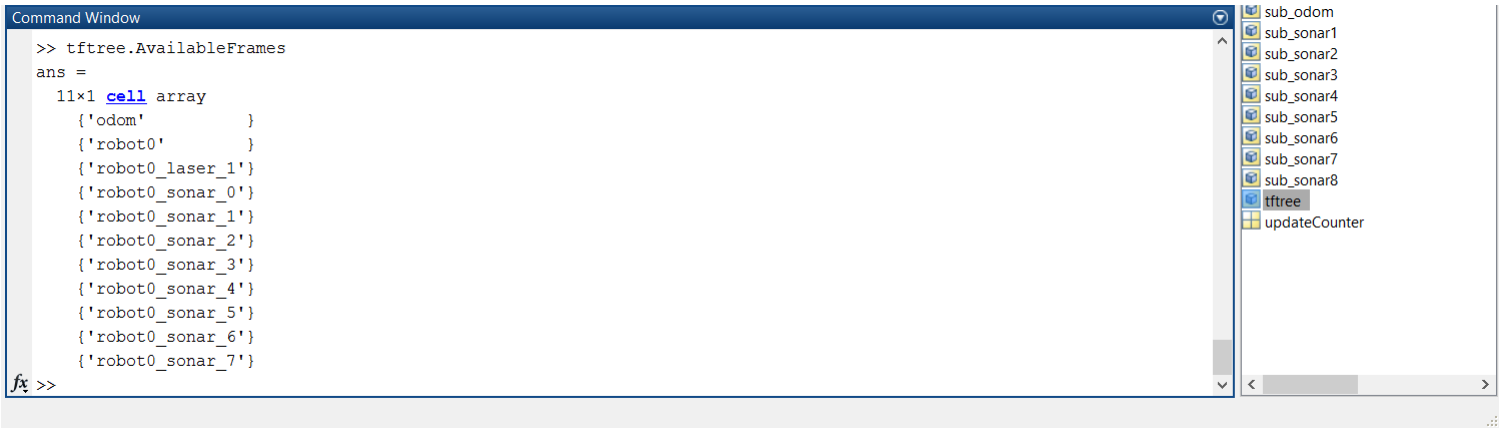
5. Insertar la medida del laser en el mapa
   - Consultar **insertRay**

3.1. Mapeado posiciones conocidas

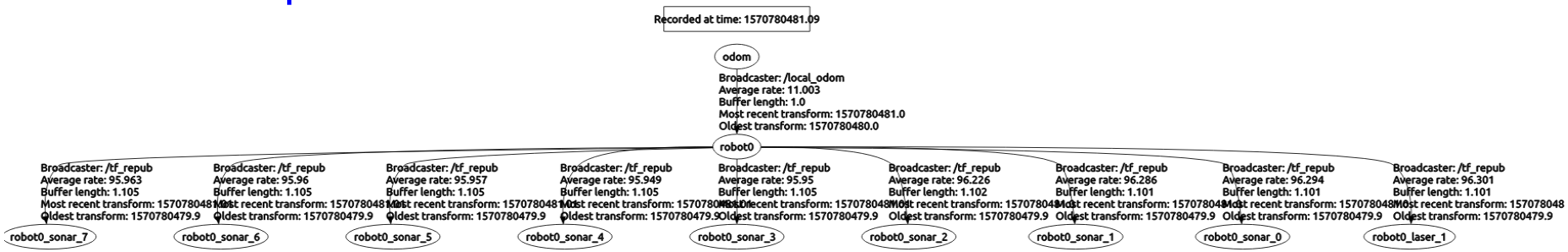
Comentarios

tftree = rostf Lectura del árbol de transformadas de ROS

- En Matlab → Propiedad tftree.AvailableFrames



- En ROS → rqt

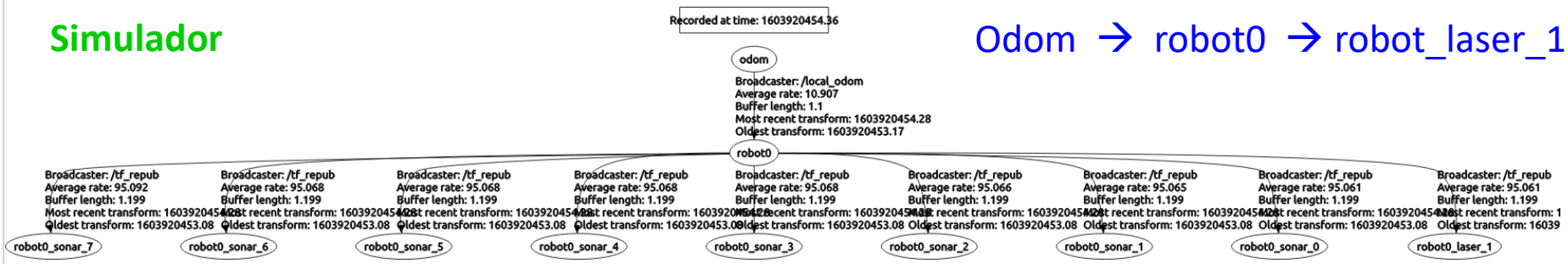


3.1. Mapeado posiciones conocidas

Comentarios

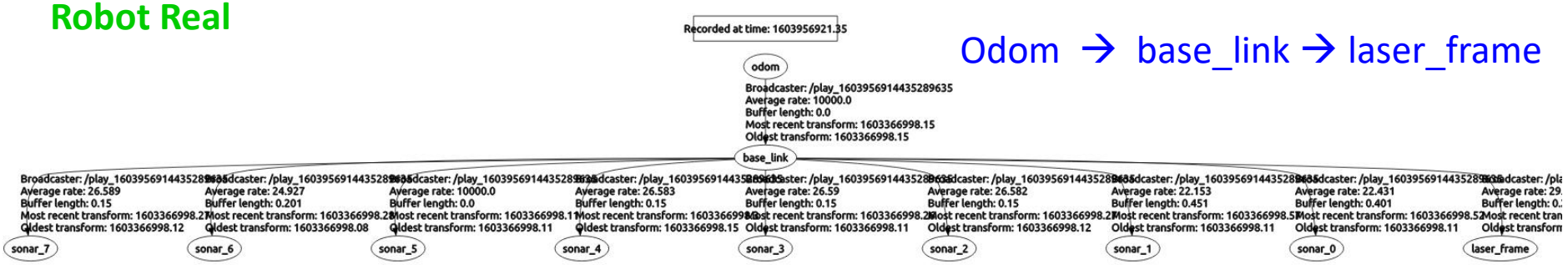
rqt

Simulador



Odom → robot0 → robot_laser_1

Robot Real



Odom → base_link → laser_frame

3.1. Mapeado posiciones conocidas
Comentarios

OccupancyMap clase

- Crea una rejilla de ocupación 2D con valores probabilísticos
  - map = occupancyMap(width,height,resolution)
    - width — Ancho del mapa (escalar en metros)
    - height — Altura del mapa (escalar en metros)
    - resolution — Resolución de cuadrícula (1 (predeterminado) | escalar en celdas por metro)

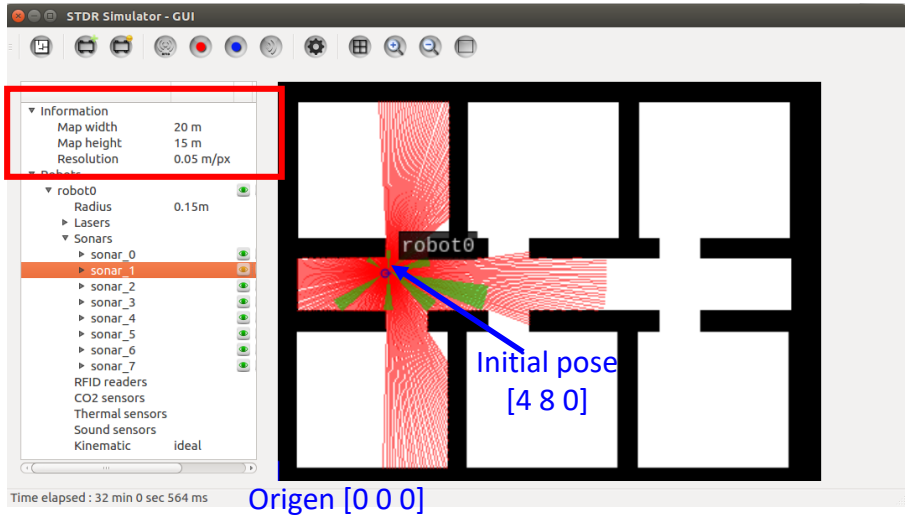
Propiedades

- GridLocationInWorld — coordenadas mundiales de la rejilla[x,y]
  - map.GridLocationInWorld=[-5 -10] Coordenadas del mundo de la esquina inferior izquierda de la cuadrícula, especificado como un vector de dos elementos.

3.1. Mapeado posiciones conocidas
Comentarios

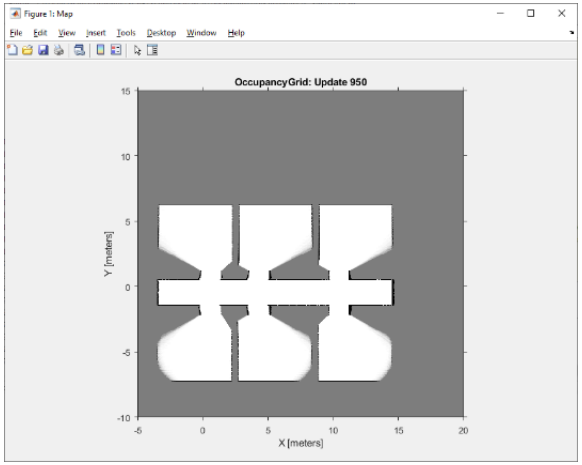
Clase occupancyMap

simple_rooms.png 400x300 pixels
Si resolución 0.05 m/píxel → 20x15 m

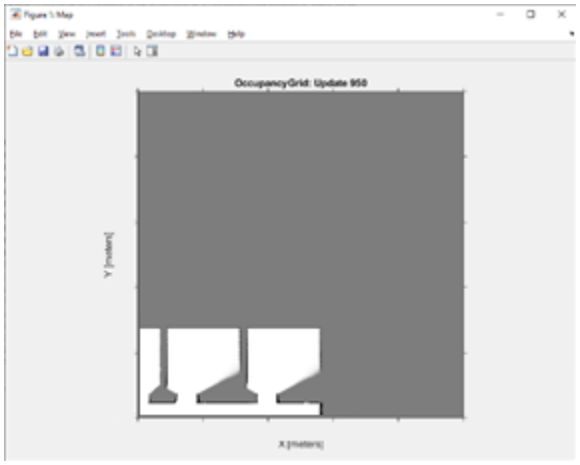


Origen [0 0 0]

occupancyMap(25,25,20)
GridLocationInWorld=[-5 -10]



occupancyMap(25,25,20)
GridLocationInWorld=[0 0]



3.1. Mapeado posiciones conocidas
Comentarios

Extract the ranges and angles from the laser scan message

```
sub_laser = rossubscriber('/robot0/laser_1', 'sensor_msgs/LaserScan');
msg_laser = sub_laser.LatestMessage;
```

Option 1:

```
ranges = msg_laser.Ranges;
angles = msg_laser.readScanAngles;
ranges(isinf(ranges)) = 8;
% Insert the laser range observation in the map
insertRay(map, robotPose, ranges, angles, 8);
```

Option 2:

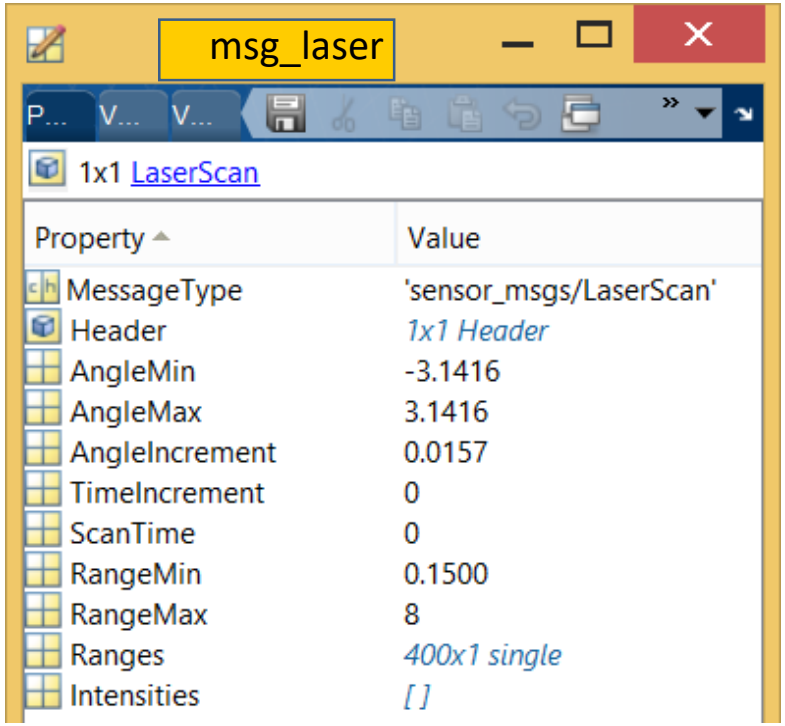
```
scans = lidarScan(msg_laser);
% Insert the laser range observation in the map
insertRay(map, robotPose, scans, 8);
```

3.1. Mapeado posiciones conocidas
Comentarios

LaserScan: Create laser scan message

```
sub_laser = rossubscriber('/robot0/laser_1', 'sensor_msgs/LaserScan');
msg_laser = sub_laser.LatestMessage;
```

Propiedades ROS Laser message



Funciones

lidarScan	Create object for storing 2-D lidar scan
plot	Display laser or lidar scan readings
readCartesian	Read laser scan ranges in Cartesian coordinates
readScanAngles	Return scan angles for laser scan range readings

Ejemplos: extraer rangos y ángulos

```
ranges = msg_laser.Ranges;
angles = msg_laser.readScanAngles;
```

3.1. Mapeado posiciones conocidas

Comentarios

lidarScan: Create object for storing 2-D lidar scan

scan = lidarScan(msg_laser) creates a lidarScan object from a LaserScan ROS message object.

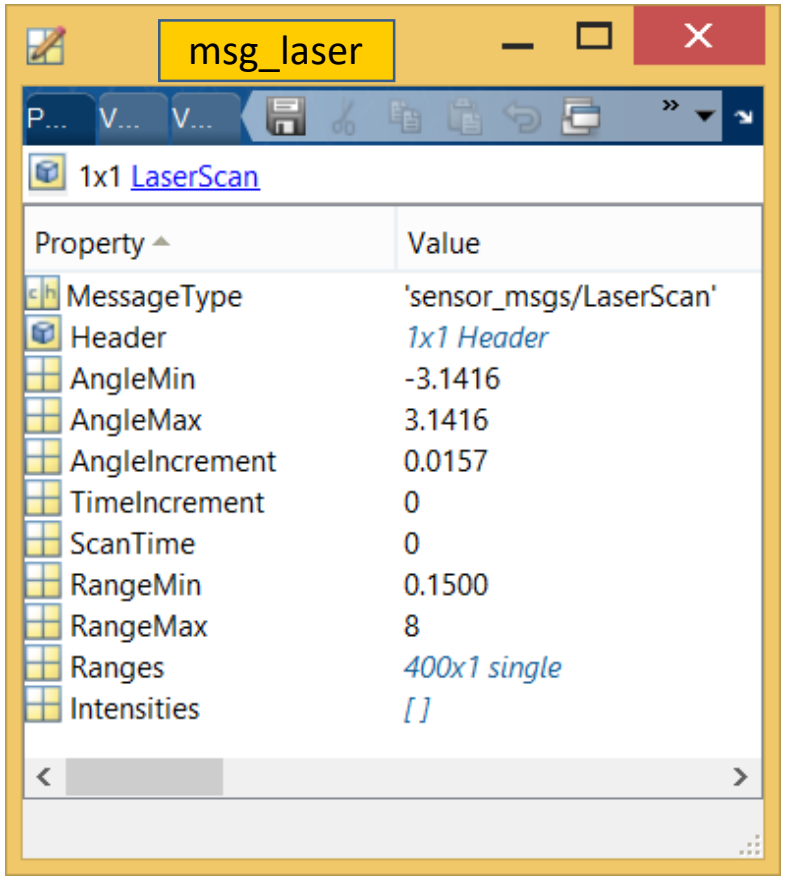
scan = lidarScan(ranges,angles) creates a lidarScan object from the ranges and angles, that represent the data collected from a lidar sensor.

3.1. Mapeado posiciones conocidas

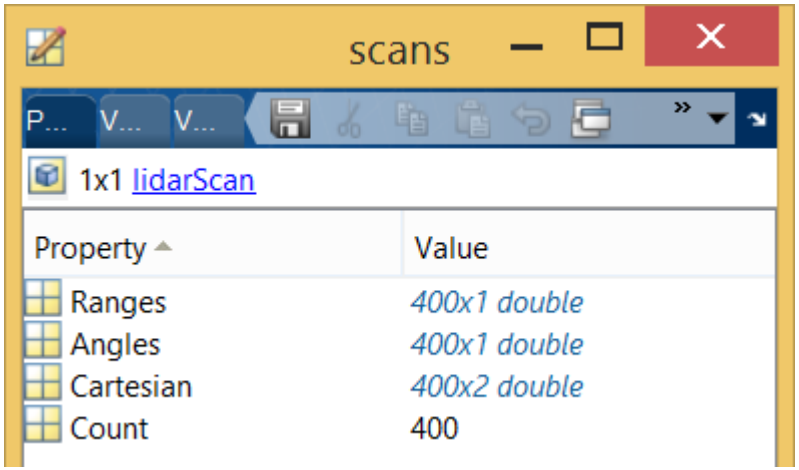
Comentarios

```
scans = lidarScan(msg_laser);
```

ROS Laser message



object for storing 2-D lidar scan



3.1. Mapeado posiciones conocidas
Comentarios

insertRay

- Insertar la medida del laser en el mapa

Option 1:

```
ranges = msg_laser.Ranges;
angles = msg_laser.readScanAngles;
ranges(isinf(ranges)) = 8;
% Insert the laser range observation in the map
insertRay(map, robotPose, ranges, angles, 8);
```

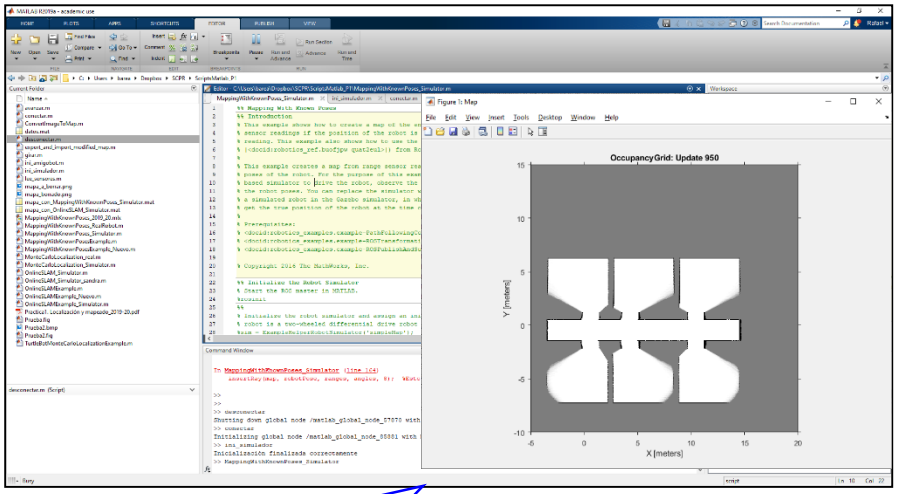
Option 2:

```
 scans = lidarScan(msg_laser);
% Insert the laser range observation in the map
insertRay(map, robotPose, scans, 8);
```

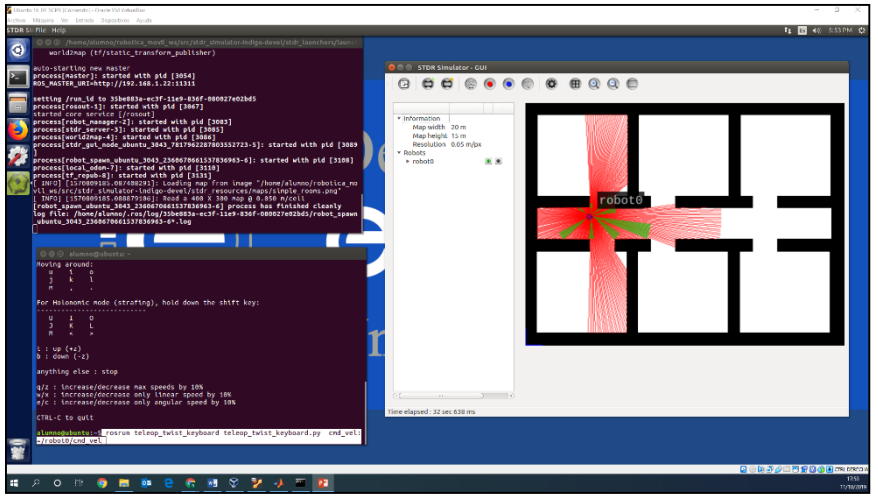
3.1. Mapeado posiciones conocidas

Simulación

PC1 Scripts matlab



PC2 Máquina virtual (MASTER)



1. Conectar.m
2. Ini_simulador.m
3. MappingWithKnownPoses_Simulator.m

T1: roslaunch stdr_launchers amigobot.launch

T2: rosrun teleop_twist_keyboard teleop_twist_keyboard.py cmd_vel:=/robot0/cmd_vel

3.1. Mapeado posiciones conocidas
Simulación

Mapa *simple_rooms*

En amigobot.launch

*amigobot.launch × *Untitled Document 1

```
<launch>

        <include file="$(find stdr_robot)/launch/robot_manager.launch" />

        <node type="stdr_server_node" pkg="stdr_server" name="stdr_server" output="screen" args="$(find stdr_resources)/maps/simple_rooms.yaml">
                <remap from="tf" to="tf_sim"/>
        </node>

        <node pkg="tf" type="static_transform_publisher" name="world2map" args="0 0 0 0 0 0  world map 100" >
                <remap from="tf" to="tf_sim"/>
        </node>

        <include file="$(find stdr_gui)/launch/stdr_gui.launch"/>
        <node pkg="stdr_robot" type="robot_handler" name="$(anon robot_spawn)" args="add $(find stdr_resources)/resources/robots/amigobot.xml 4 8 0" >
                <remap from="tf" to="tf_sim"/>
        </node>

         <include file="/home/alumno/robotica_movil_ws/src/aux_files/launch/aux_files.launch"/>
```

Plain Text ▾ Tab Width: 8 ▾ Ln 4, Col 1

En simple_rooms.yaml

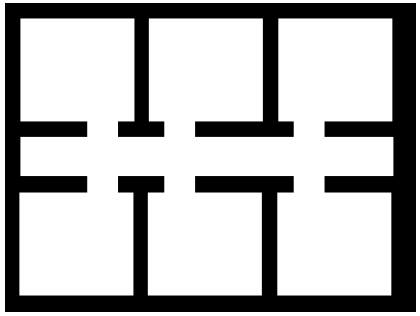
simple_rooms.yaml (~/robotica_movil_ws/src/stdr_simulator-indigo-

Open ▾ Save

```
image: simple_rooms.png
resolution: 0.05
origin: [0.0, 0.0, 0.0]
occupied_thresh: 0.6
free_thresh: 0.3
negate: 0
```

YAML ▾ Tab Width: 8 ▾ Ln 6, Col 10 ▾ INS

simple_rooms.png



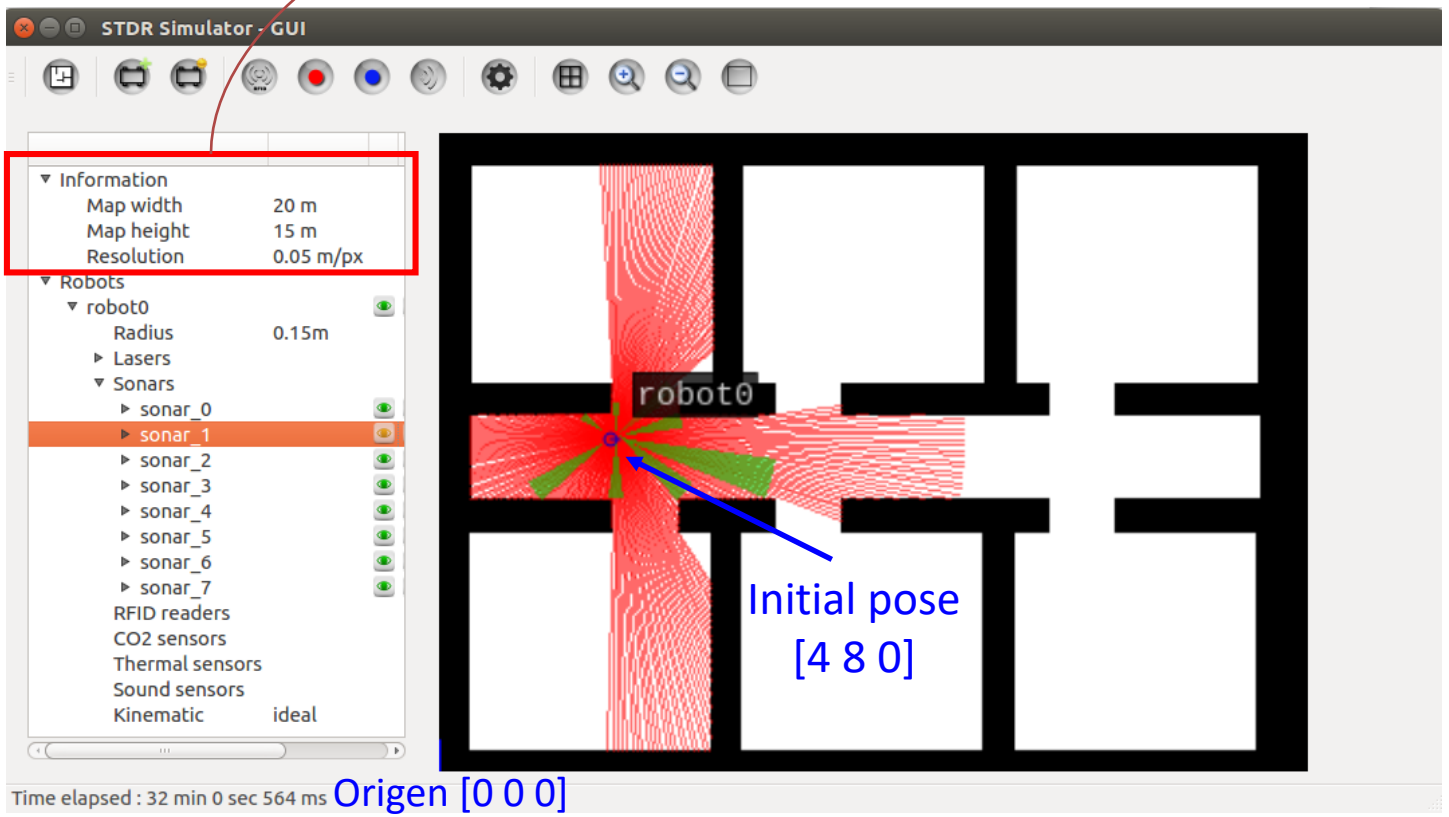
400x300 pixels

3.1. Mapeado posiciones conocidas
Simulación

Mapa *simple_rooms*

En STDR simulator

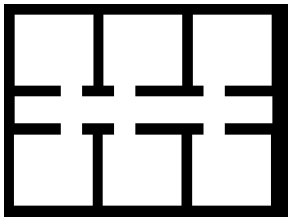
simple_rooms.png 400x300 pixels
Si resolución 0.05 m/píxel → 20x15 m



3.1. Mapeado posiciones conocidas

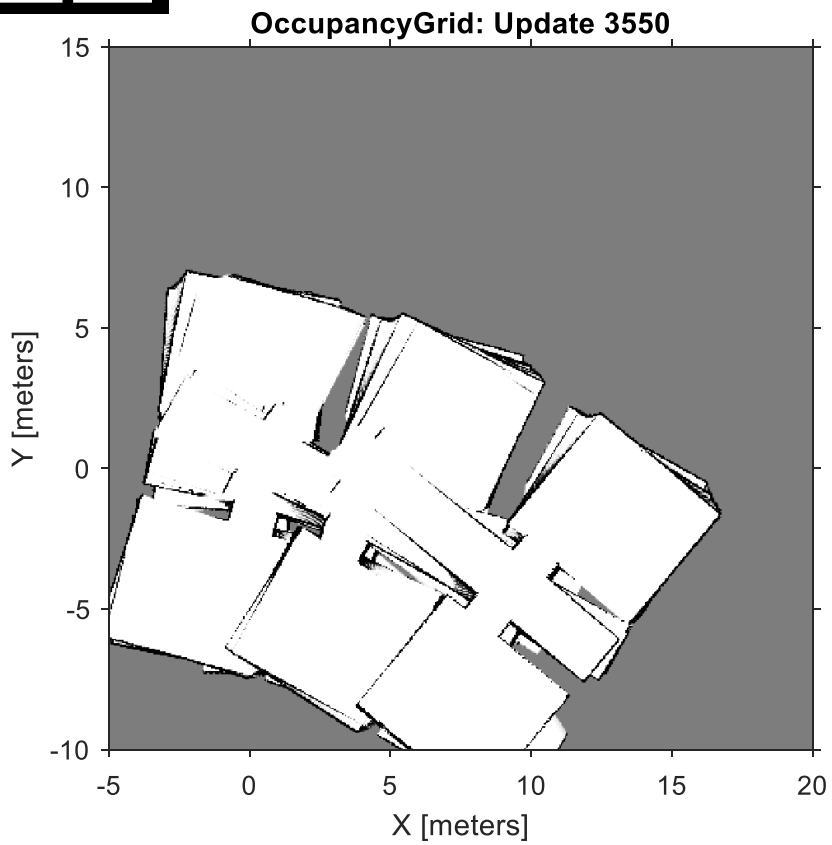
Resultados simulación

Resultados mapa *simple_rooms*



sin ruido

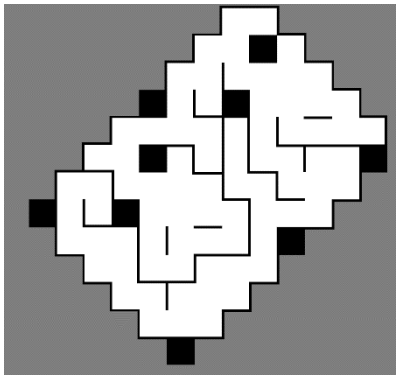
con ruido

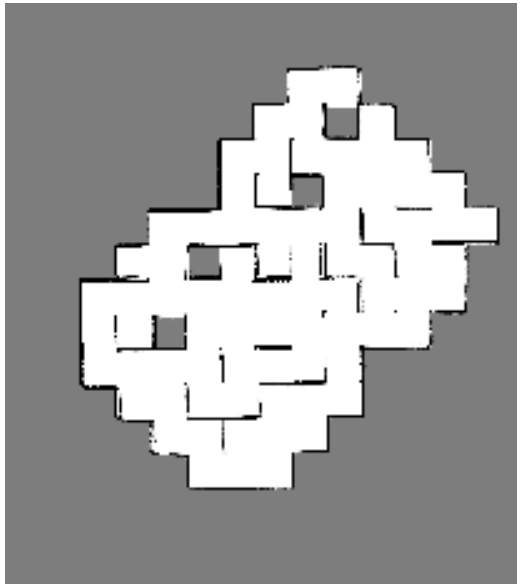


3.1. Mapeado posiciones conocidas

Resultados simulación

mapa *robocup_ver2*





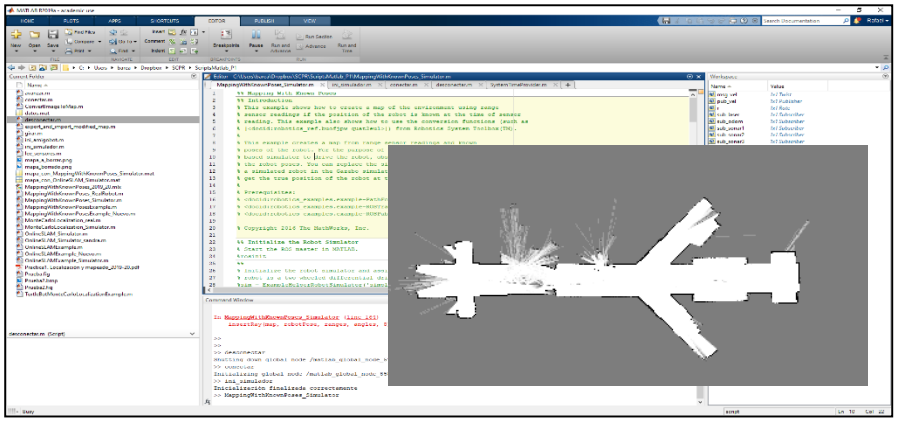
sin ruido

con ruido

3.1. Mapeado posiciones conocidas

Robot Real

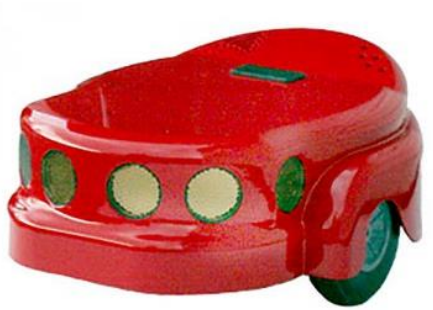
PC1 Scripts matlab



PC2 Máquina virtual

1. Conectar.m
2. Ini_amigobot.m
3. MappingWithKnownPoses_RealRobot.m

T1: rosrun teleop_twist_keyboard teleop_twist_keyboard.py
~~cmd_vel:=/robot0/cmd_vel~~??????????????????



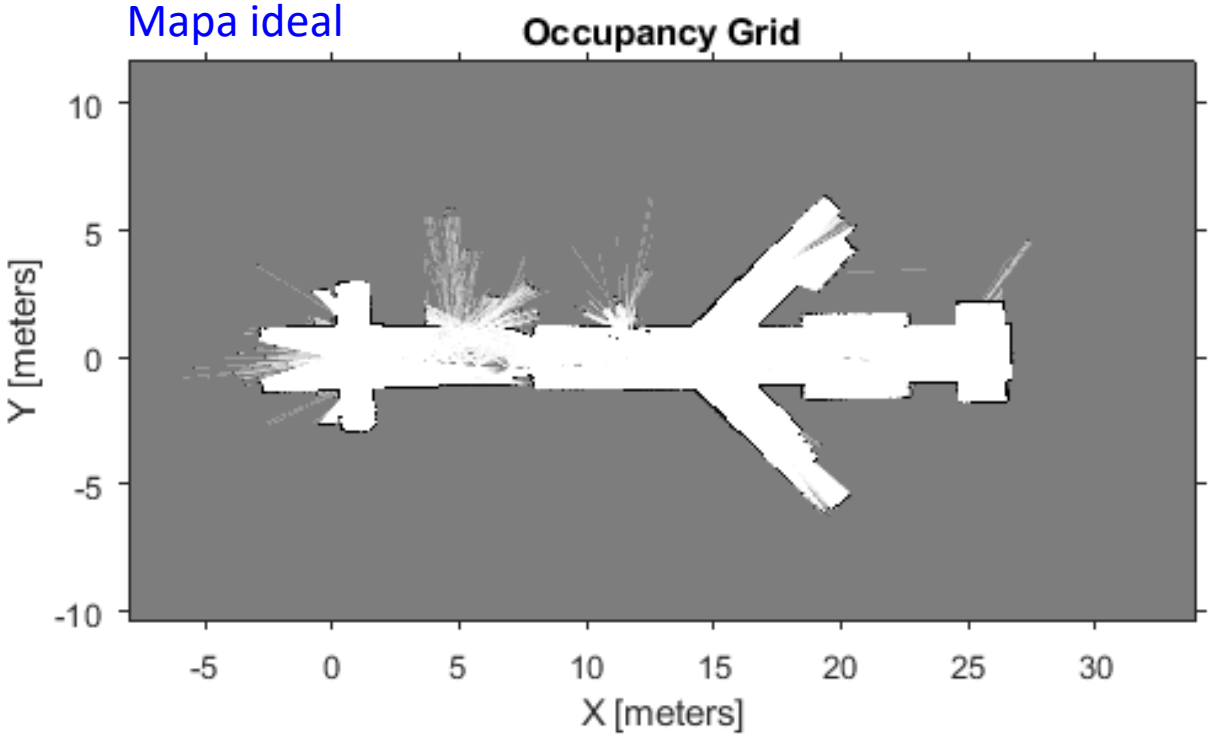
AmigoBot (MASTER)

3.1. Mapeado posiciones conocidas

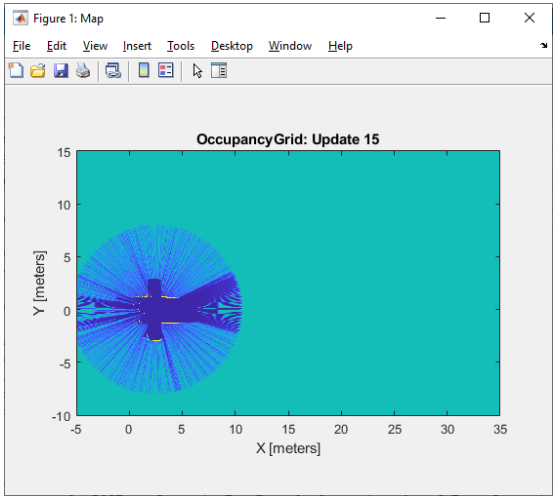
Resultados Robot Real

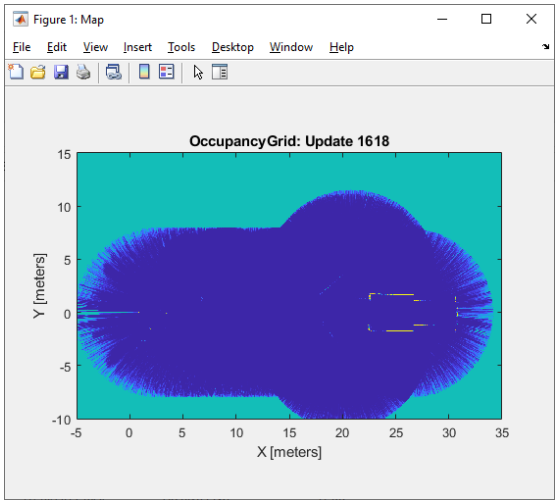
Pasillo laboratorio

Mapa ideal



Problema: mapa real generado

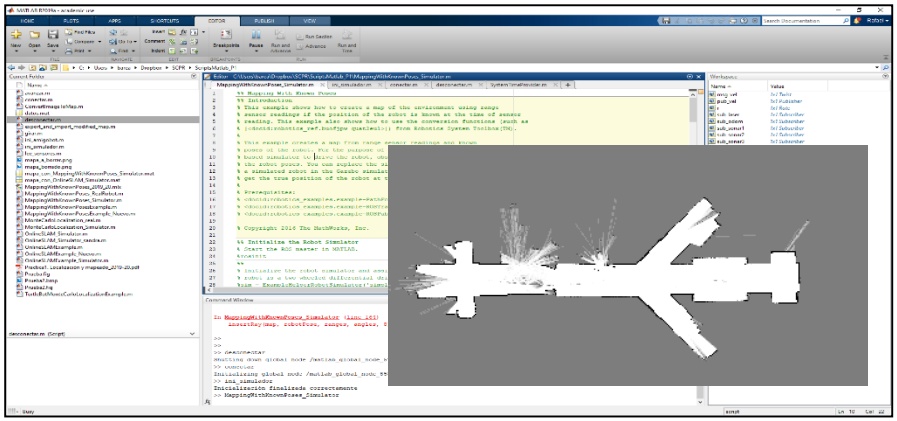




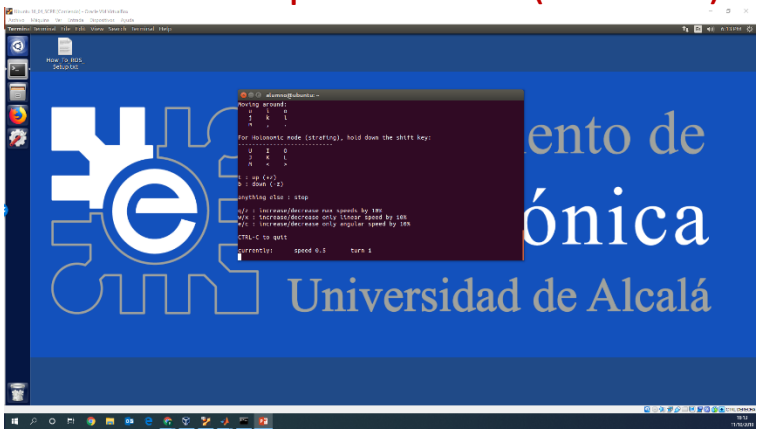
3.1. Mapeado posiciones conocidas

Rosbag

PC1 Scripts matlab



PC2 Máquina virtual (MASTER)

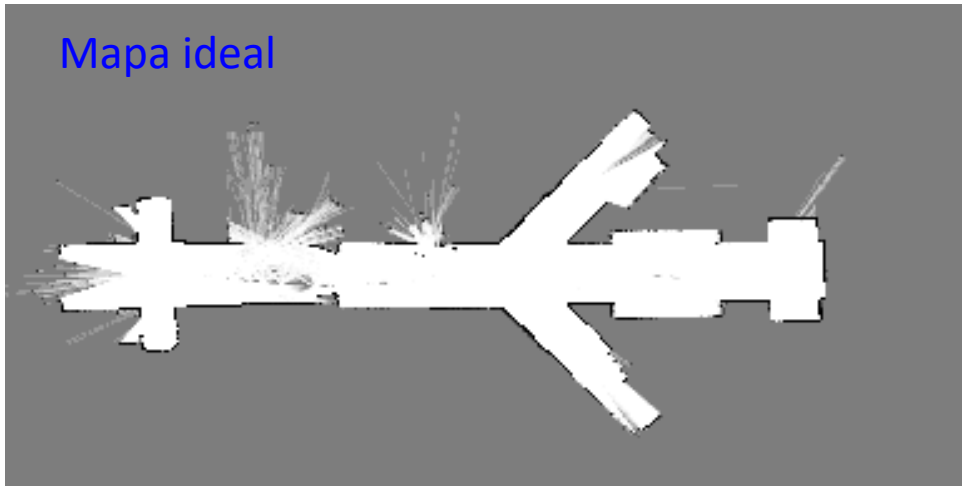


1. Conectar.m
2. Ini_amigobot.m
3. MappingWithKnownPoses_RealRobot.m

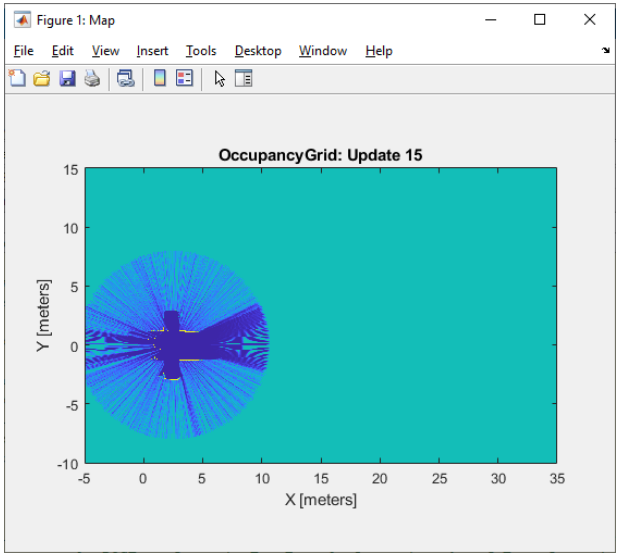
T1: roscore
T2: rosbag play rosbag_real_pasillo.bag

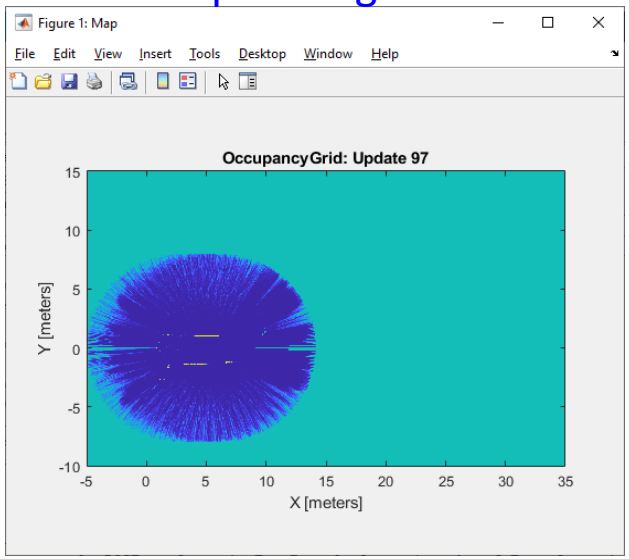
3.1. Mapeado posiciones conocidas

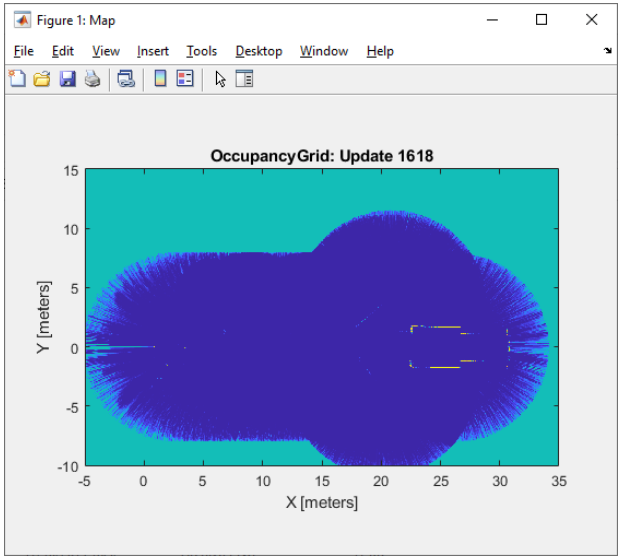
Rosbag - Problemas



Mapa real generado







3.2. SLAM
Localización y Mapeado Simultáneo

Punto de partida → OnlineSLAM_2022a.mlx

- Objetivos

  - Generar Script: OnlineSLAM_2022a_Simulator.m
    - Mapeado del entorno en simulación configurando un error nulo para la odometría en el simulador STDR.
    - Mapeado del entorno en simulación CON errores en la odometría del robot.
      - ✓ Ajustar/comentar errores en aux_files.launch
    - Comparar resultados y obtener conclusiones

  - Generar script: OnlineSLAM_2022a_RealRobot.m
    - Mapeado del entorno con el robot real (pasillo exterior al laboratorio)
    - Obtener conclusiones

3.2. SLAM
Localización y Mapeado Simultáneo

Punto de partida → OnlineSLAM_2022a.mlx

- Create Lidar Slam Object
  - slamObj = lidarSLAM(mapResolution, maxLidarRange);
- Create lidarScan object to pass to the Lidar-Slam object.
  - scans = lidarScan(msg_laser)
- Observe the Effect of Loop Closure and Optimization Process
  - addScan(slamObj, scans);
- Build Occupancy Grid Map
  - map = buildMap(scans, optimizedPoses, mapResolution, maxLidarRange);
- Visualize the occupancy grid map
  - show(map);

3.2. SLAM
Funciones

robotics.LidarSLAM clase → Create Lidar Slam Object (AMCL object)

- Realiza la localización y mapeo simultáneos (SLAM) para las entradas del sensor LiDAR.

  - slamObj = lidarSLAM(mapResolution, maxLidarRange)
    - mapResolution — Resolución mapa cuadrícula de ocupación (celdas por metro)
    - maxLidarRange - Rango máximo del sensor LiDAR (en metros)

- Otras Propiedades

  - slamObj.LoopClosureThreshold = 200; %Umbral para aceptar cierres de bucle
  - slamObj.LoopClosureSearchRadius = 3; %Radio de búsqueda para detección de cierre de bucle

3.2. SLAM
Funciones

lidarScan → Create object for storing 2-D lidar scan.

- msg_laser = sub_laser.LatestMessage;
- scans = lidarScan(msg_laser);

addScan → Add scan to lidar SLAM map.

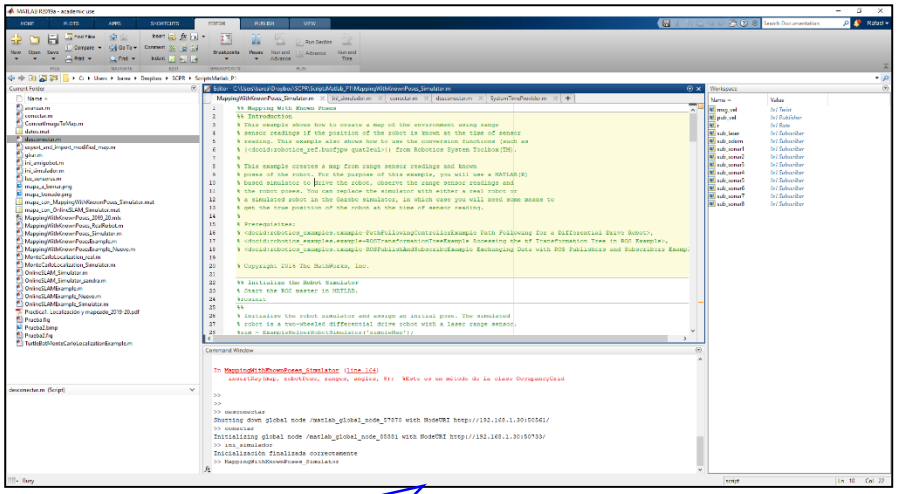
- [isScanAccepted, loopClosureInfo, optimizationInfo] = addScan(slamObj, scans);

show(slamObj)

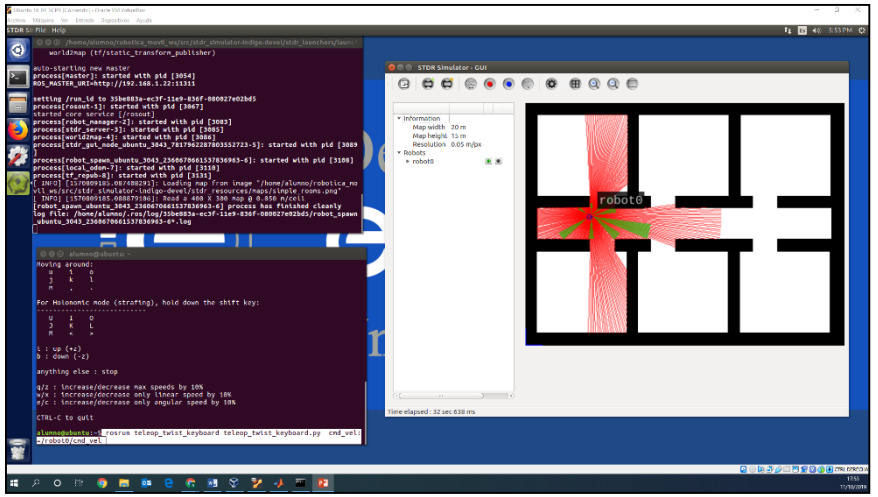
- Visualize how scans plot and poses are updated as robot navigate through virtual scene

3.2. SLAM
Simulación

PC1 Scripts matlab



PC2 Máquina virtual (MASTER)



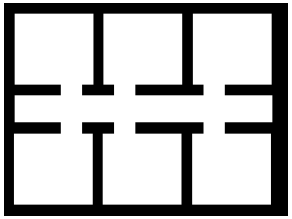
1. Conectar.m
2. Ini_simulador.m
3. OnlineSLAM_Simulator.m

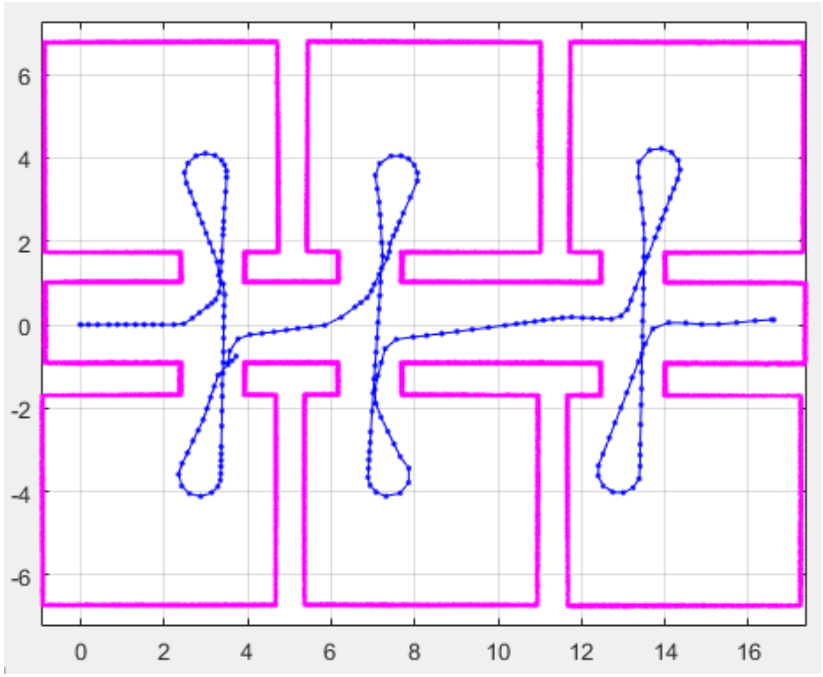
T1: roslaunch stdr_launchers amigobot.launch
T2: rosrun teleop_twist_keyboard teleop_twist_keyboard.py cmd_vel:=/robot0/cmd_vel

3.2. SLAM
Simulación

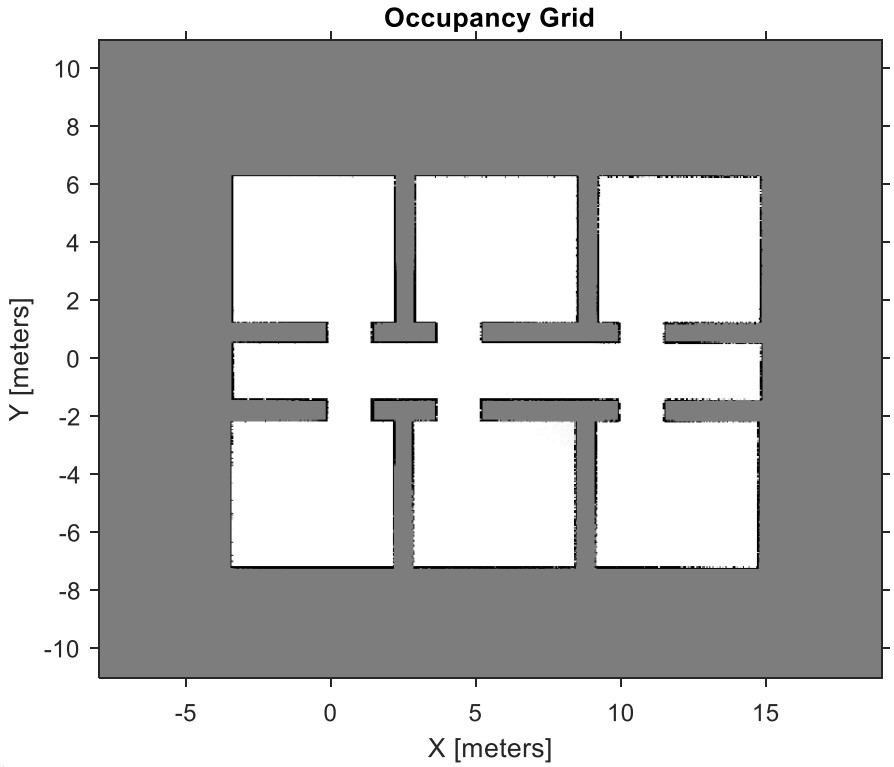
mapa *simple_rooms*

con ruido





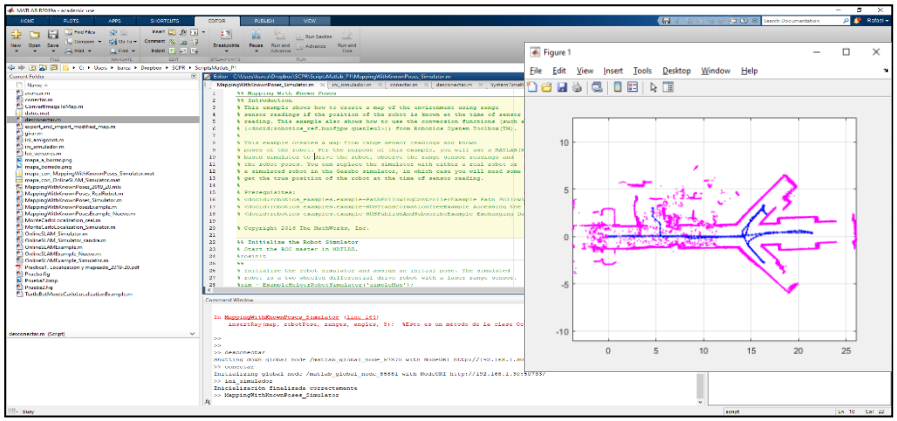
Final built map after all scans are added to the slamObj object



Occupancy Grid Map

3.2. SLAM
Robot Real

PC1 Scripts matlab



PC2 Máquina virtual

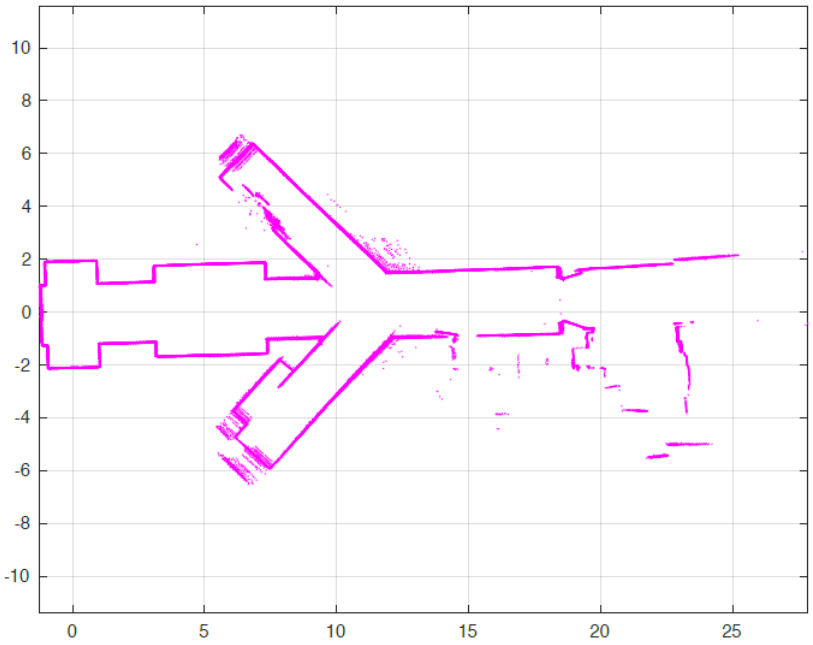
1. Conectar.m
2. Ini_amigobot.m
3. . OnlineSLAM_RealRobot.m

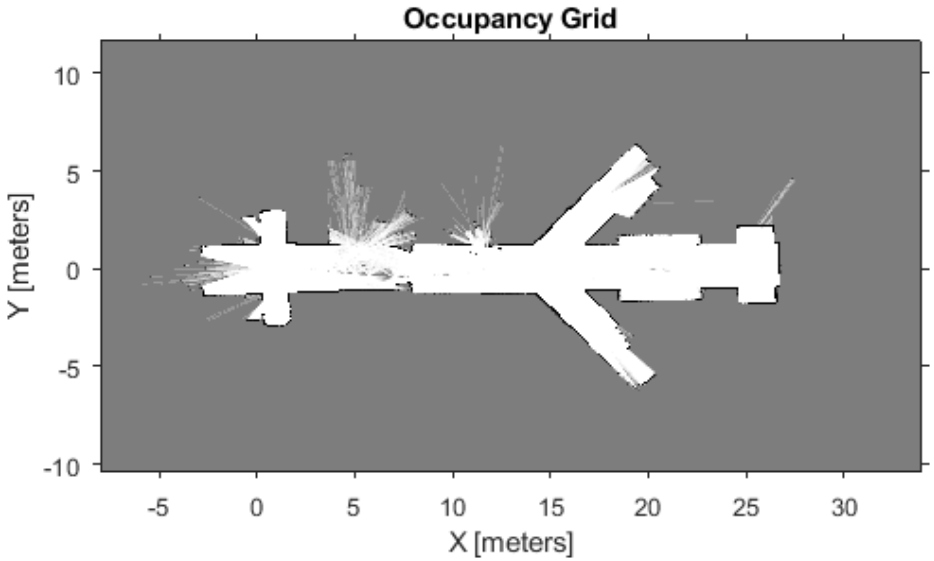
T1: rosrun teleop_twist_keyboard teleop_twist_keyboard.py
~~cmd_vel:=/robot0/cmd_vel~~??????????????????

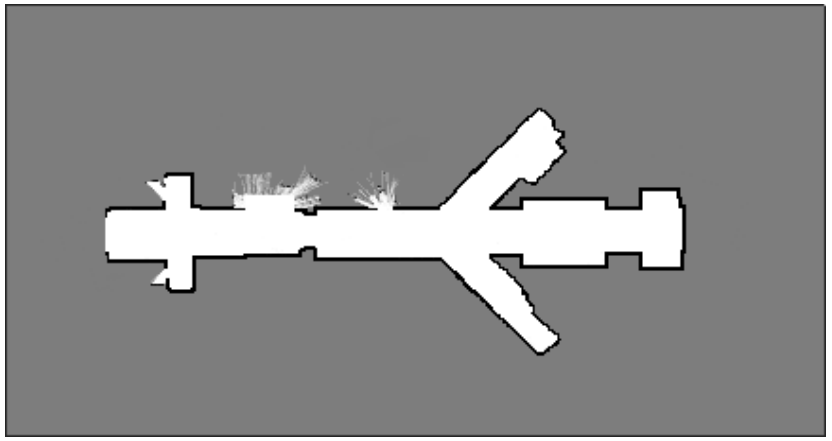
AmigoBot (MASTER)

3.2. SLAM
Robot Real

Pasillo laboratorio

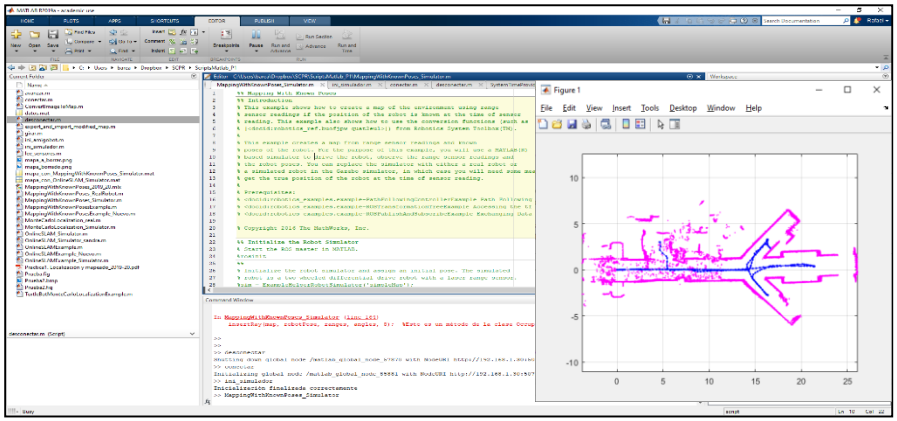




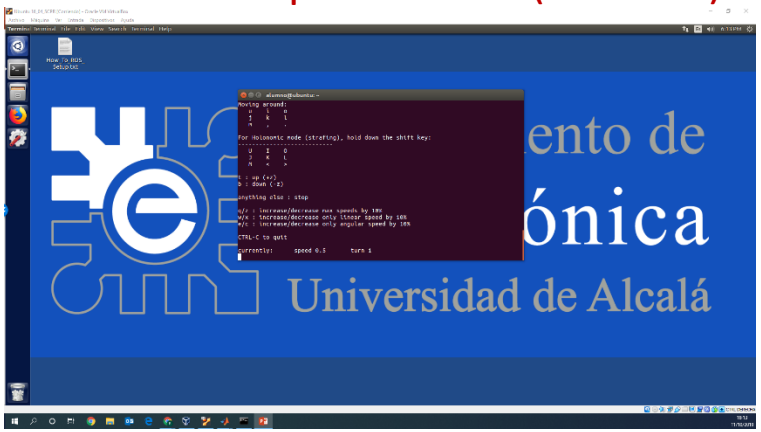


3.2. SLAM
Rosbag

PC1 Scripts matlab



PC2 Máquina virtual (MASTER)

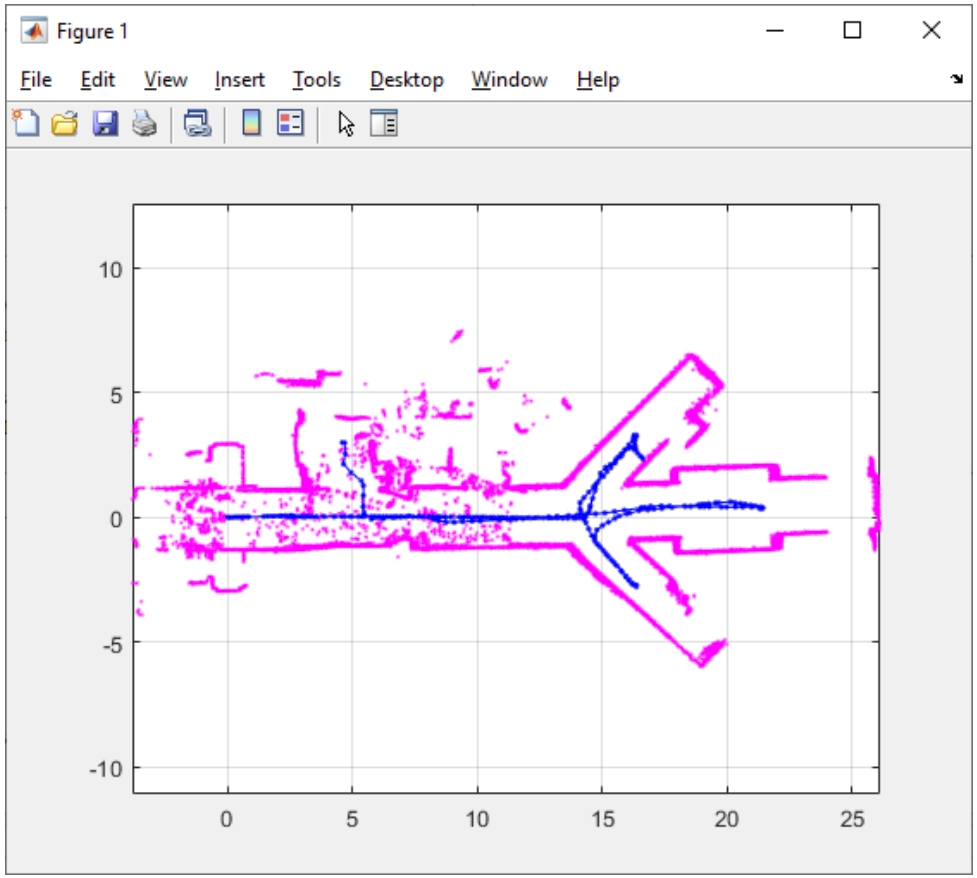


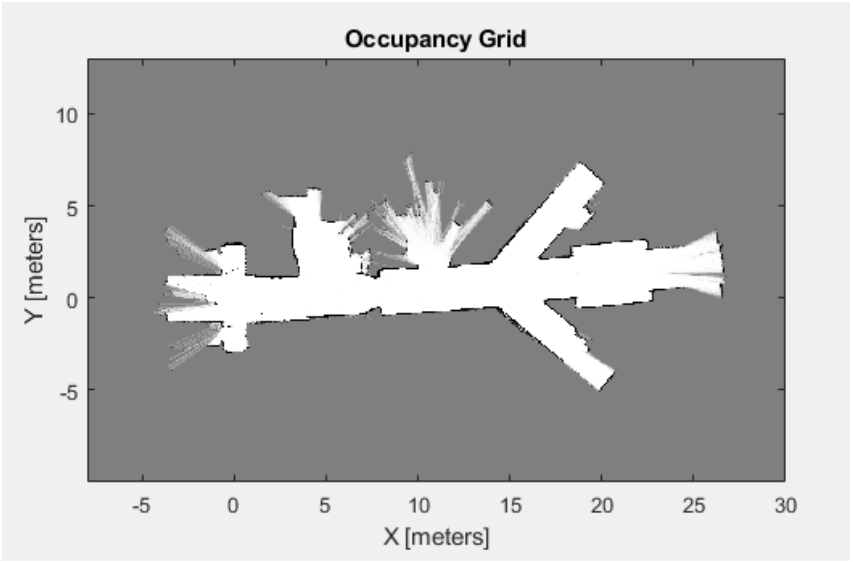
1. Conectar.m
2. Ini_amigobot.m
3. OnlineSLAM_RealRobot.m

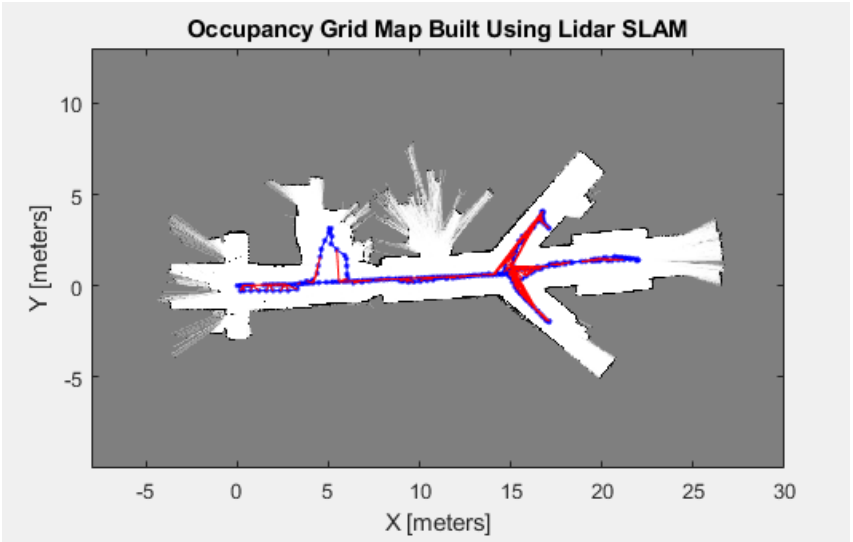
T1: roscore
T2: rosbag play rosbag_real_pasillo.bag

3.2. SLAM
Rosbag

Pasillo laboratorio



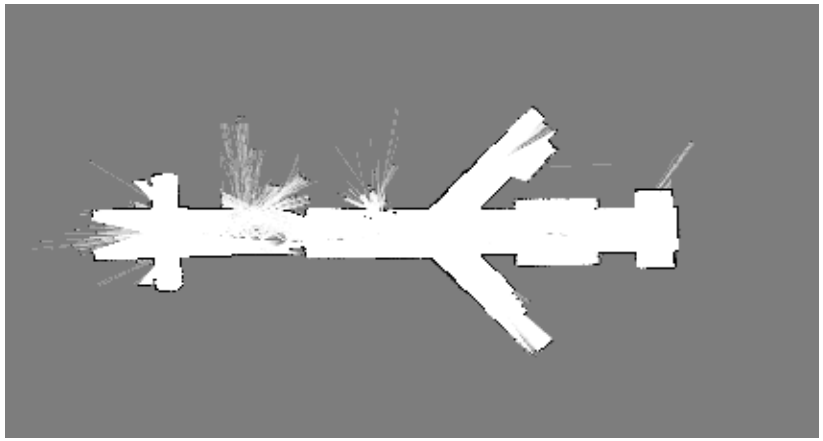




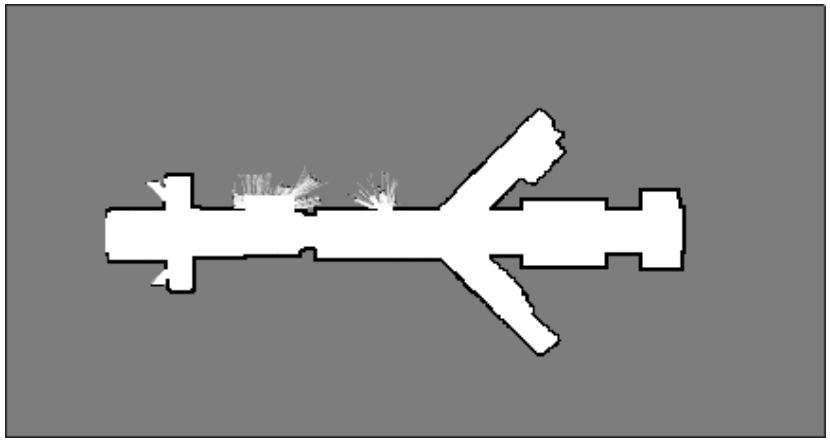
Modificar/limpiar un mapa (occupancy grid map object)

Fichero: export_and_import_modified_map.m

- Objetivos:
  - Limpiar el ruido del mapa
  - Resaltar el contorno de mapa



Mapa original



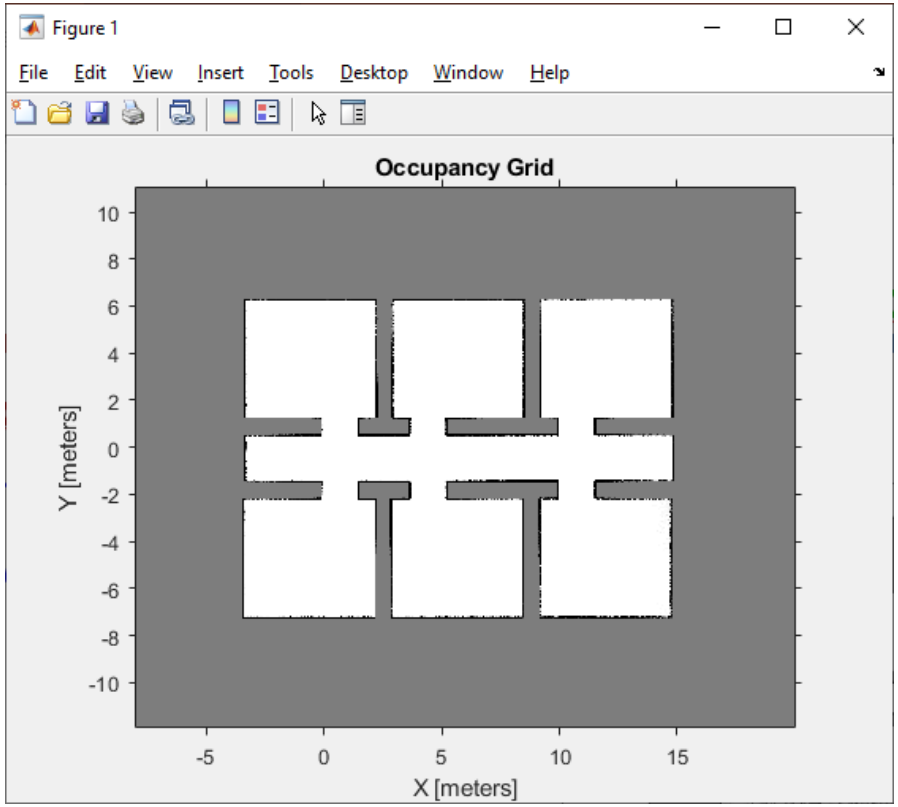
Mapa modificado

Modificar/limpiar un mapa (occupancy grid map object)

Step 1: Mostrar mapa con ruido en matlab y calcular dimensiones

Figure;

show (map)

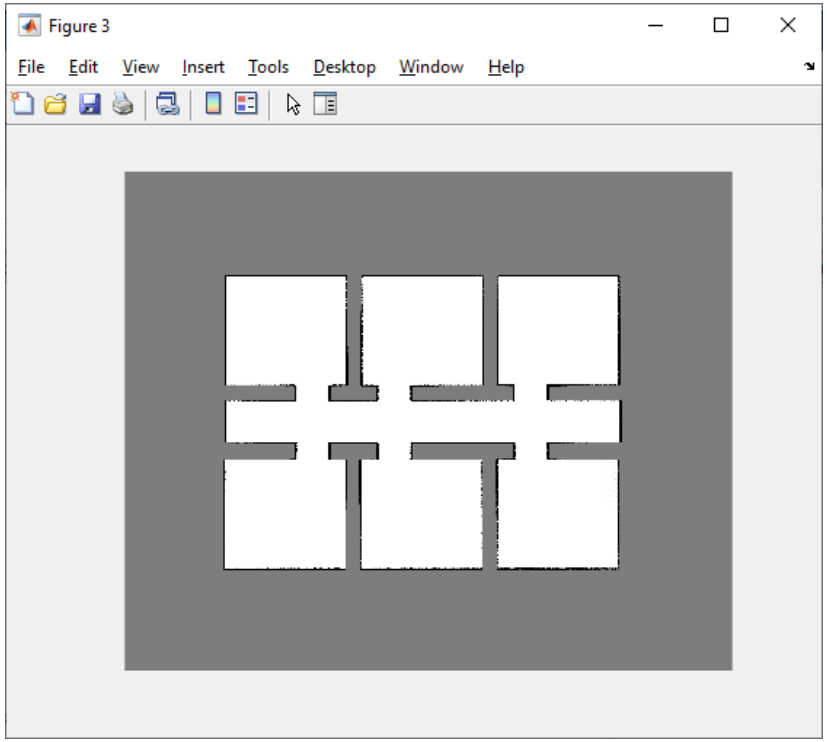


Creado con map.resolution=20

Modificar/limpiar un mapa (occupancy grid map object)

Step 2: Guardar mapa en formato imagen png sin bordes ni labels

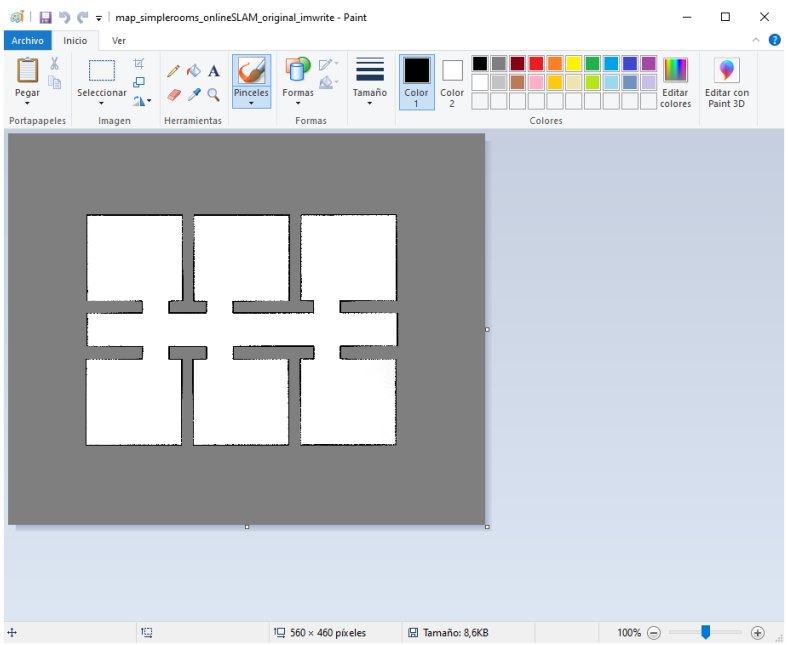
```
Figure;
show (map)


axis('off')
title('')
ylabel('')
xlabel('')
imwrite((1.-map.occupancyMatrix),'map_original.png')
```

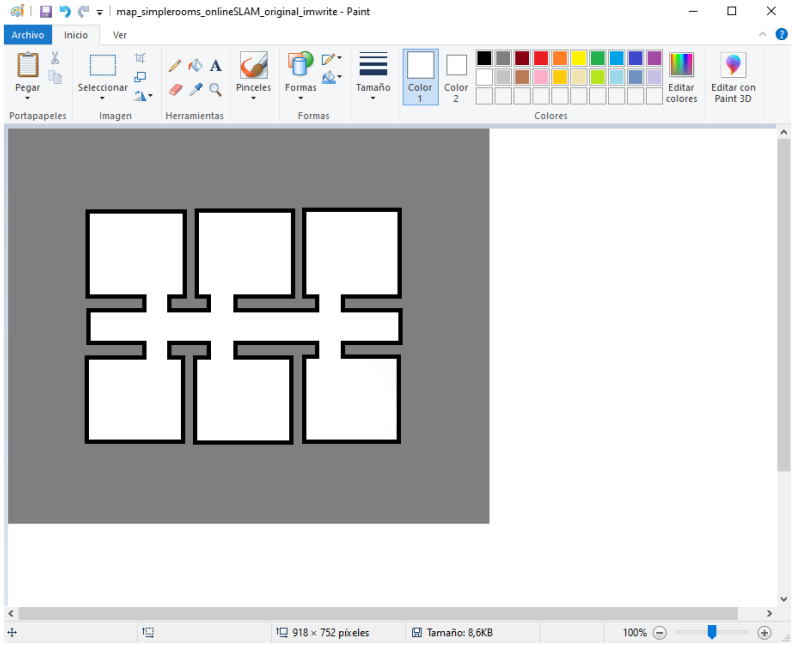


Modificar/limpiar un mapa (occupancy grid map object)

Step 3: Procesar imagen en Paint.

- Abrir map_original con Paint
- Limpiar imagen y resaltar contornos sin cambiar la resolución ni tamaño





- Guardar como **map_modificado.png**

Modificar/limpiar un mapa (occupancy grid map object)

Step 4: Leer imagen modificada

```
image = imread('map_modificado.png');

% Unknown areas (gray) should be removed and treated as free space. Create a logical matrix based on a threshold. Depending on your image, this value could be different. Occupied space should be set as 1 (white in image).

image=image(:,:,1);  %Esta línea hay que comentarla si la imagen sólo tiene una componente A x L x 1

imageNorm = double(image)/255;

imageOccupancy = 1 - imageNorm;
```

Modificar/limpiar un mapa (occupancy grid map object)

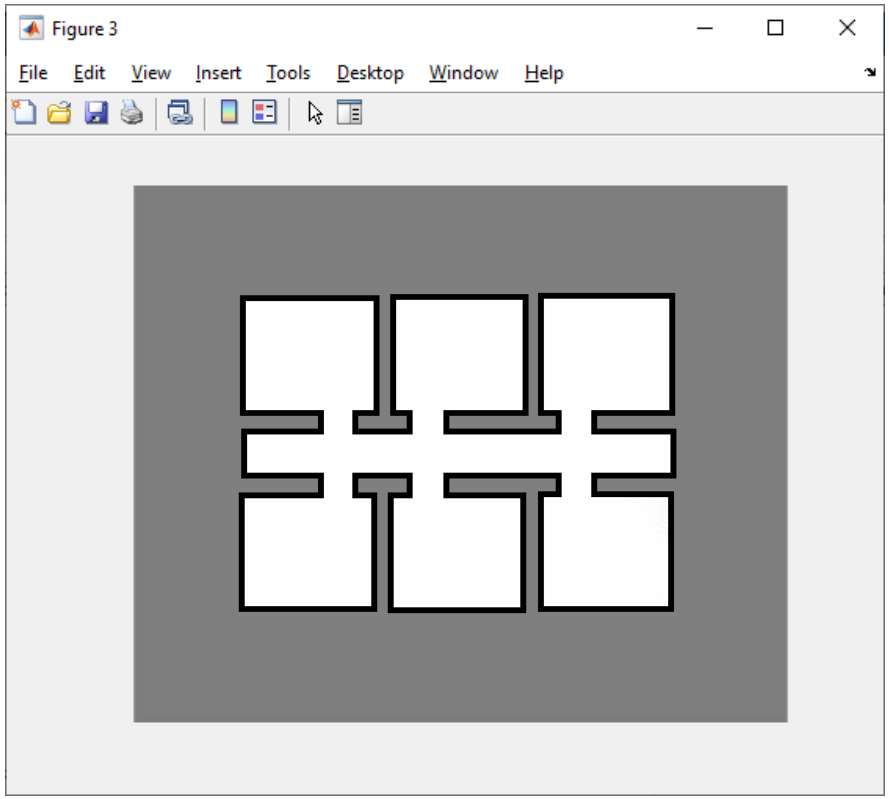
Step 5: Create occupancy grid map object using adjusted map image with the same word coordinates than original

```
% Create  a 2D occupancy grid map object using adjusted map image.
map_modified=occupancyMap(imageOccupancy,map.resolution);
%map.resolution= pixels/metros(medidos antes de convertir a imagen)

%locate the new map in the same word coordinates than the original
map_modified.GridLocationInWorld = map.GridLocationInWorld
```

Modificar/limpiar un mapa (occupancy grid map object)

Step 6: Visualizar y guardar el mapa modificado

```
figure()
show(map_modified)
```

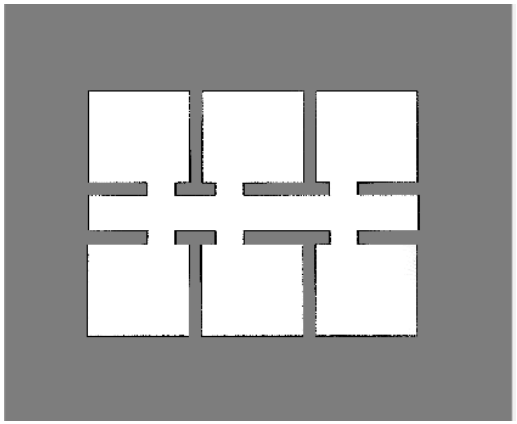


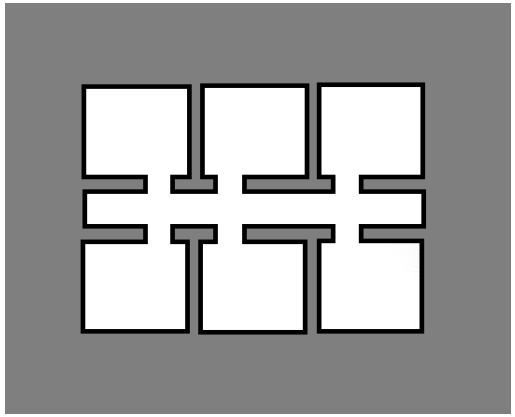
Save map_modified.mat map_modified

Modificar/limpiar un mapa

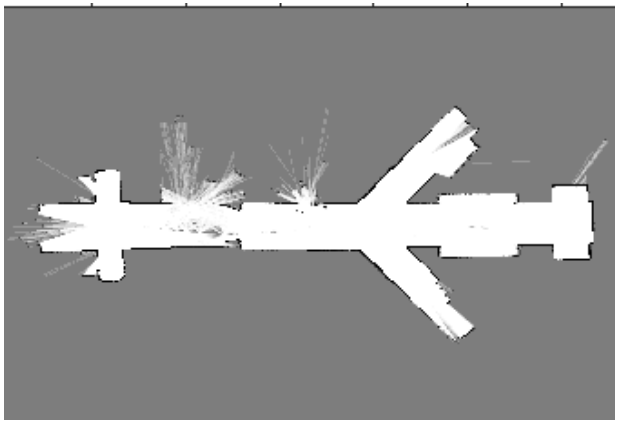
Resultados

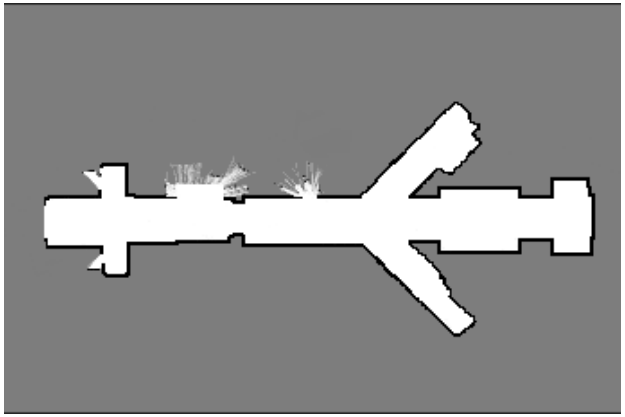
Simple rooms



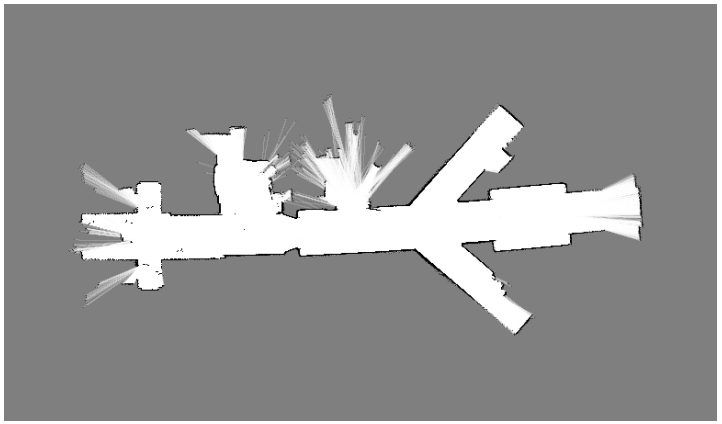


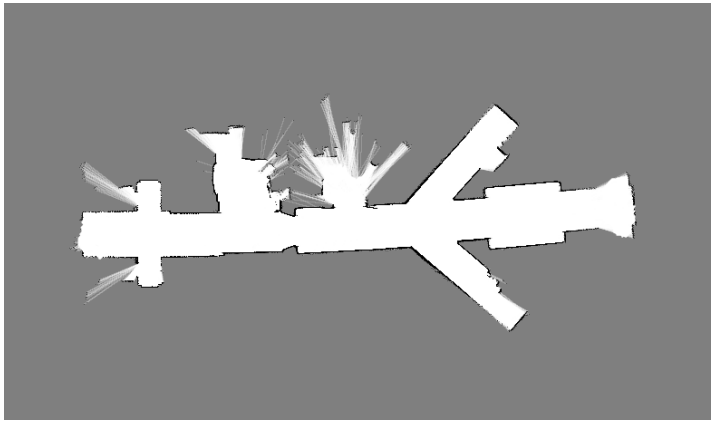
AmigoBot pasillo





Rosbag pasillo





4. Localización AMCL
Monte Carlo Localization

Punto de partida → AMCL_Localization_2022a.mlx

- Objetivo
  - Localizar un robot utilizando datos del láser y la odometría sobre un mapa conocido a priori.

  - Generar Script: AMCL_Localization_Simulator.m
    - Localización en mapa generado con el simulador STDR.
      - ✓ Distribución inicial partículas → global y local
    - Comparar resultados y obtener conclusiones

  - Generar script: AMCL_Localization_RealRobot.m
    - Localización en mapa generado con el robot real (pasillo laboratorio)
      - ✓ Distribución inicial partículas → global y local
    - Comparar resultados y obtener conclusiones

4. Localización AMCL
Monte Carlo Localization

AMCL_Localization_2022a.mlx

- Load the map
  - load mapa_con_OnlineSLAM_Simulator.mat
  - show(map);
- Setup amigobot motion model
  - odometryModel = odometryMotionModel;
  - odometryModel.Noise = [0.2 0.2 0.2 0.2];
- Setup the laser sensor model
  - rangeFinderModel = likelihoodFieldSensorModel;
  - rangeFinderModel.SensorLimits = [0 8];
  - rangeFinderModel.Map = map;

4. Localización AMCL
Monte Carlo Localization

AMCL_Localization_2022a.mlx

- Query the Transformation Tree
  - tftree = rostf;
  - waitForTransform(tftree,'/robot0','/robot0_laser_1');
  - sensorTransform = getTransform(tftree,'/robot0', '/robot0_laser_1');

- Setup amigobot motion model
  - laserQuat = [sensorTransform.Transform.Rotation.W
    sensorTransform.Transform.Rotation.X sensorTransform.Transform.Rotation.Y
    sensorTransform.Transform.Rotation.Z];
  - laserRotation = quat2eul(laserQuat, 'ZYX');

- Setup the SensorPose
  - rangeFinderModel.SensorPose = ...
    [sensorTransform.Transform.Translation.X sensorTransform.Transform.Translation.Y
    laserRotation(1)];;

4. Localización AMCL
Monte Carlo Localization

AMCL_Localization_2022a.mlx

- Initialize AMCL object
  - amcl = monteCarloLocalization;
  - amcl.UseLidarScan = true;
- Assign the MotionModel and SensorModel properties in the amcl object.
  - amcl.MotionModel = odometryModel;
  - amcl.SensorModel = rangeFinderModel;
- Setup AMCL (particle filter)
  - amcl.UpdateThresholds = [0.2,0.2,0.2];
  - amcl.ResamplingInterval = 1;

4. Localización AMCL
Monte Carlo Localization

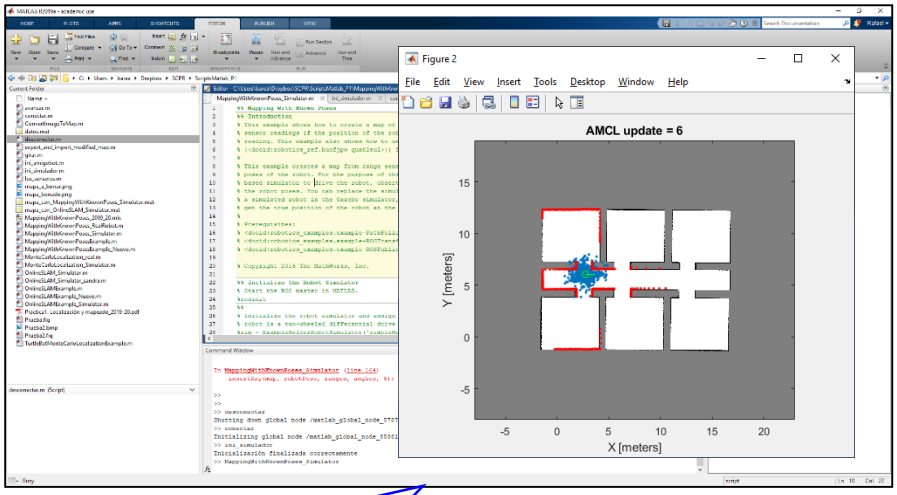
AMCL_Localization_2022a.mlx

- Configure AMCL object for localization with initial pose estimate
  - amcl.ParticleLimits = [500 50000];
  - amcl.GlobalLocalization = true; % false for local pose estimate
  - amcl.InitialPose = [0 0 0];
  - amcl.InitialCovariance = eye(3)*0.5;;

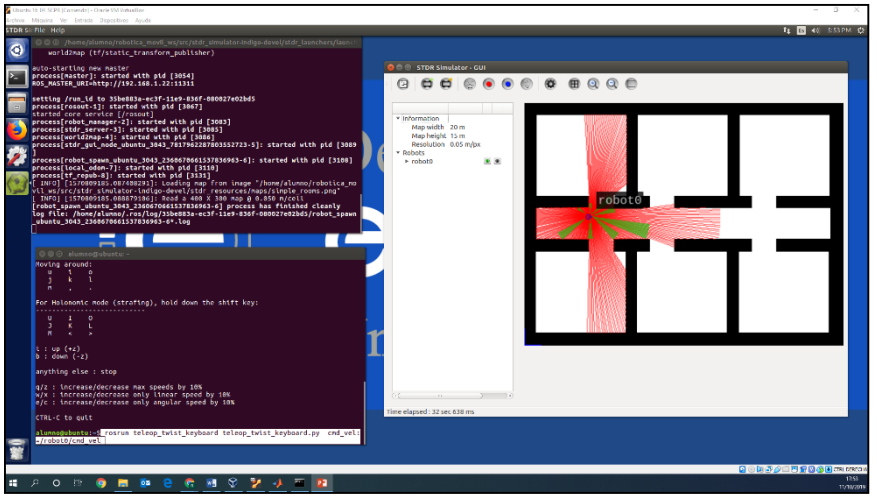
- Localization procedure → repetir blucle hasta localizarse correctamente
  - % Receive laser scan and odometry message.
    - scan = receive(sub_laser);
    - odompose = sub_odom.LatestMessage;
    - scans = lidarScan(scan);
  - % Compute robot's pose [x,y,yaw] from odometry message.
    - pose = [odompose.Pose.Pose.Position.X, odompose.Pose.Pose.Position.Y odomRotation(1)];
  - % Update estimated robot's pose and covariance using odometry and sensor readings.
    - [isUpdated,estimatedPose, estimatedCovariance] = amcl(pose, scans);

4. Localización AMCL
MCL- SIMULADOR

PC1 Scripts matlab



PC2 Máquina virtual (MASTER)

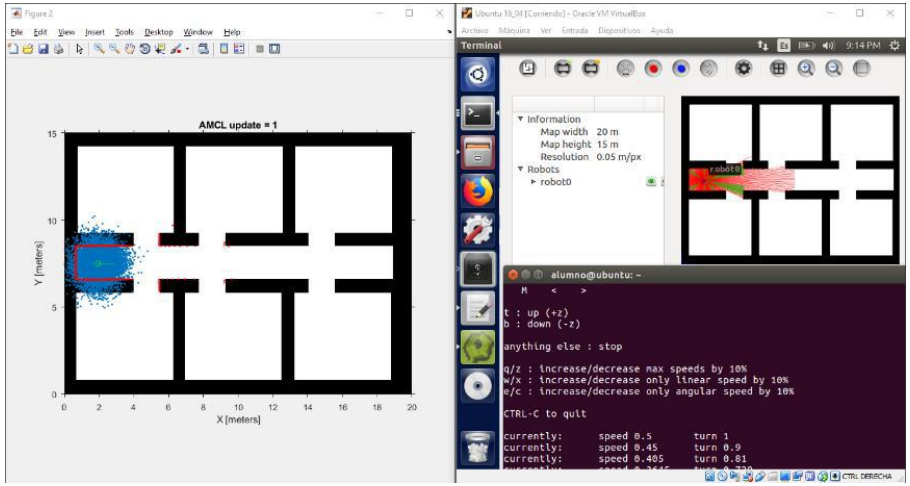


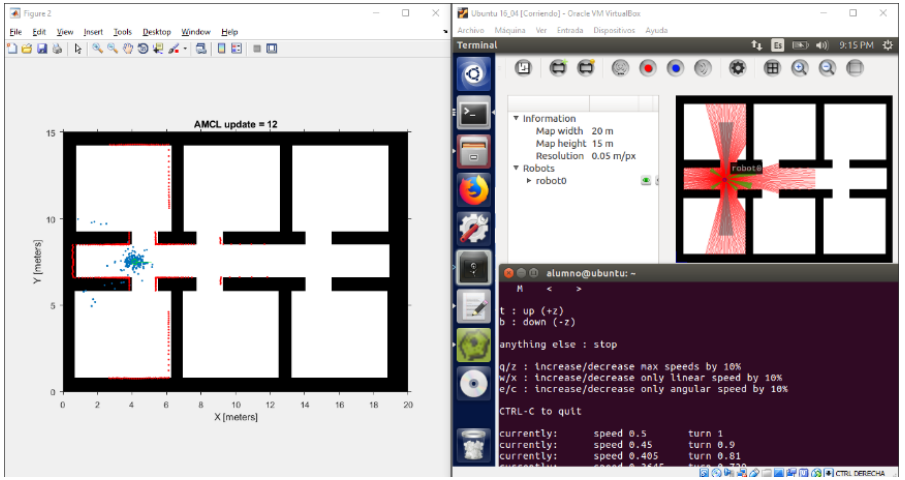
1. Conectar.m
2. Ini_simulador.m
3. AMCL_Localization_simulator.m

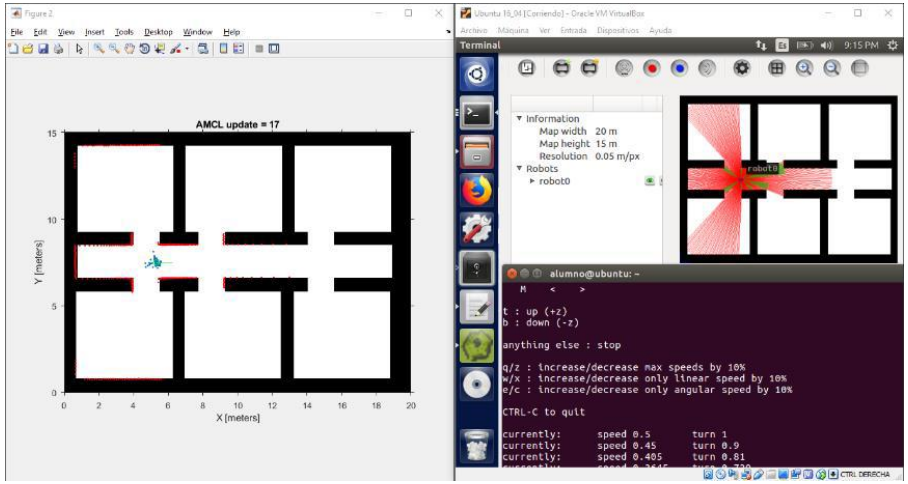
T1: roslaunch stdr_launchers amigobot.launch
T2: rosrun teleop_twist_keyboard teleop_twist_keyboard.py cmd_vel:=/robot0/cmd_vel

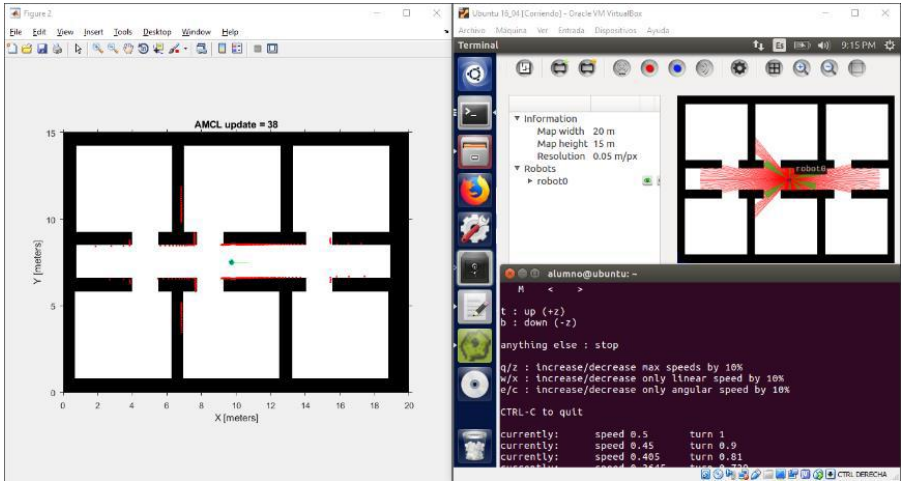
4. Localización AMCL
Monte Carlo Localization

Simple_rooms (prueba 1 – inicialización local en pasillo)



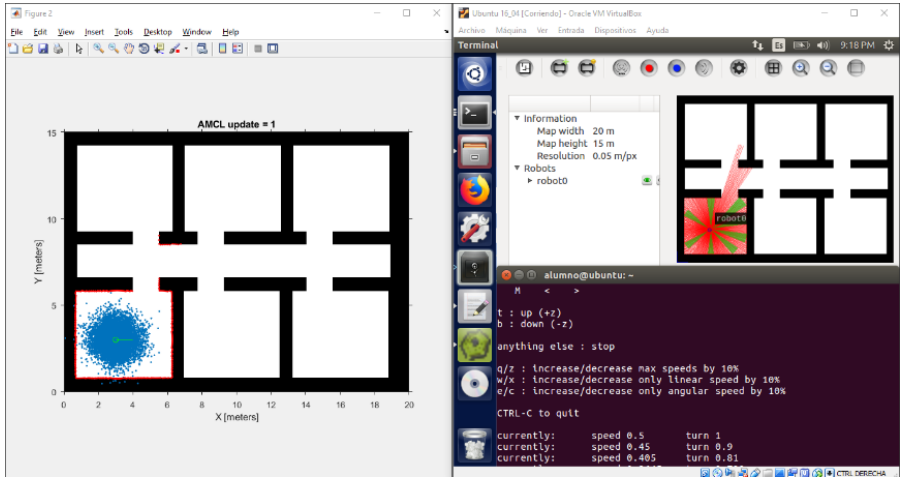


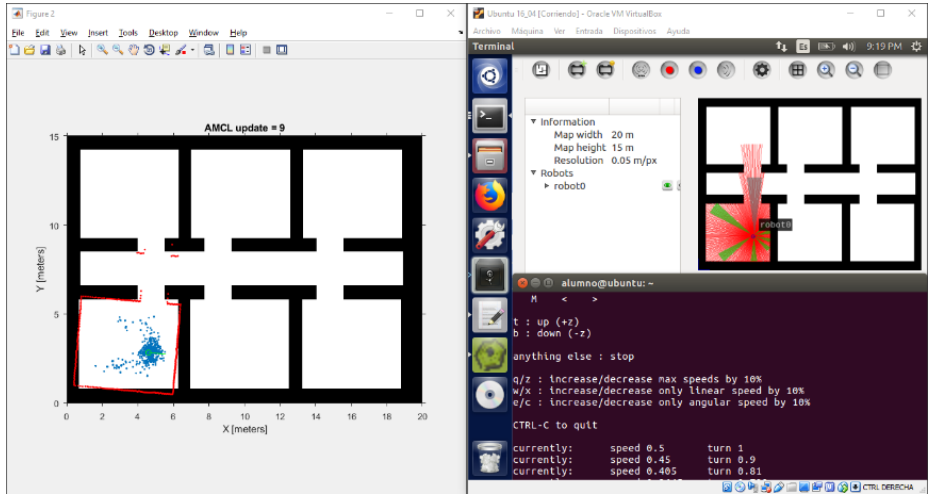


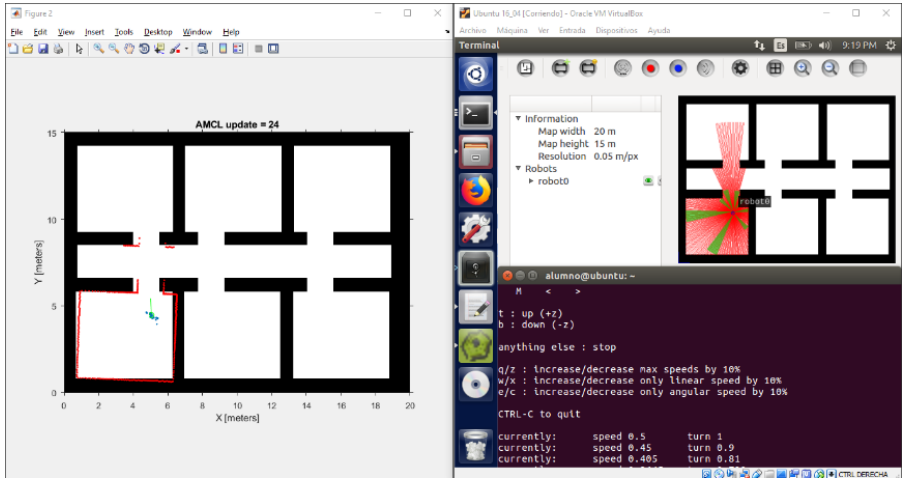


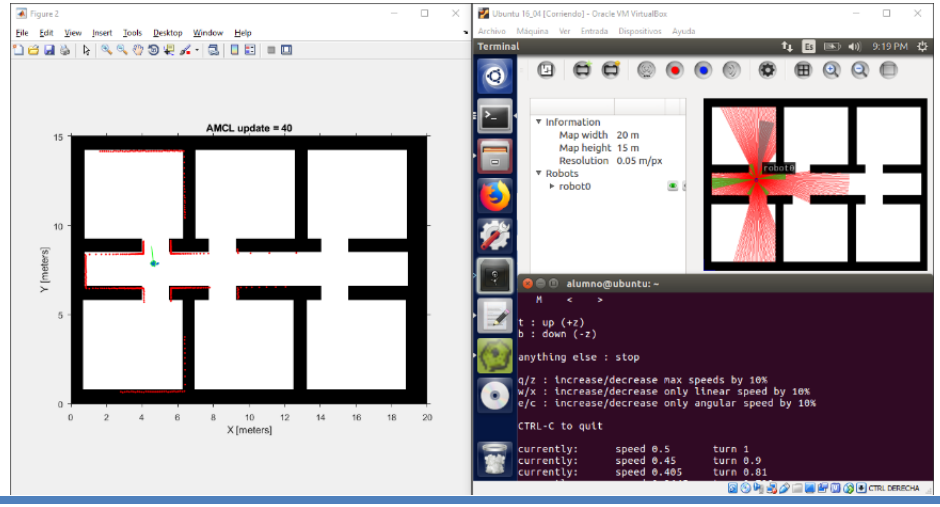
4. Localización AMCL
Monte Carlo Localization

Simple_rooms (prueba 2 – inicialización local en habitación)



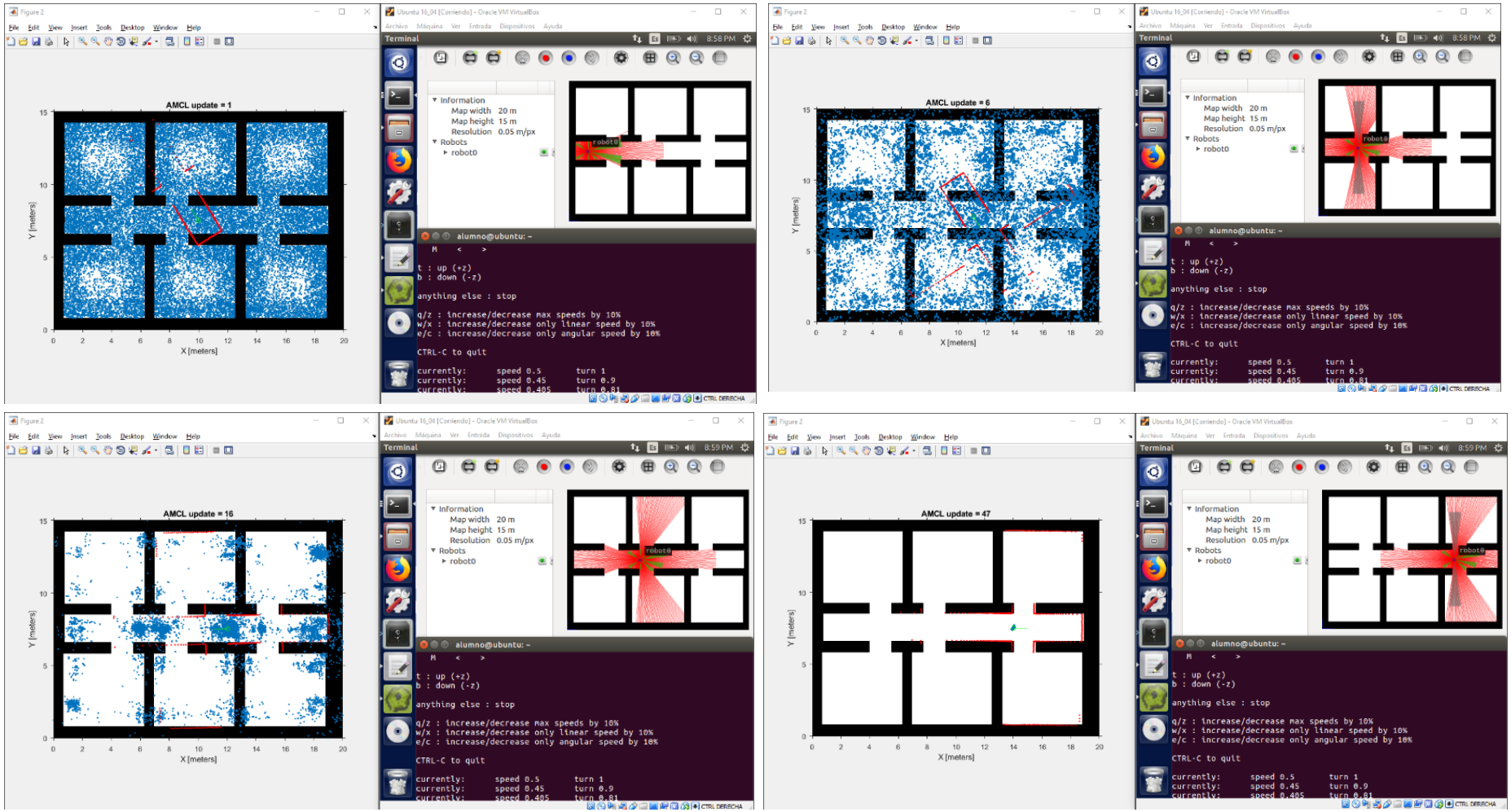






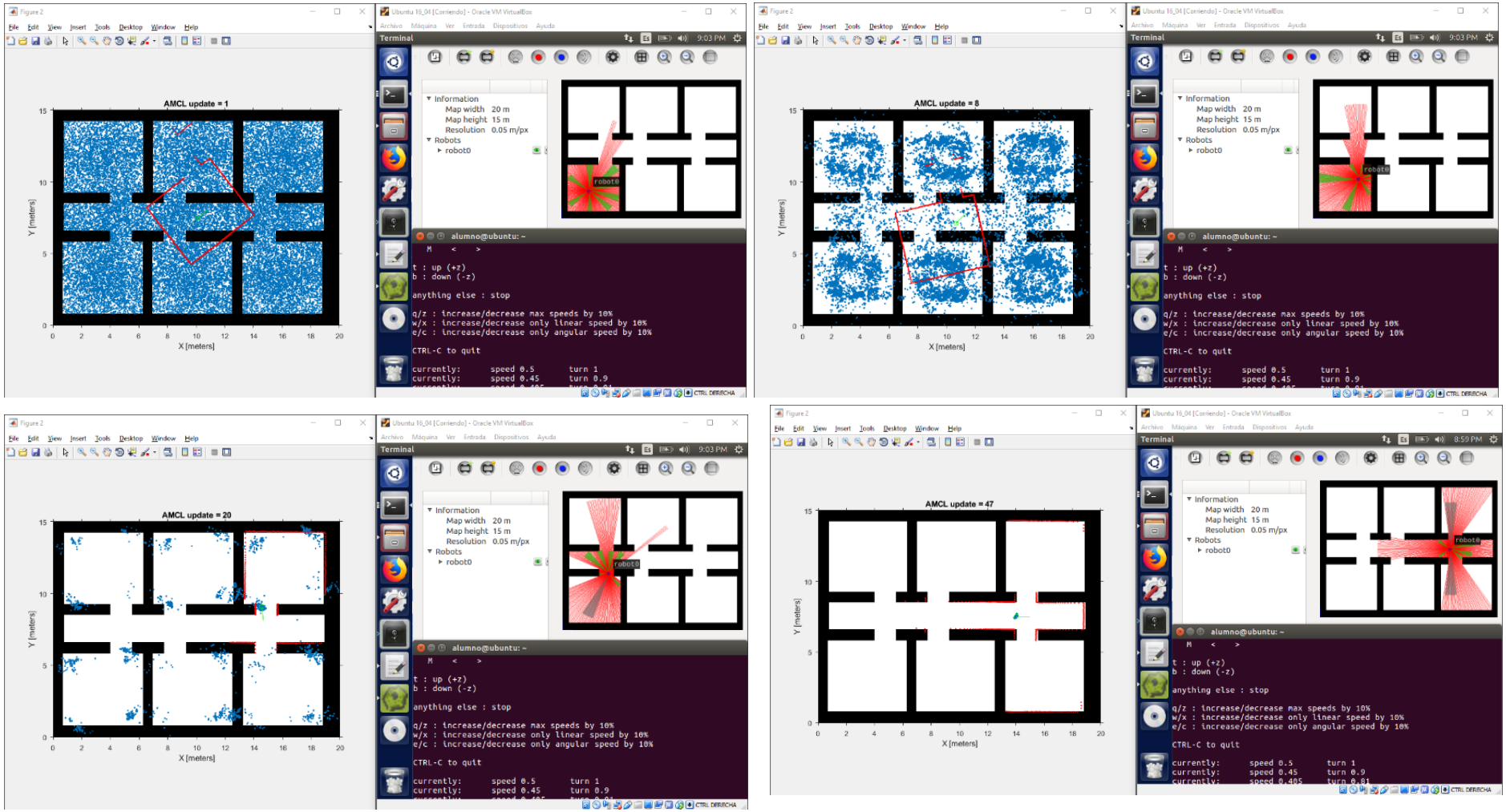
4. Localización AMCL
Monte Carlo Localization

Simple_rooms (prueba 3 – localización global)



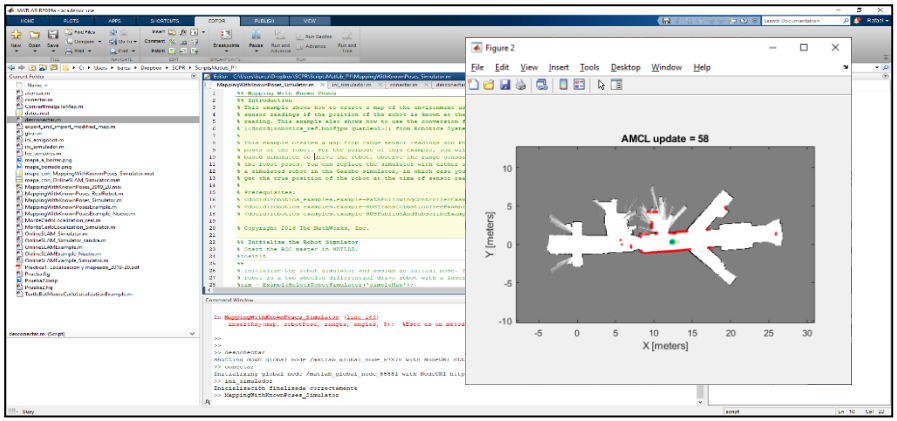
4. Localización AMCL
Monte Carlo Localization

Simple_rooms (prueba 4 – localización global)



4. Localización AMCL
MCL – Robot Real

PC1 Scripts matlab



PC2 Máquina virtual

1. Conectar.m
2. Ini_amigobot.m
3. AMCL_Localization_RobotReal.m

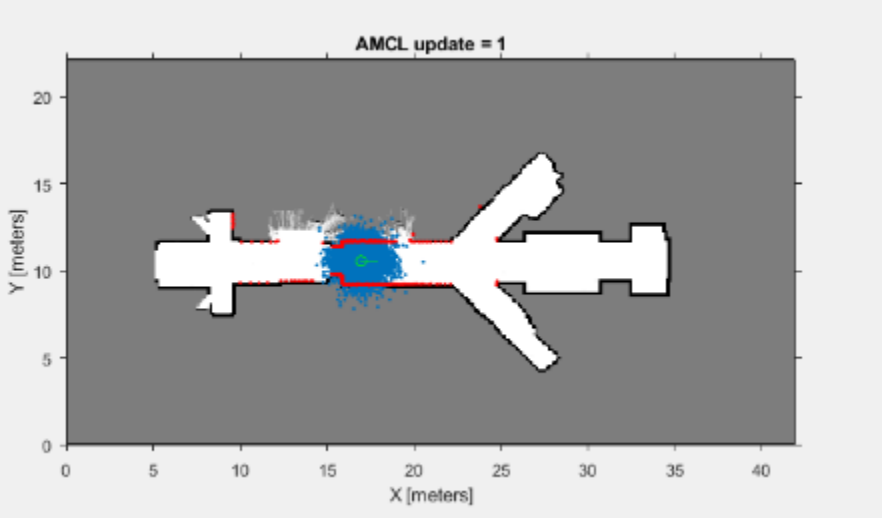
T1: rosrun teleop_twist_keyboard teleop_twist_keyboard.py
~~cmd_vel:=/robot0/cmd_vel~~??????????????????

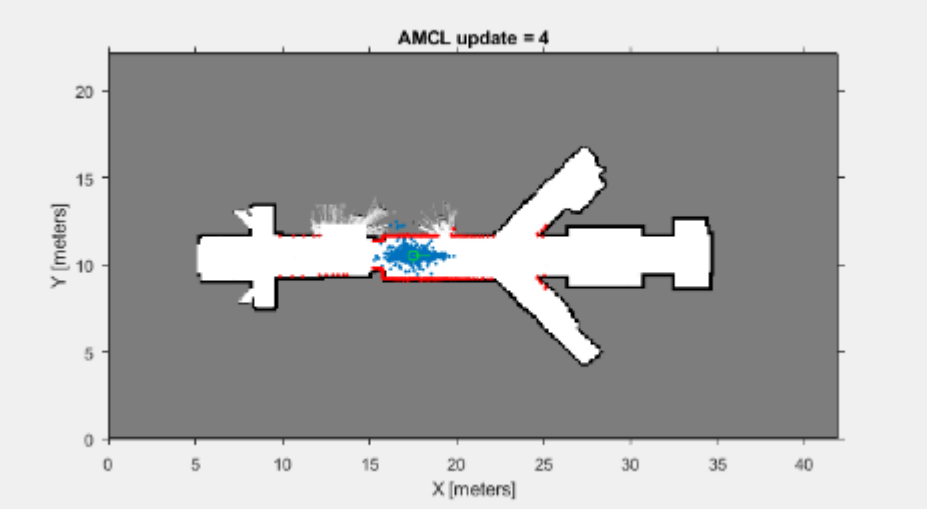
AmigoBot (MASTER)

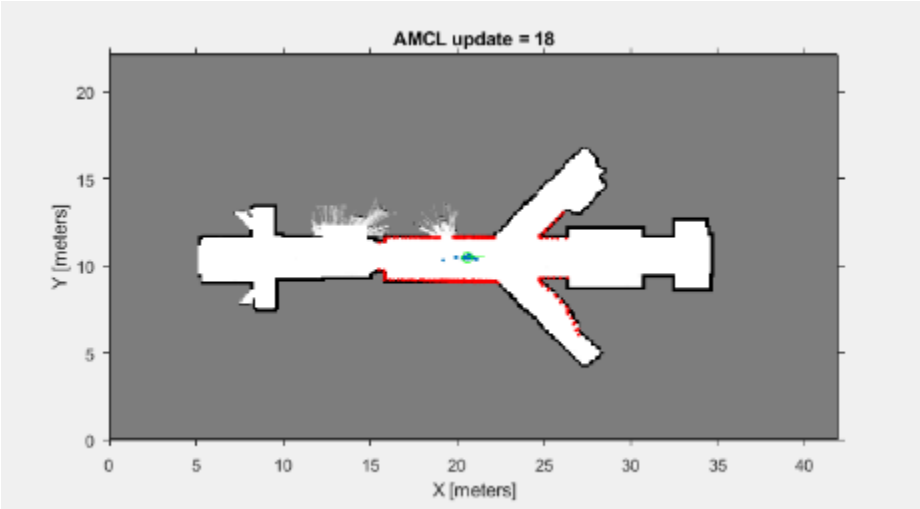
4. Localización AMCL

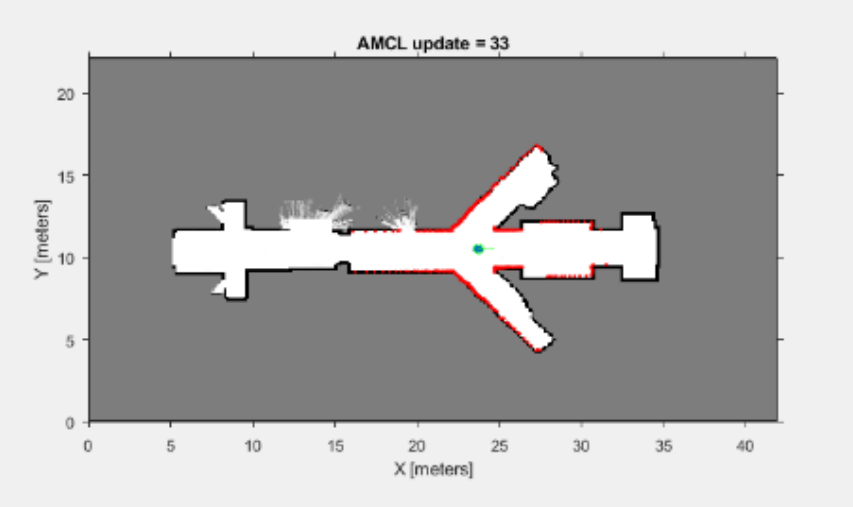
MCL – Robot Real

Robot real (pasillo laboratorio)





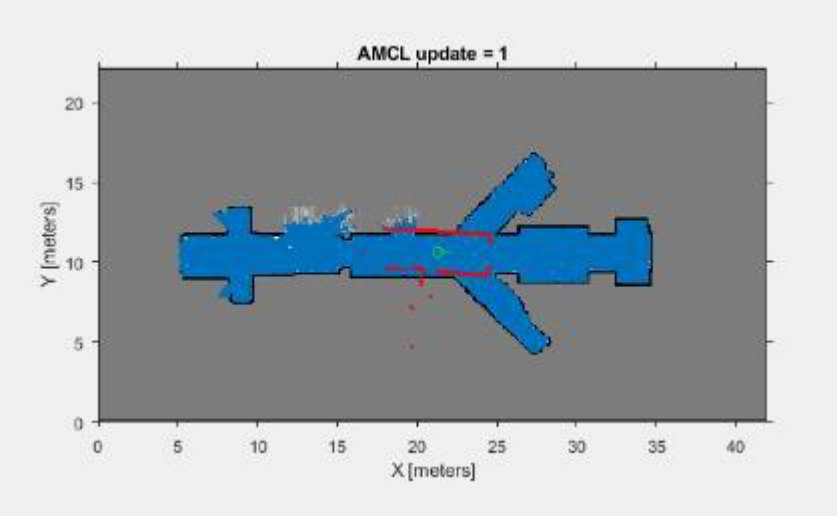


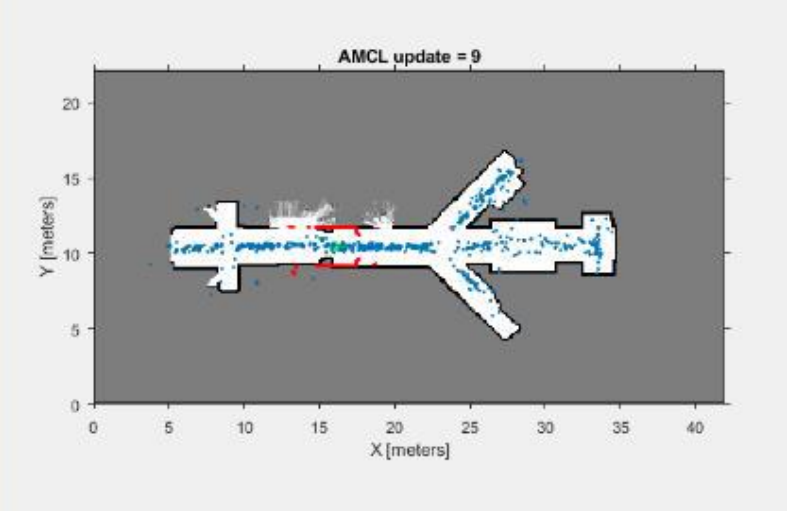


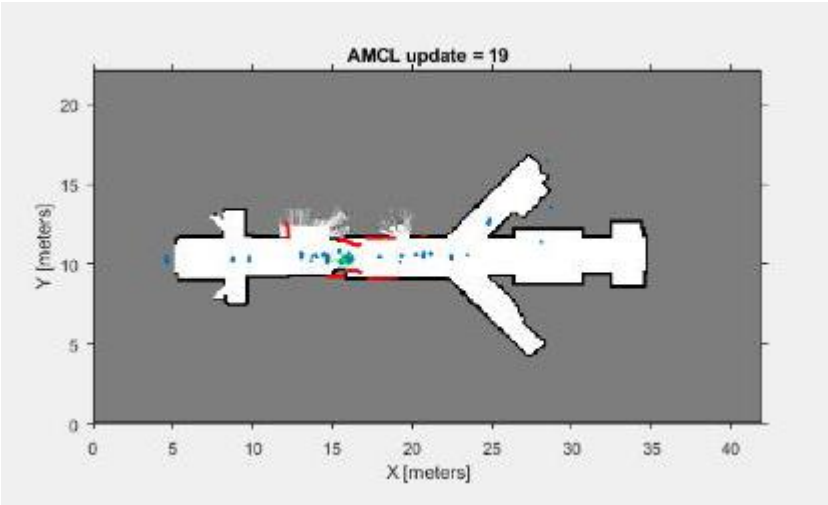
4. Localización AMCL

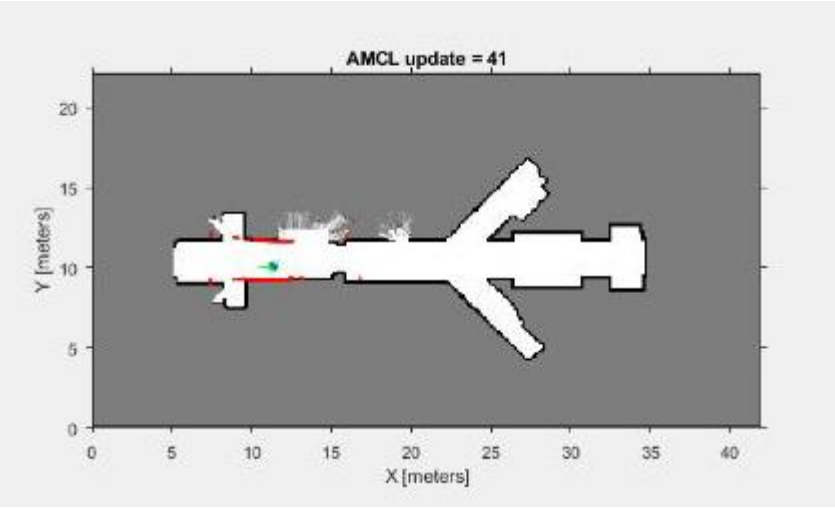
MCL – Robot Real

Robot real (pasillo laboratorio)





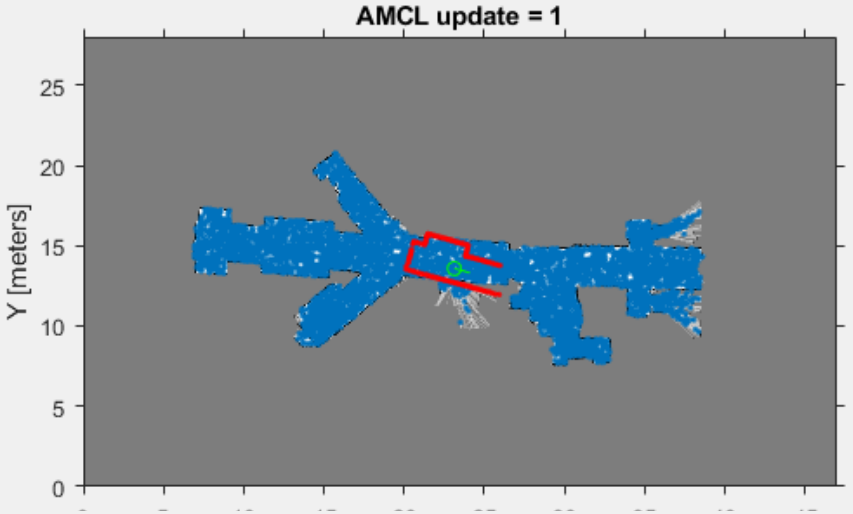


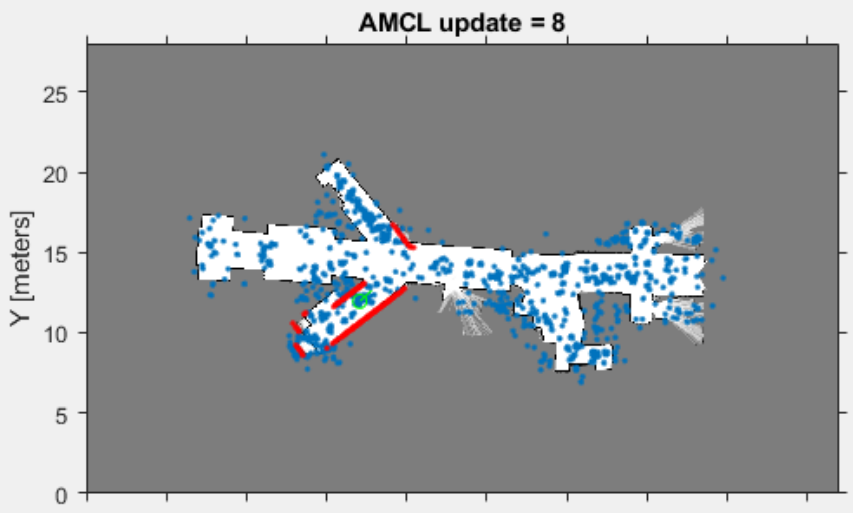


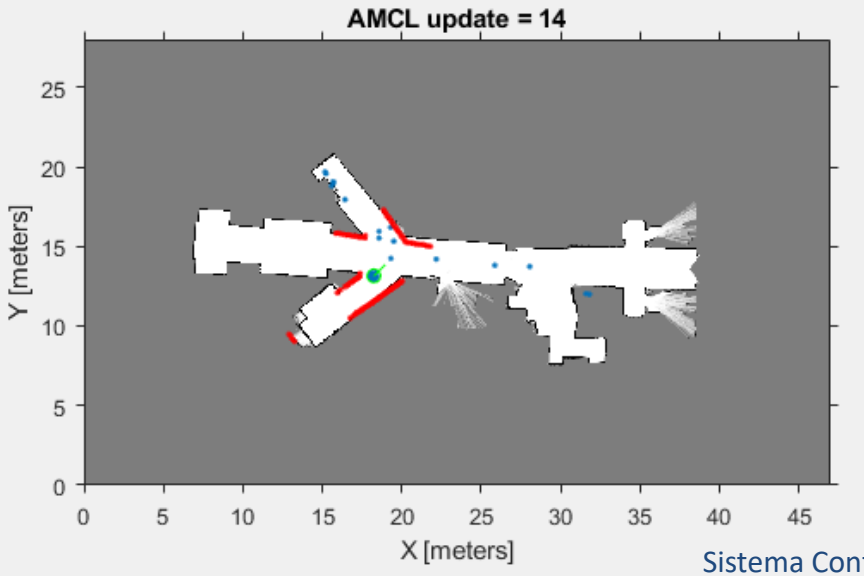
4. Localización AMCL

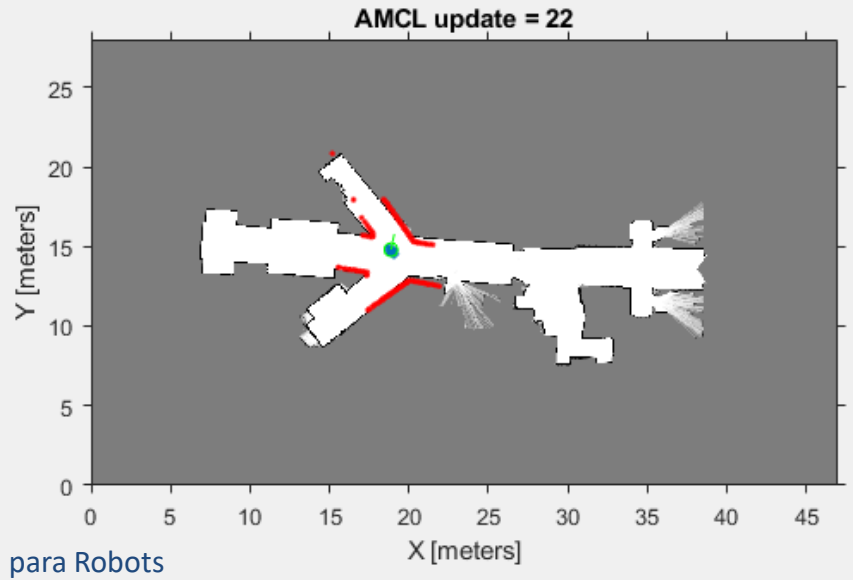
MCL – Robot Real

Robot real (pasillo laboratorio)



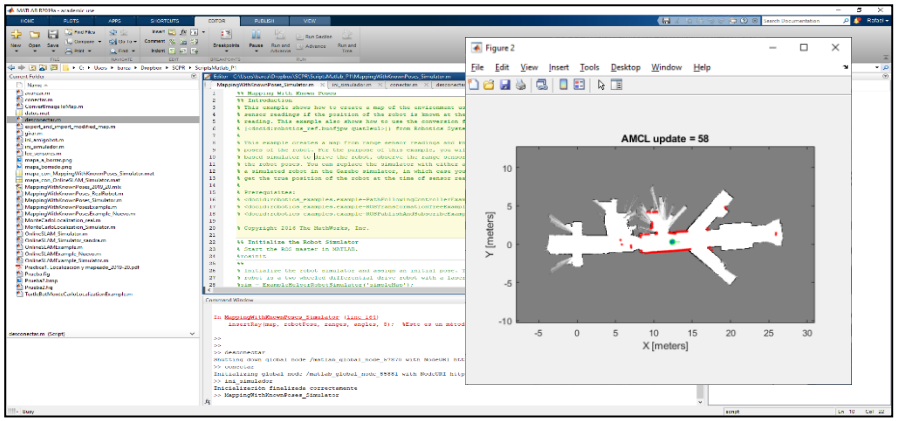




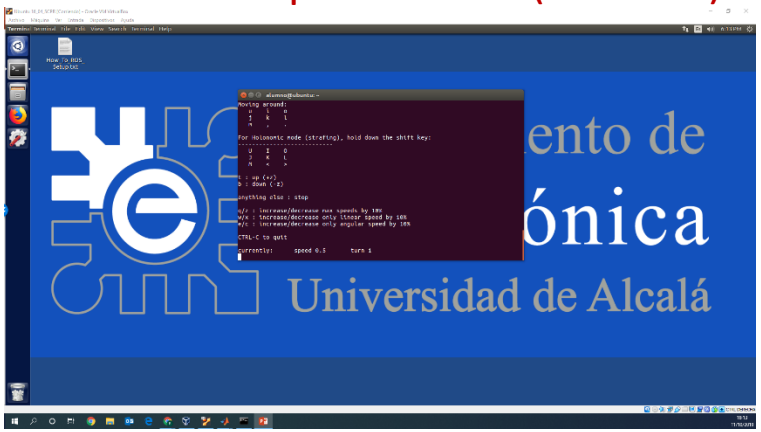


4. Localización AMCL
MCL - Rosbag

PC1 Scripts matlab



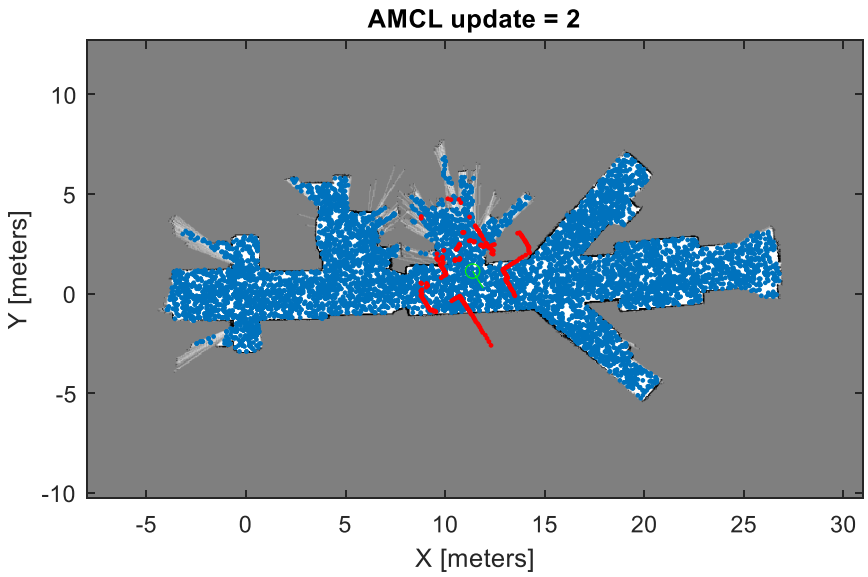
PC2 Máquina virtual (MASTER)

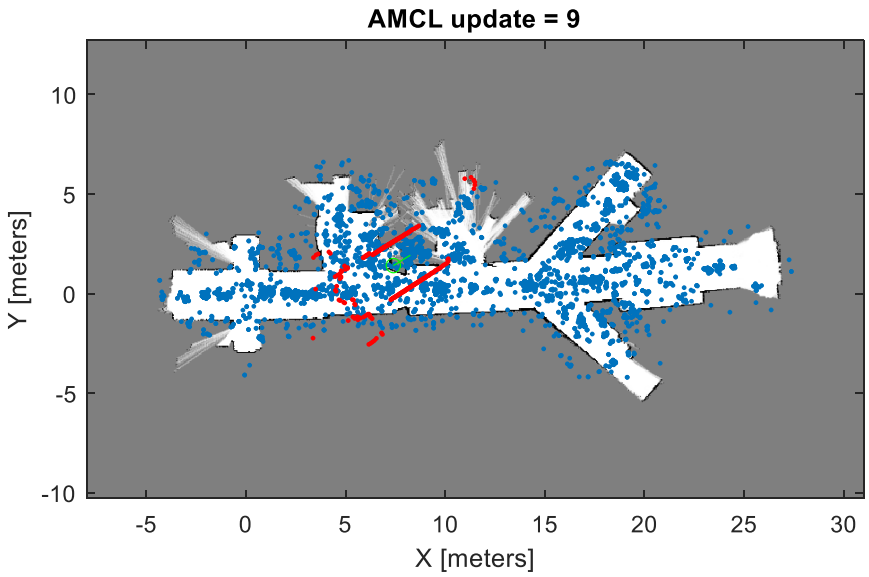


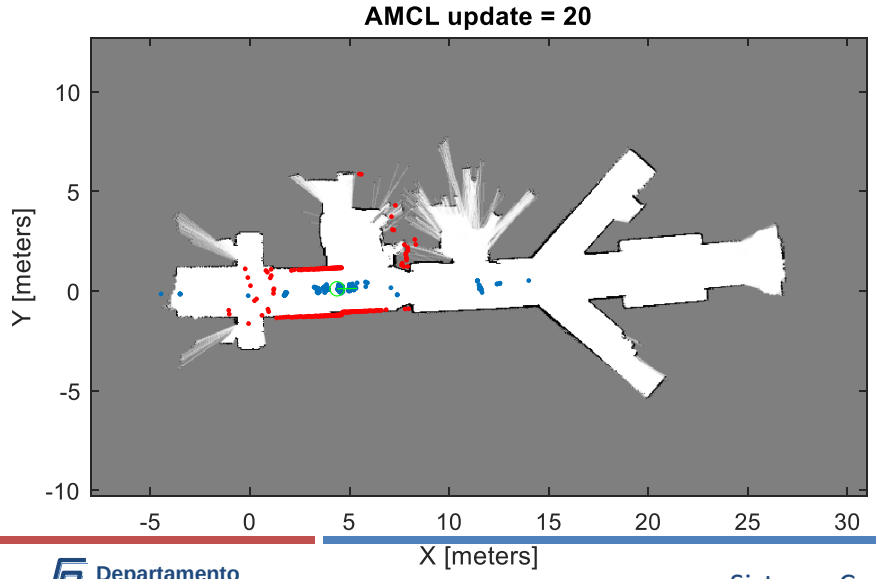
1. Conectar.m
2. Ini_amigobot.m
3. AMCL_Localization_RobotReal.m

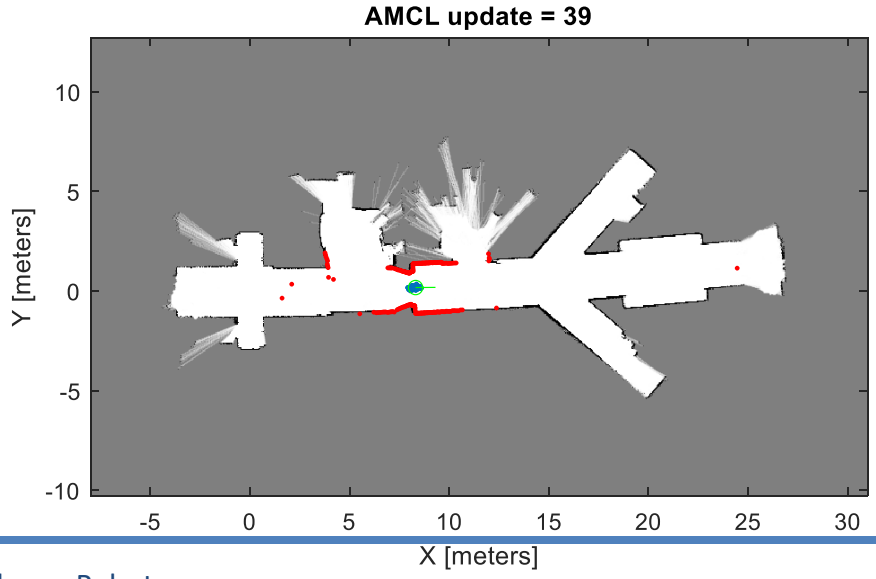
T1: roscore
T2: rosbag play --rate=0.5 rosbag_real_pasillo.bag

4. Localización AMCL
MCL - Rosbag



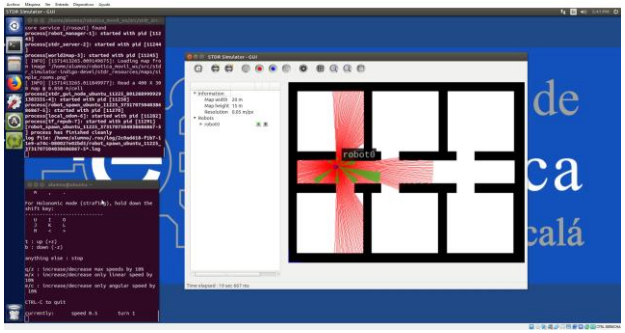


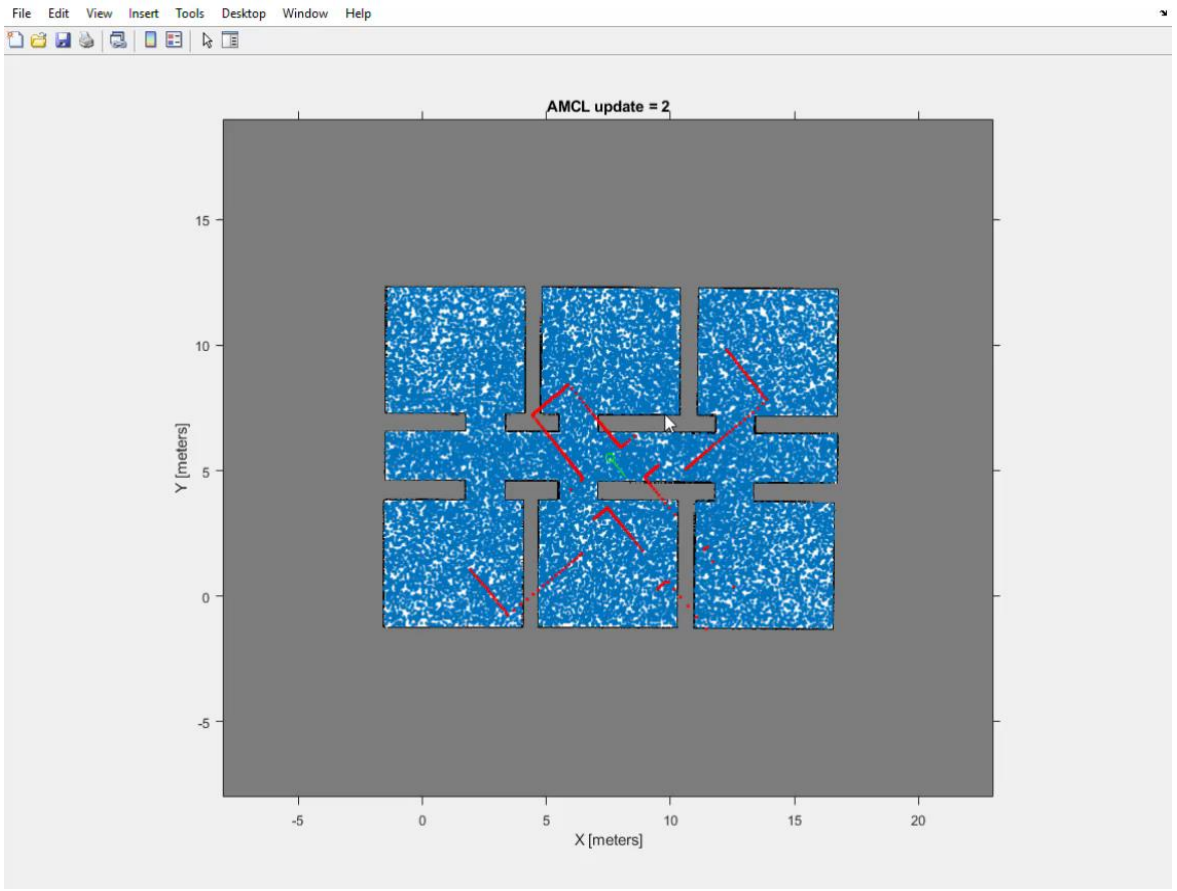




4. Localización AMCL
vídeos

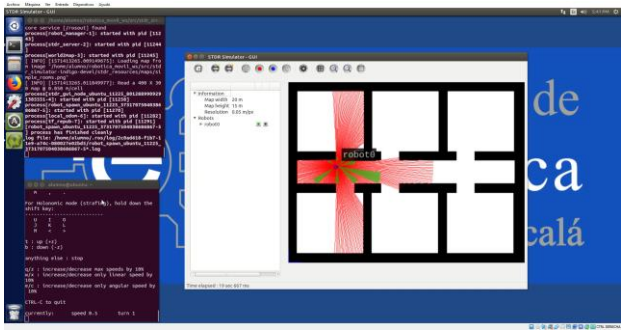
Simple_rooms (localización global)

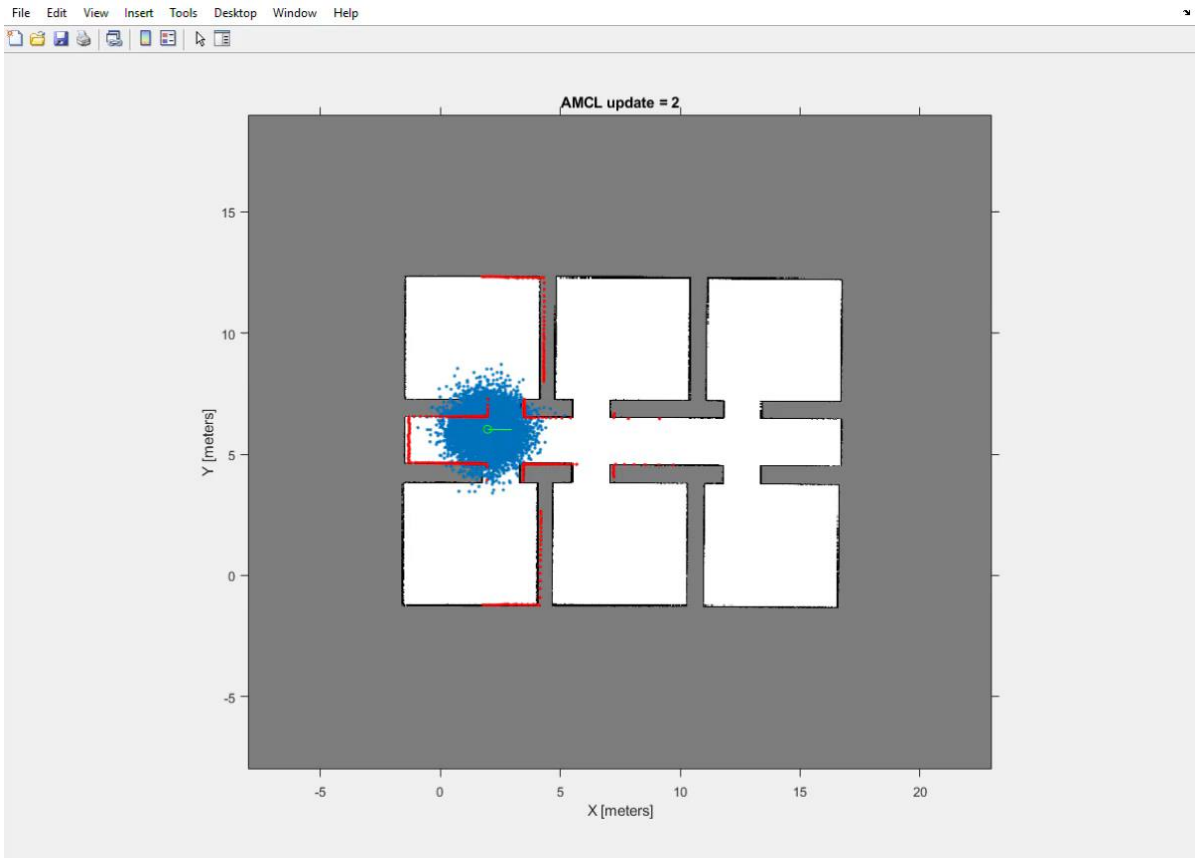




4. Localización AMCL
vídeos

Simple_rooms (localización local)





4. Localización AMCL
vídeos

Rosbag pasillo (localización global)