Técnicas de los Sistemas Inteligentes

Práctica1: Robótica.
Sesion4. Mapeo, Localización y Navegación

Curso 2016-17

Objetivo para las siguientes 2 sesiones

- Conocer las técnicas básicas de **mapeo**, **localización** y **navegación** en ROS para poder implementar un explorador de mapas basado en la técnica de exploración basada en fronteras.
- Demostración de exploración basada en fronteras

gmapping ==SLAM

- SLAM: Simultaneous Localization and Mapping
  - Calcular las poses del robot y el mapa del entorno a la vez
  - Técnica usada por robots para construir un mapa en un entorno **desconocido** mientras que, a la vez, trata de estimar la pose actual.
- El problema del huevo o la gallina:
  - es necesario un mapa conocido para estimar la posición
  - es necesaria una posición fiable para poder construir un mapa.
- Soluciona dos problemas fundamentales de la robótica
- **Localization (Localización):** estimar la pose del robot.
- **Mapping (Mapeo):** construir el mapa

gmapping

- http://wiki.ros.org/gmapping
- El paquete gmapping porporciona SLAM basado en laser como un nodo ROS llamado **slam_gmapping**
- Usa el algoritmo FastSLAM
- Toma los scans laser y la odometry y construye un mapa 2D representado como una occupancy grid
- El mapa lo publica en el topic **/map**
- Actualiza el estado del mapa conforme el robot se mueve
- ROS with gmapping video

Run gmapping

- Primero lanzar Gazebo (o Stage) con Turtlebot

```
$ roslaunch turtlebot_gazebo turtlebot_world.launch
```

```
$ roslaunch turtlebot_stage turtlebot_in_stage.launch
```

- Ahora arrancar gmapping en una nueva terminal

```
$ rosrun gmapping slam_gmapping
```

```
viki@c3po:~$ rosrun gmapping slam_gmapping
[ INFO] [1482222664.730535530, 36.260000000]: Laser is mounted upwards.
 -maxUrange 9.99 -maxUrange 9.99 -sigma     0.05 -kernelSize 1 -lstep 0.05 -lobs
Gain 3 -astep 0.05
 -srr 0.1 -srt 0.2 -str 0.1 -stt 0.2
 -linearUpdate 1 -angularUpdate 0.5 -resampleThreshold 0.5
 -xmin -100 -xmax 100 -ymin -100 -ymax 100 -delta 0.05 -particles 30
[ INFO] [1482222664.748575358, 36.280000000]: Initialization complete
update frame 0
update ld=0 ad=0
Laser Pose= -0.0858969 0.0494185 -0.022192
m_count 0
Registering First Scan
```

(C)2016 Roi Yehoshua

Run gmapping and teleoperation

- Now move the robot using teleop

```
$ roslaunch turtlebot_teleop keyboard_teleop.launch
```

- El mapa se publica en el topic /map

```
$ rostopic echo /map -n1
```

- Message type is nav_msgs/OccupancyGrid
- Occupancy is represented as an integer with:
  - 0 meaning completely free
  - 100 meaning completely occupied
  - the special value -1 for completely unknown

(C)2016 Roi Yehoshua

Nav_msgs/OccupancyGrid

nav_msgs/OccupancyGrid Message

File: nav_msgs/OccupancyGrid.msg

Raw Message Definition

```
# This represents a 2-D grid map, in which each cell represents the probability of
# occupancy.

Header header

#MetaData for the map
MapMetaData info

# The map data, in row-major order, starting with (0,0).  Occupancy
# probabilities are in the range [0,100].  Unknown is -1.
int8[] data
```

Compact Message Definition

```
std_msgs/Header header
nav_msgs/MapMetaData info
int8[] data
```

Nav_msgs/MapMetaData

nav_msgs/MapMetaData Message

File: `nav_msgs/MapMetaData.msg`

Raw Message Definition

```
# This hold basic information about the characterists of the OccupancyGrid

# The time at which the map was loaded
time map_load_time
# The map resolution [m/cell]
float32 resolution
# Map width [cells]
uint32 width
# Map height [cells]
uint32 height
# The origin of the map [m, m, rad].  This is the real-world pose of the
# cell (0,0) in the map.
geometry_msgs/Pose origin
```

Compact Message Definition

```
time map_load_time
float32 resolution
uint32 width
uint32 height
geometry_msgs/Pose origin
```

Guardar mapas: map_server

- map_server allows you to load and save maps
- To install the package:

```
$ sudo apt-get install ros-indigo-map-server
```

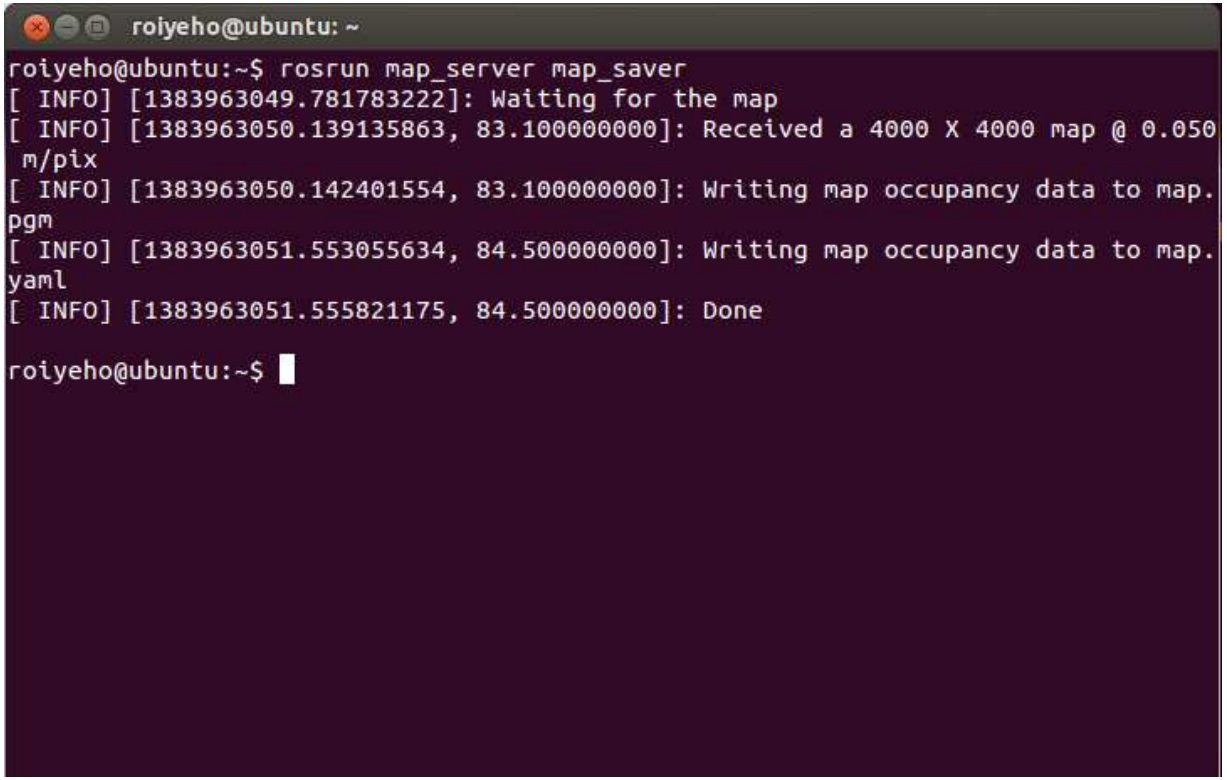
- To save dynamically generated maps to a file:

```
$ rosrun map_server map_saver [-f mapname]
```

- La opción **-f** sirve para poner una base de nombre diferente para los ficheros de salida.
- map_saver generates the following files in the current directory:
  - **map.pgm** – the map itself
  - **map.yaml** – the map's metadata

(C)2016 Roi Yehoshua

Saving the map using map_server



(C)2016 Roi Yehoshua

map_server

- You can open the pgm file with the default Ubuntu image viewer program (eog)

```
$ eog map.pgm
```

(C)2016 Roi Yehoshua

Formato de la imagen

- La imagen describe el estado de ocupación de cada celda en el mundo en el color del pixel correspondiente.
- Pixels más claros representan espacio libre, más oscuros espacio ocupado, entre ambos colores representan desconocido.
- Los umbrales para dividir las categorías están definidos en un fichero YAML.

(C)2013 Roi Yehoshua


```
image: map.pgm
resolution: 0.050000
origin: [-100.000000, -100.000000, 0.000000]
negate: 0
occupied_thresh: 0.65
free_thresh: 0.196
```

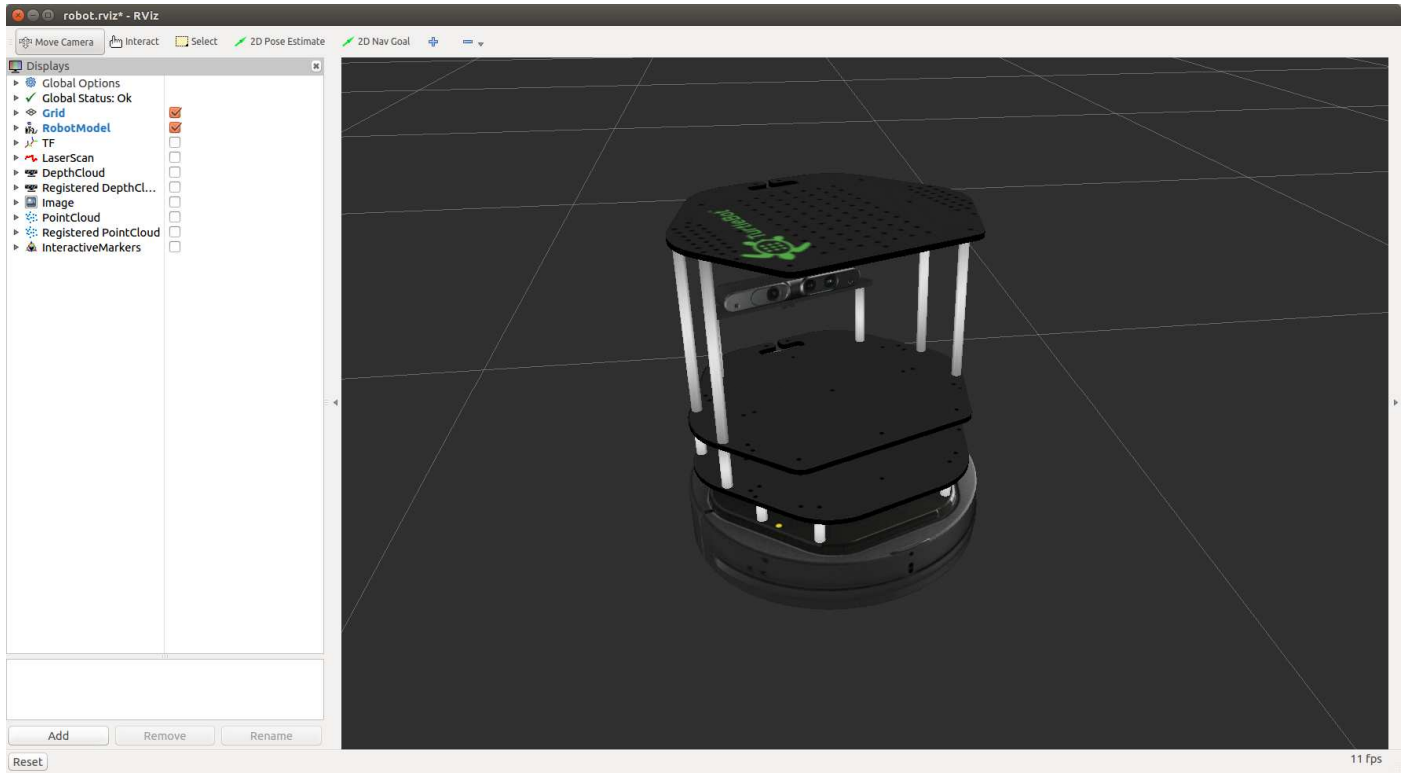
- Important fields:
  - **resolution**: Resolution of the map, meters / pixel
  - **origin**: The 2-D pose of the lower-left pixel in the map as (x, y, yaw)
  - **occupied_thresh**: Pixels with occupancy probability greater than this threshold are considered completely occupied.
  - **free_thresh**: Pixels with occupancy probability less than this threshold are considered completely free.

(C)2013 Roi Yehoshua

rviz with TurtleBot

- You can start rviz already configured to visualize the robot and its sensor's output:

```
$ roslaunch turtlebot_rviz_launchers view_robot.launch
```

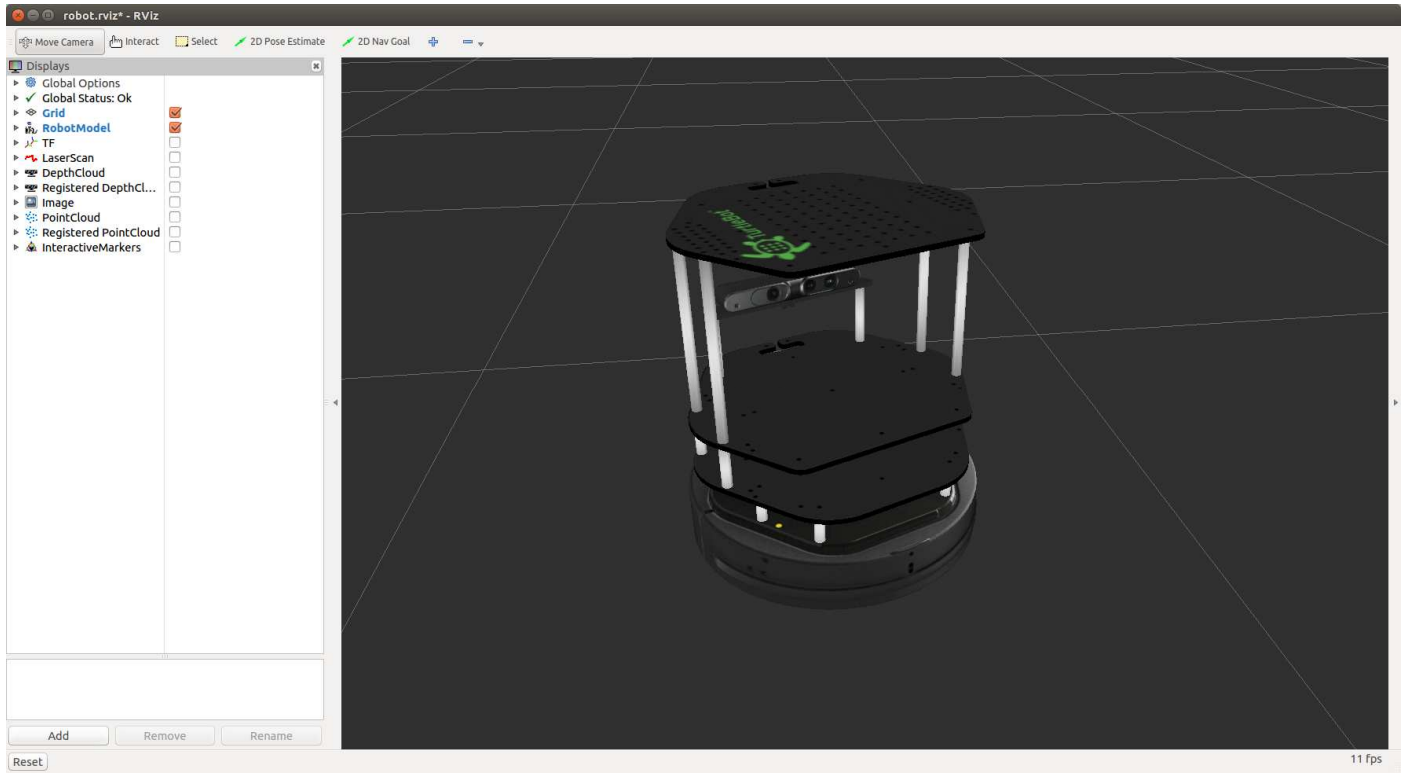


(C)2016 Roi Yehoshua

rviz with TurtleBot

- You can start rviz already configured to visualize the robot and its sensor's output:

```
$ roslaunch turtlebot_rviz_launchers view_robot.launch
```

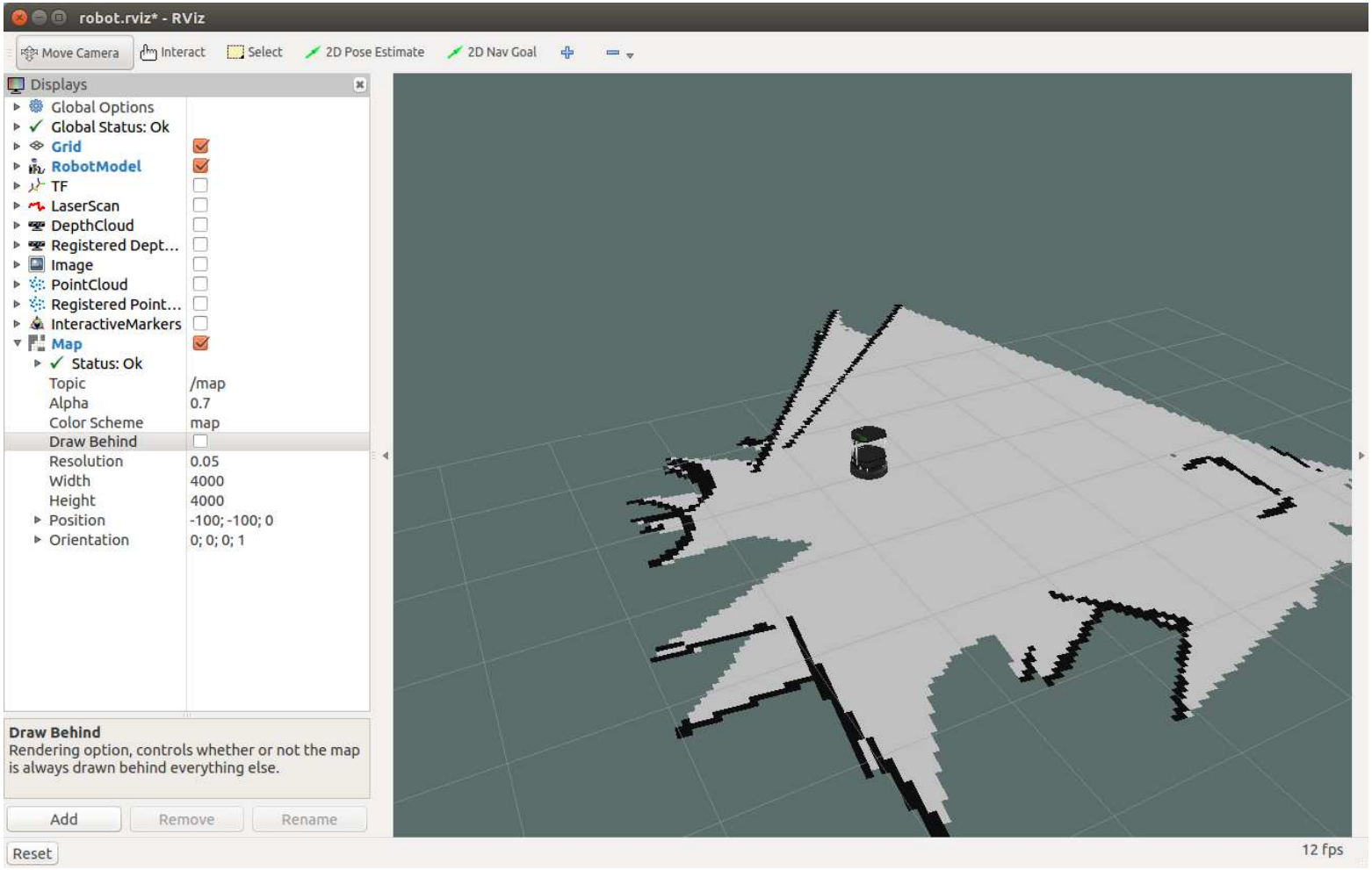


(C)2016 Roi Yehoshua

- Add the Map display
- Set the **topic** to /map
- Now you will be able to watch the mapping progress in rviz

(C)2016 Roi Yehoshua

Map Display



(C)2016 Roi Yehoshua

Loading and Saving Configuration

- You can save your rviz settings by choosing File > Save Config from the menu
- Your settings will be saved to a .rviz file
- Then, you can start rviz with your saved configuration:

```
$ rosrun rviz rviz -d my_config.rviz
```

(C)2016 Roi Yehoshua

Probar con otros mundos de Gazebo

- Crear un paquete llamado mi_mapeo
- No olvidar hacer <workspace>source devel/setup.sh
- Crear un directorio launch dentro de <workspace>/src/mi_mapeo
- Copiar el archivo mi_gmapping.launch descargándolo desde el zip con mismo nombre en PRADO.

Launch File for gmapping

Este archivo launch lanza un mundo de turtlebot para hacer SLAM, pero observar el nuevo argumento “world_file” que permite configurar qué mundo de Gazebo usar.

```
<launch>
<!--  Launch turtle bot world  -->
 <include file="$(find turtlebot_gazebo)/launch/turtlebot_world.launch">
<arg name="world_file"
       value= /opt/ros/indigo/share/turtlebot_gazebo/worlds/corridor.world"/>

<!-- otro valor puede ser  "worlds/willowgarage.world"/>  -->

</include>

 <!-- Run gmapping node -->
 <node name = "gmapping" pkg = "gmapping" type="slam_gmapping"  />

<!-- Launch rviz -->
<include file="$(find turtlebot_rviz_launchers)/launch/view_robot.launch"/></launch>
```

Hacer de nuevo la experimentación

- Ejecutar teleoperación y comprobar cómo va reconociendo el mapa.
- Ahora vamos a añadir un sensor láser basado en el producto comercial “real” hokuyo láser.

Añadir un sensor Láser a TurtleBot

- El descubrimiento de mapas depende de la calidad del sensor de rango que se utilice.
  - Una kinect (basada en depth camera) tiene campo de visión limitado (60º en nuestro caso) y bastante ruido en las medidas.
  - Un láser tiene en general más campo de visión y las medidas son más precisas. El proceso de SLAM es más rápido y efectivo con un láser que con otro tipo de sensores de rango.
  - Vamos a añadir al modelo simulado de Turtlebot un nuevo sensor láser y comprobaremos la diferencia en el proceso de SLAM mapeo

Añadir un sensor láser a TurtleBot

Fuente: http://amanbreakingthings.blogspot.com.es/2014/11/adding-hokuyo-lidar-to-turtlebot-in-ros.html

1. Descargar el archivo SensorLaser.zip desde PRADO.
2. Contiene dos ficheros
    1. hokuyo.urdf.xacro
    2. trozo_xacro.xml
3. Copiar (sudo cp) el archivo hokuyo.urdf.xacro en
   /opt/ros/indigo/share/turtlebot_description/urdf/sensors
4. Crear una copia backup del archivo turtlebot_gazebo.urdf.xacro localizado en /opt/ros/indigo/share/turtlebot_description/urdf y modificarlo de la siguiente forma:
    1. Editar el fichero trozo_xacro.xml (del zip), seleccionar todo su contenido y copiar (Ctrl-c).
    2. Editar (sudo) el fichero turtlebot_gazebo.urdf.xacro
    3. Pegar el contenido copiado en turtlebot_gazebo.urdf.xacro, después de </xacro:macro> tag que cierra la macro que configura la Kinect. Guardar el archivo
5. Modificar el archivo (haciendo copia backup) /turtlebot_description/urdf/turtlebot_library.urdf.xacro añadiendo la línea
   <xacro:include filename="$(find turtlebot_description)/urdf/sensors/hokuyo.urdf.xacro"/>
6. (siguiente transparencia)

Añadir un sensor láser a Turtlebot

- Vamos a crear un nuevo fichero de descripción de robot.
  1. Ir al directorio de ros `/turtlebot_description/robots` hacemos una copia (sudo cp) del fichero `kobuki_hexagons_kinect.urdf.xacro` en `kobuki_hexagons_hokuyo.urdf.xacro`.
  2. Editar (sudo) este último fichero :
  3. Añadiendo
     ```
     <xacro:include filename="$(find turtlebot_description)/urdf/sensors/hokuyo.urdf.xacro"/>
     ```
     debajo de las lineas <xacro:include ….etc.
  4. Añadiendo
     ```
     <sensor_hokuyo parent="base_link"/>
     ```
     como última línea del fichero, antes de </robot>

Añadir un sensor láser a TurtleBot

- Esta configuración nueva de robot está disponible en la instalación de Turtlebot para ROS actual (si se reinstalan paquetes de Turtlebot hay que volver a hacerlo).
- La configuración del robot está guardada en el fichero `kobuki_hexagons_hokuyo.urdf.xacro` y es utilizada para modelar un robot Turtlebot con Láser comercial tipo Hokuyo en Gazebo
  - El fichero turtlebot_world.launch incluye un fichero kobuki.launch.xml
  - El fichero kobuki.launch.xml lanza gazebo_ros pasándole un fichero de descripción de robot que depende de parámetros definidos en variables de entorno.
- **IMPORTANTE!!!** Para usar el robot configurado con láser, antes de lanzar gazebo hay que modificar el valor de una variable de entorno de la siguiente forma
  - `export TURTLEBOT_3D_SENSOR="hokuyo"`

Repetimos el mapeo con láser

- Lanzar Gazebo con Turtlebot y Láser

```
$ export TURTLEBOT_3D_SENSOR= “hokuyo”
$ roslaunch mi_mapeo mi_gmapping
```

- Ahora arrancar teleoperación, observar que requerimos menos tiempo para reconocer el mapa y en cada actualización la cantidad de mapa conocido es mayor.

(C)2016 Roi Yehoshua

Localization

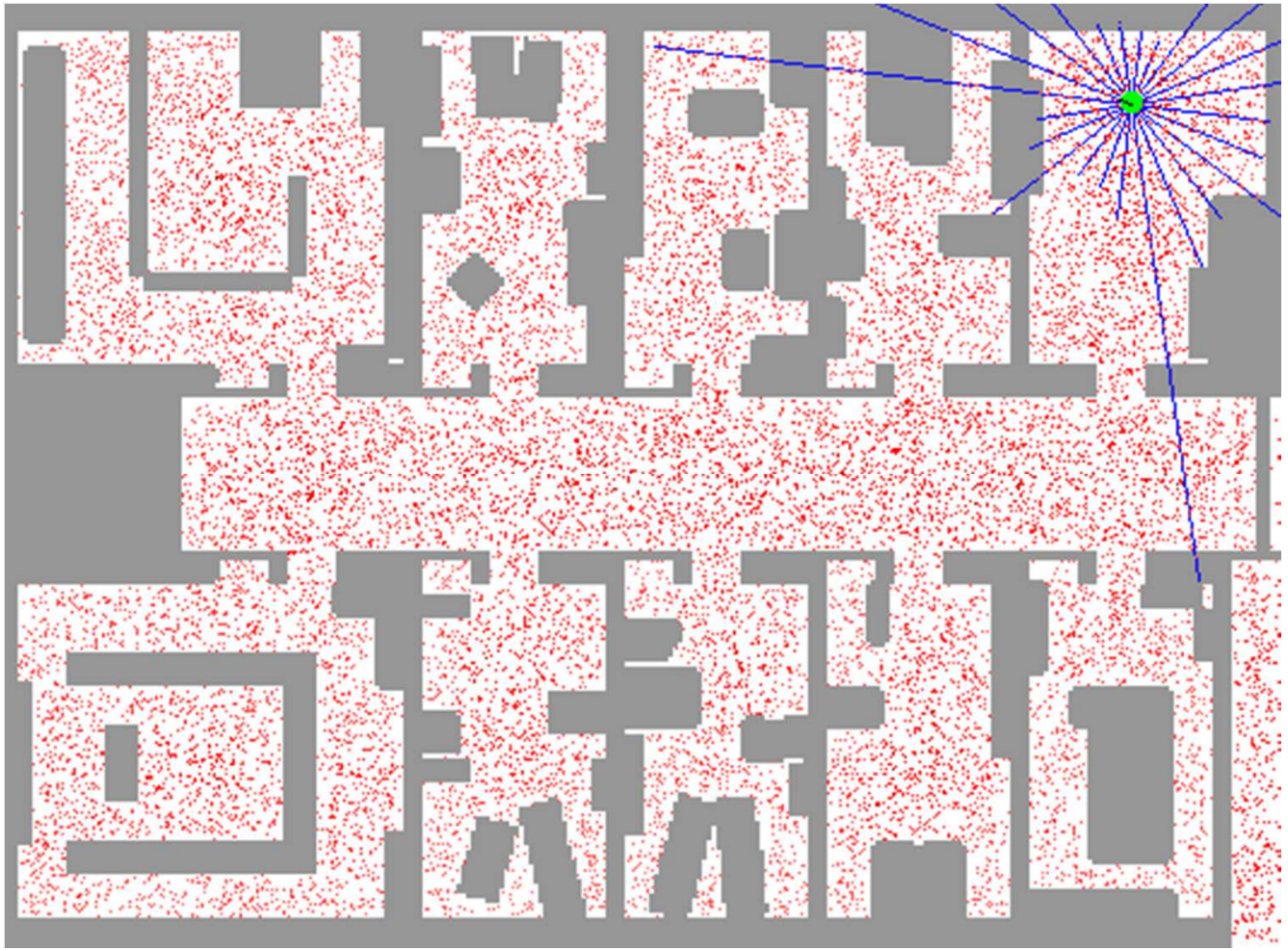
- **Localization** es el problema de estimar la pose de un robot relativa a un mapa conocido.
- Localization no es muy sensible a la situación exacta de objetos, por lo que puede manejar pequeños cambios en los emplazamientos de objetos.
- ROS usa el paquete **amcl** para localización

(C)2015 Roi Yehoshua

- amcl es un sistema de localización probabilístico para un robot en 2D
- Implementa la técnica **Adaptive Monte Carlo Localization** que usa un filtro de partículas para registrar y actualizar la pose de un robot en un mapa conocido
- El algoritmo está descrito en el libro **Probabilistic Robotics** by Thrun, Burgard, and Fox (http://www.probabilistic-robotics.org/)
- amcl funciona solo con scaner láser
  - Aunque se puede extender para trabajar con otros sensores.

(C)2015 Roi Yehoshua

AMCL - Particle Filter



(C)2016 Roi Yehoshua

- amcl toma un mapa, laser scans, y devuelve estmimaciones de la pose del robot
- Subscribed topics:
  - scan – Laser scans
  - tf – Transforms
  - initialpose – Mean and covariance with which to (re-) initialize the particle filter
  - map – the map used for laser-based localization
- Published topics:
  - **amcl_pose** – Robot's estimated pose in the map, with covariance.
  - Particlecloud – The set of pose estimates being maintained by the filter

(C)2015 Roi Yehoshua

Localización en un mapa conocido

- amcl_demo.launch - launch file for navigation demo

```
<launch>
 <!-- Map server -->
 <arg name="map_file" default="$(env TURTLEBOT_GAZEBO_MAP_FILE)"/>
 <node name="map_server" pkg="map_server" type="map_server" args="$(arg map_file)" />

 <!-- Localization -->
 <arg name="initial_pose_x" default="0.0"/>
 <arg name="initial_pose_y" default="0.0"/>
 <arg name="initial_pose_a" default="0.0"/>
 <include file="$(find turtlebot_navigation)/launch/includes/amcl.launch.xml">
   <arg name="initial_pose_x" value="$(arg initial_pose_x)"/>
   <arg name="initial_pose_y" value="$(arg initial_pose_y)"/>
   <arg name="initial_pose_a" value="$(arg initial_pose_a)"/>
 </include>

 <!-- Move base -->
 <include file="$(find turtlebot_navigation)/launch/includes/move_base.launch.xml"/>
</launch>
```

(C)2016 Roi Yehoshua

Localización en un mapa conocido

- Launch Gazebo with turtlebot

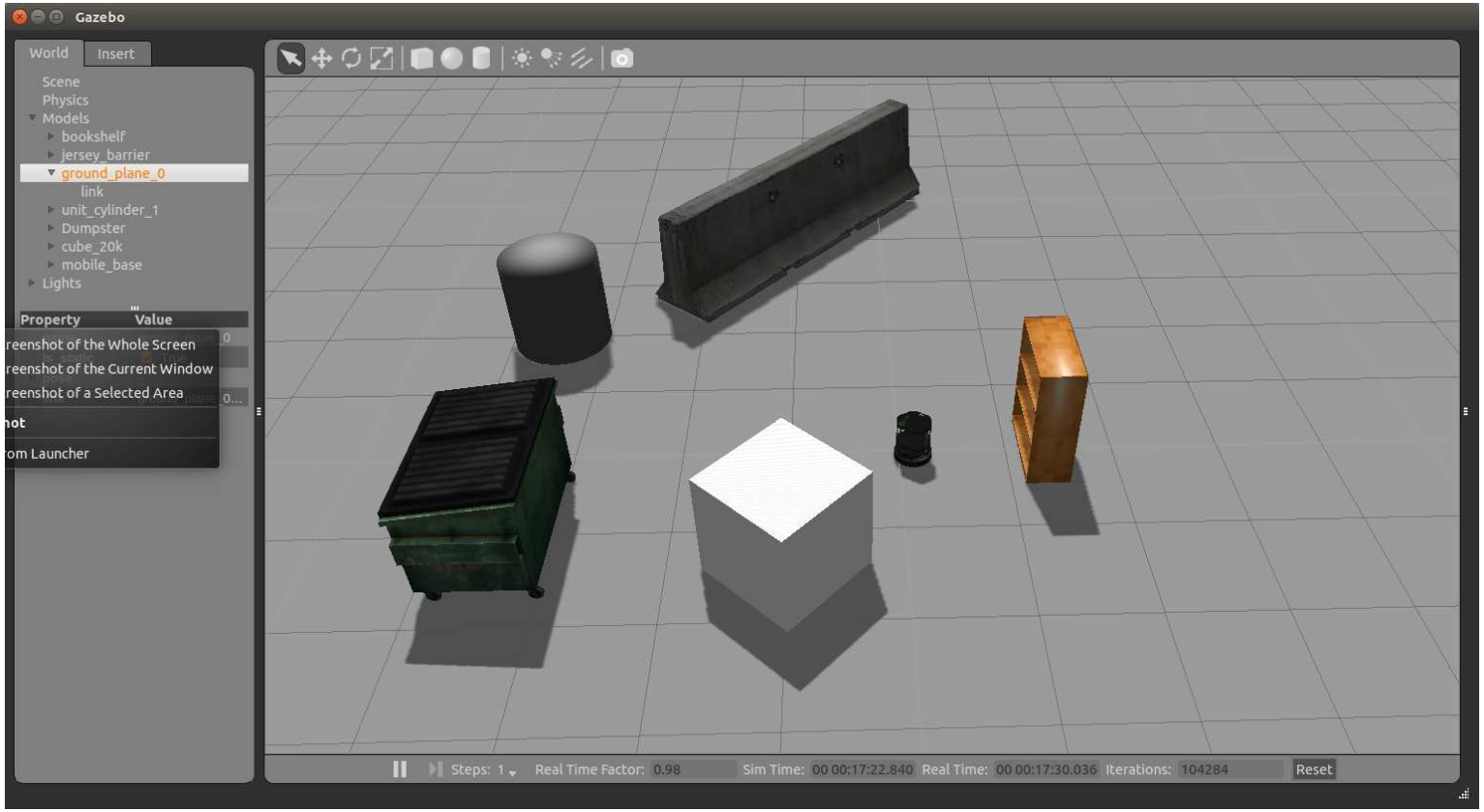
```
$ roslaunch turtlebot_gazebo turtlebot_world.launch
```

- Run the amcl demo

```
$ roslaunch turtlebot_gazebo amcl_demo.launch
```

(C)2016 Roi Yehoshua

Localización en un mapa conocido



(C)2016 Roi Yehoshua

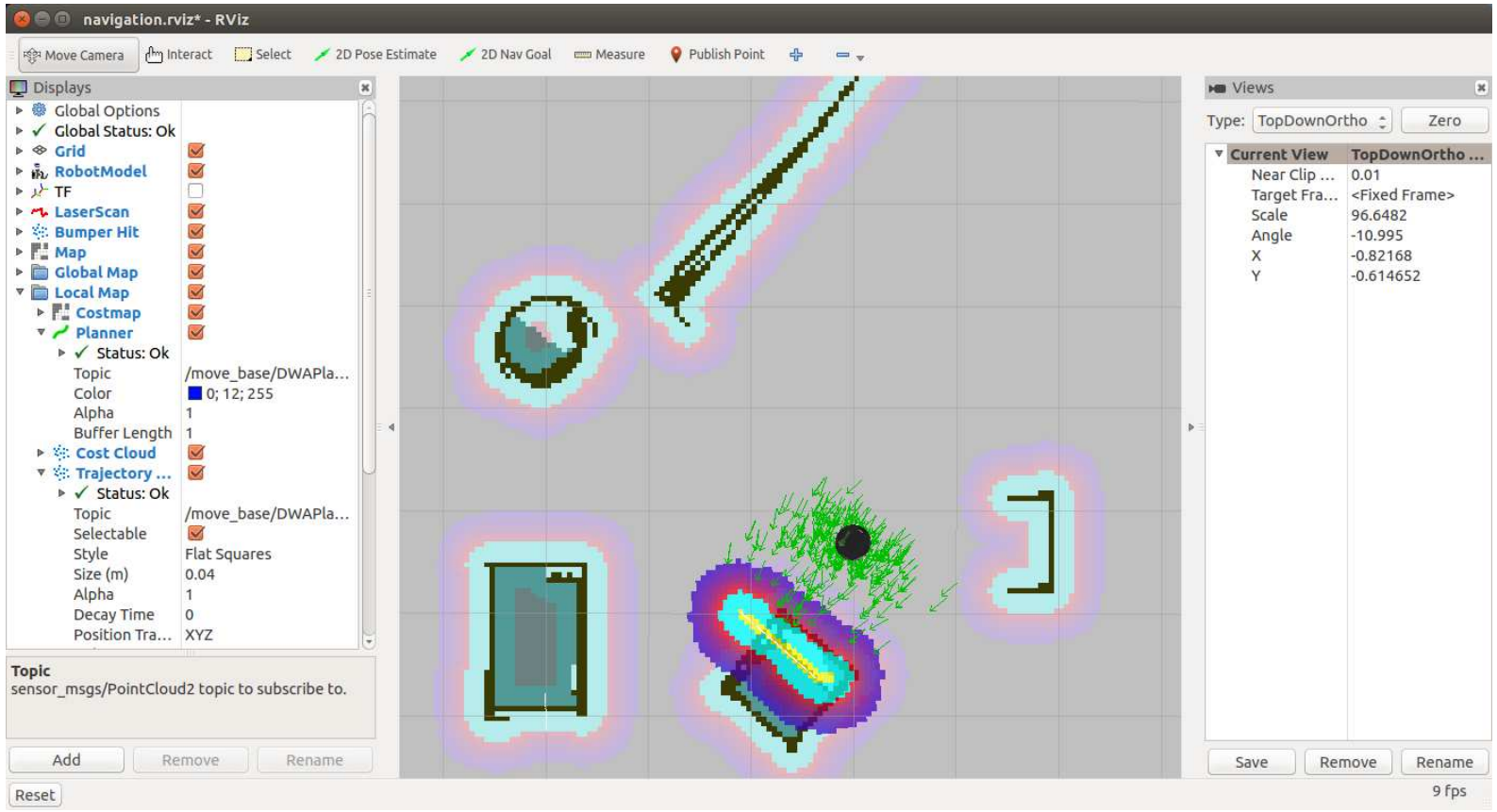
rviz with Localization

- rviz nos permite:
  - Proporcionar una localización aproximada del robot (el robot no sabe dónde está cuando arranca)
  - Enviar goals al robot.
  - Visualizar información relevante a la localización y navegación (en la siguiente sesión)(planned path, costmap, etc.)
- Launch rviz:

```
$ roslaunch turtlebot_rviz_launchers view_navigation.launch
```

(C)2016 Roi Yehoshua

rviz with Navigation Stack



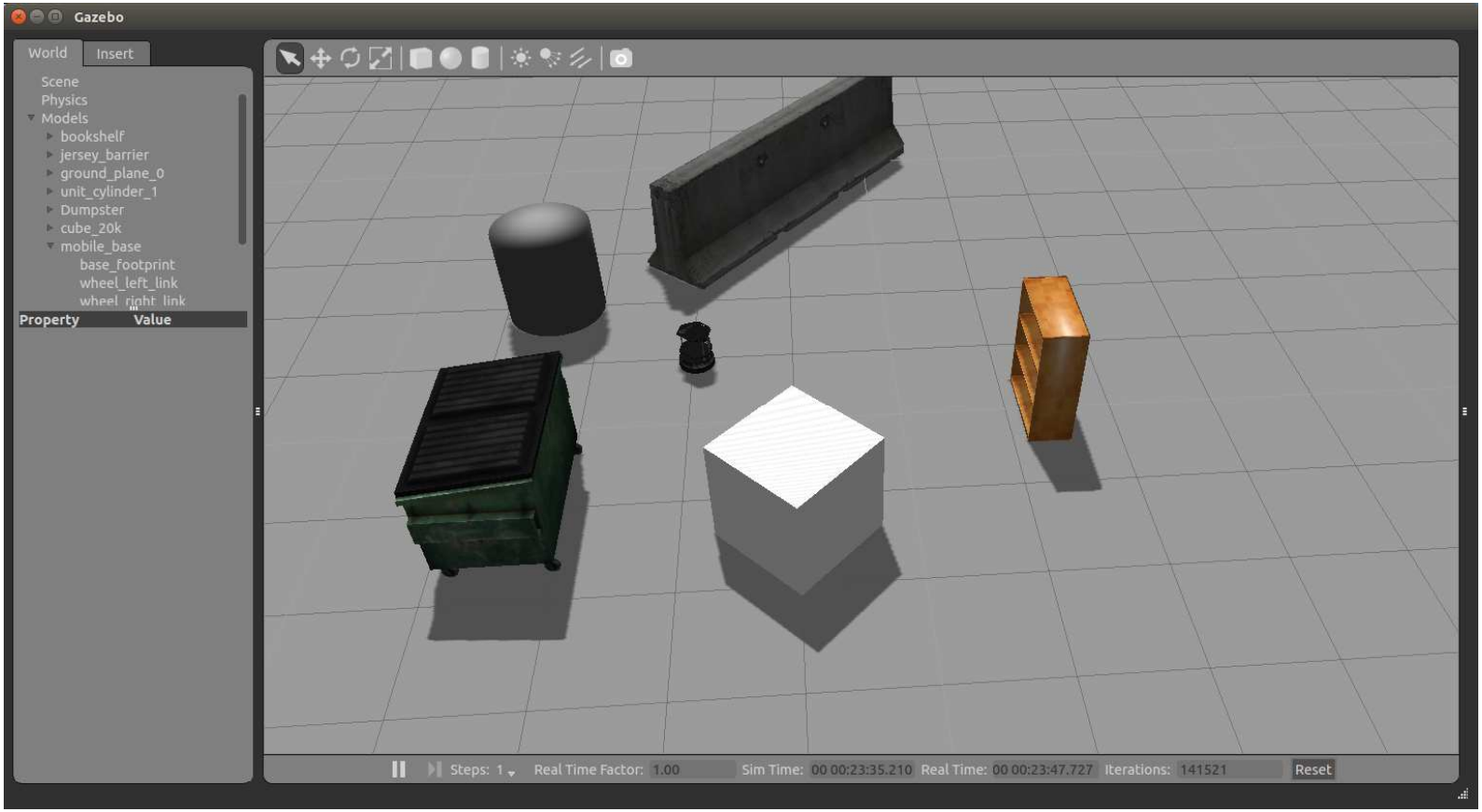
(C)2016 Roi Yehoshua

Localizar Turtlebot.

- El robot no sabe donde está al principio.
- For example, let’s move the robot in Gazebo to (-1,-2)
- Now to provide it its approximate location on the map:
  - Click the "2D Pose Estimate" button
  - Click on the map where the TurtleBot approximately is and drag in the direction the TurtleBot is pointing
- You will see a collection of arrows which are hypotheses of the position of the TurtleBot
- The laser scan should line up approximately with the walls in the map
  - If things don't line up well you can repeat the procedure

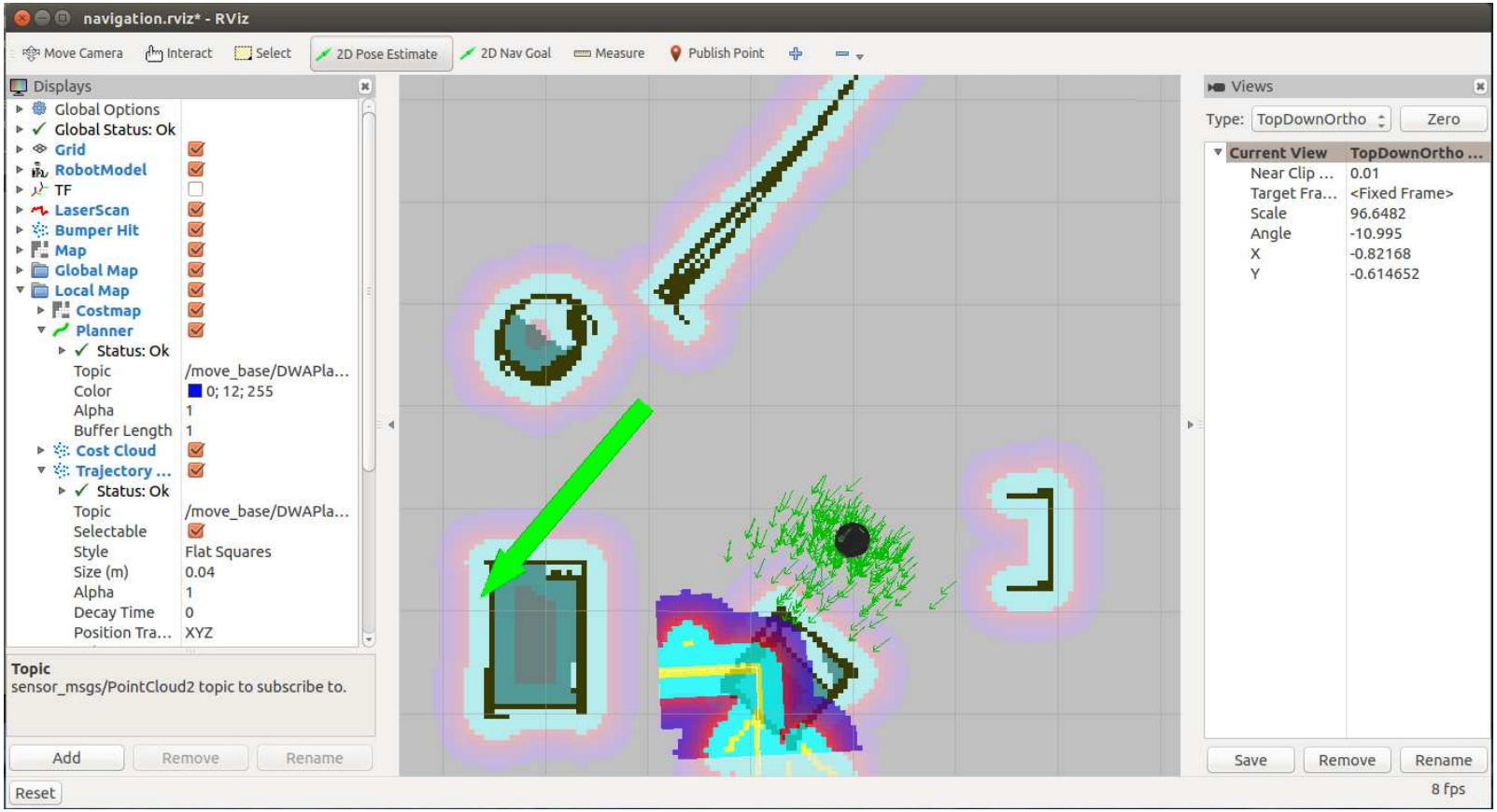
(C)2016 Roi Yehoshua

Localize the TurtleBot



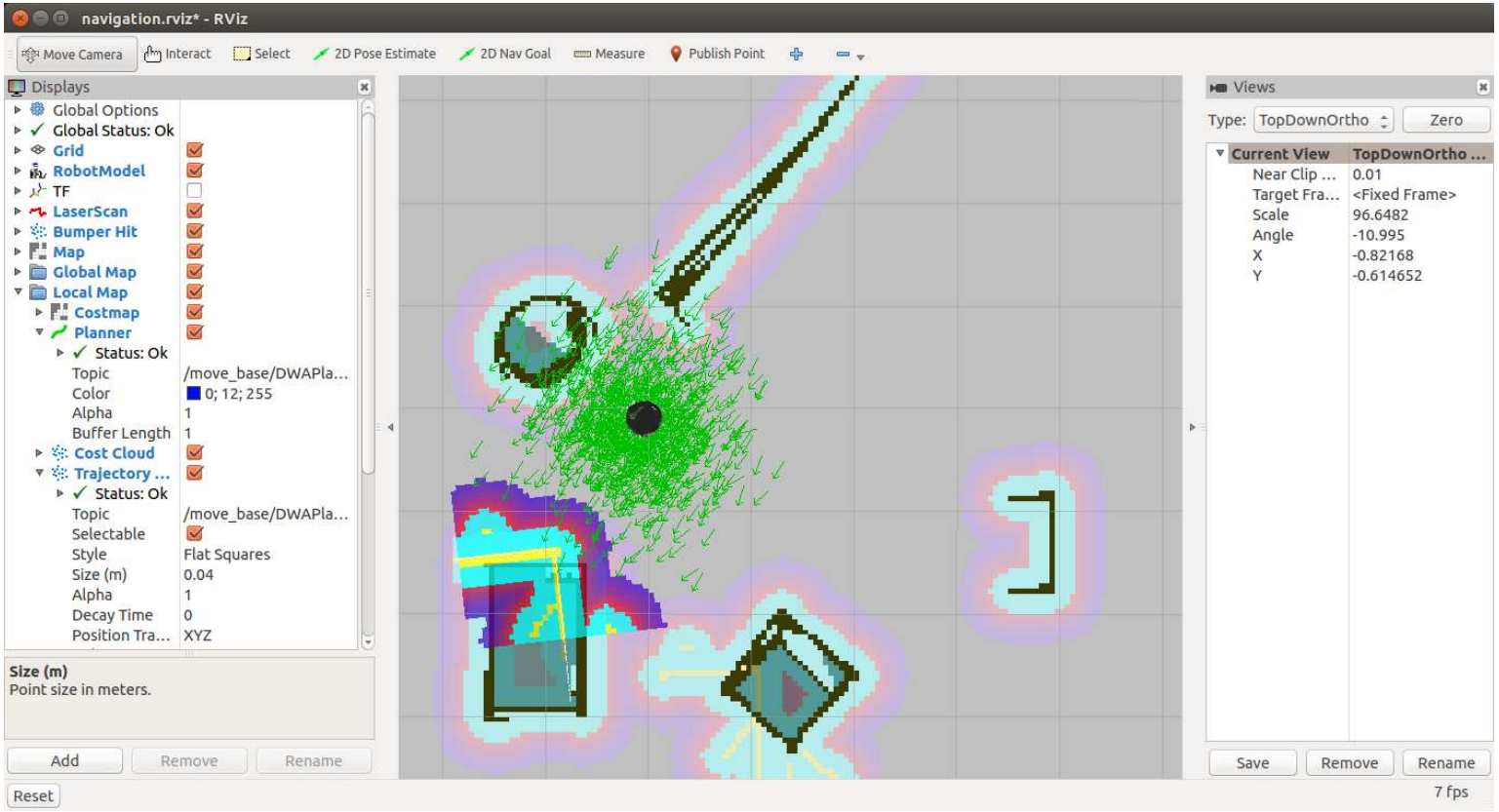
(C)2016 Roi Yehoshua

Localize the TurtleBot



(C)2016 Roi Yehoshua

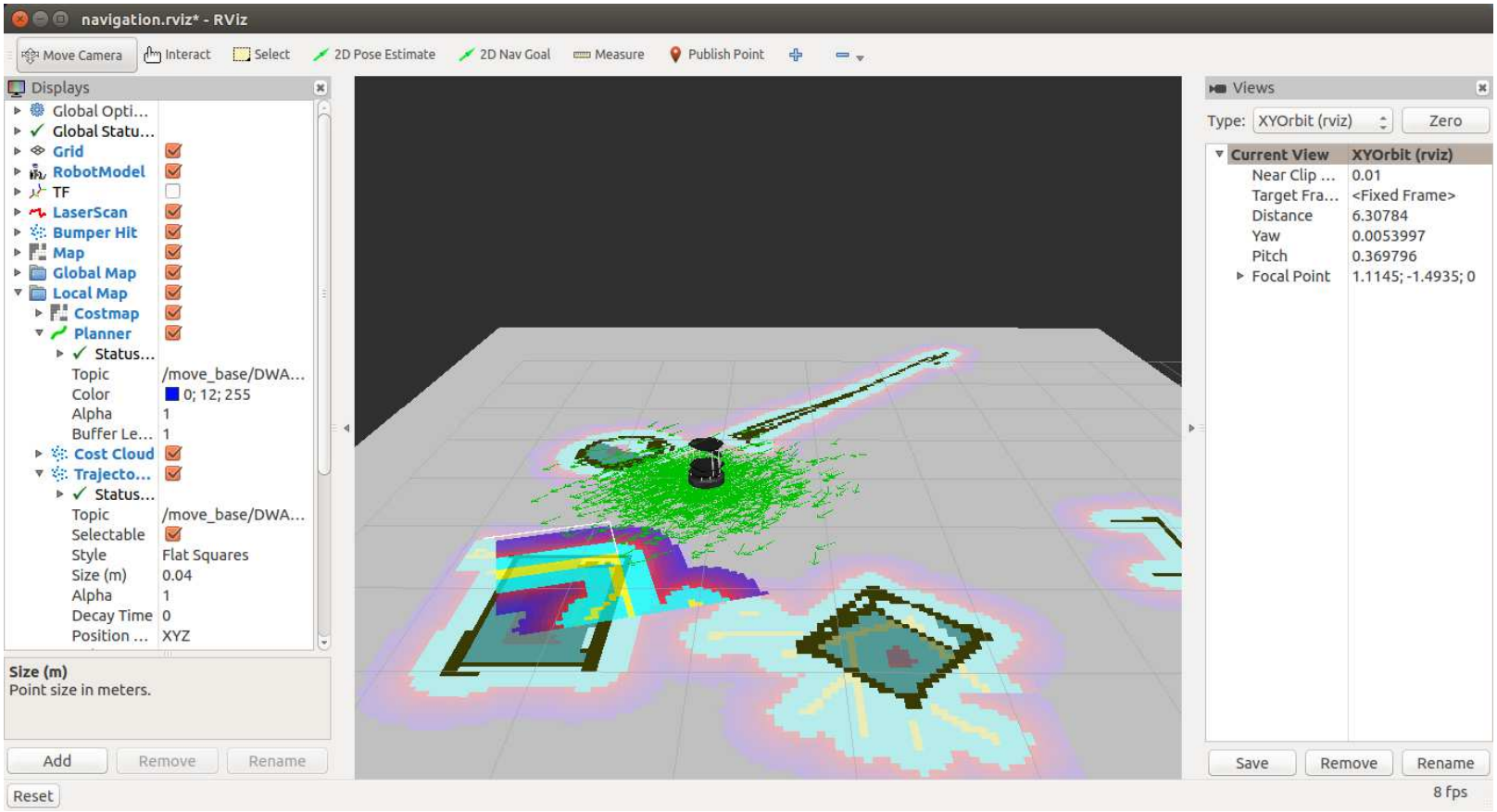
Localize the TurtleBot



(C)2016 Roi Yehoshua

Localize the TurtleBot

- You can change the current view (on right panel):



(C)2016 Roi Yehoshua

Particle Cloud in rviz

- The **Particle Cloud** display shows the particle cloud used by the robot's localization system
- The spread of the cloud represents the localization system's uncertainty about the robot's pose
- As the robot moves about the environment, this cloud should shrink in size as additional scan data allows amcl to refine its estimate of the robot's position and orientation
- To watch the particle cloud in rviz:
  - Click Add Display and choose Pose Array
  - Set topic name to /particlecloud

(C)2016 Roi Yehoshua

Localización en un mapa desconocido

- Descargar el fichero mi_gmapping_amcl.launch desde PRADO

```
<launch>
<!--  Launch turtle bot world  -->
 <include file="$(find turtlebot_gazebo)/launch/turtlebot_world.launch">
  <arg name="world_file" value= "/opt/ros/indigo/share/turtlebot_gazebo/worlds/corridor.world"/> <!-- otro valor
puede ser  "worlds/willowgarage.world"/>  -->
 </include>

 <!-- Run gmapping node -->
 <node name = "gmapping" pkg = "gmapping" type="slam_gmapping"  />


<!-- Localization -->
 <arg name="initial_pose_x" default="0.0"/>
 <arg name="initial_pose_y" default="0.0"/>
 <arg name="initial_pose_a" default="0.0"/>
 <include file="$(find turtlebot_navigation)/launch/includes/amcl.launch.xml">
   <arg name="initial_pose_x" value="$(arg initial_pose_x)"/>
   <arg name="initial_pose_y" value="$(arg initial_pose_y)"/>
   <arg name="initial_pose_a" value="$(arg initial_pose_a)"/>
   <arg name="use_map_topic"  value ="true"/>
 </include>

 <!-- Launch rviz -->
 <include file="$(find turtlebot_rviz_launchers)/launch/view_robot.launch"/>
</launch>
```

Teleoperacion para descubrir el mapa

- Vamos a ejecutar el código fuente que vamos a usar como base para implementar el explorador de fronteras.

- Borrar el directorio mi_mapeo.zip
- Descargar mi_mapeo.zip
- Descomprimir en vuestro espacio de trabajo
- Hacer
  - catkin_make
  - source devel/setup.sh
- Ejecutar en tres terminales
  - roslaunch mi_mapeo mi_gmapping_amcl.launch
  - rosrun mi_mapeo mi_mapeo
  - roslaunch turtlebot_teleop keyboard…