PAL Robotics

Université de Bourgogne
November 18th and 19th 2013

VIBOT tutorial on REEM simulation

Paul Mathieu
Jordi Pages

Overview

1. **Introduction**
2. Installation
3. REEM in Gazebo
4. Visualization in rviz
5. Motions
6. Perception
7. Autonomous navigation
8. Remote lab demo

Introduction

Overview

1. Introduction
2. **Installation**
3. REEM in Gazebo
4. Visualization in rviz
5. Motions
6. Perception
7. Autonomous navigation
8. Remote lab demo

Installation

ROS Fuerte installation

Add ROS repositiories:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu precise main" > /etc/apt/sources.list.d/ros-latest.list'

wget http://packages.ros.org/ros.key -O - | sudo apt-key add -
```

Install the following packages:

```
sudo apt-get update
sudo apt-get install ros-fuerte-desktop-full
sudo apt-get install ros-fuerte-arm-navigation
sudo apt-get install ros-fuerte-arm-navigation-experimental
sudo apt-get install ros-fuerte-pr2-simulator
sudo apt-get install ros-fuerte-pr2-interactive-manipulation
sudo apt-get install ros-fuerte-vision-visp
sudo apt-get install ros-fuerte-turtlebot-apps
sudo apt-get install ros-fuerte-urdfdom-headers
sudo apt-get install ros-fuerte-console-bridge
sudo apt-get install ros-fuerte-urdfdom
```

Installation

bullet 2.81 installation

cd ~/Downloads

For 32-bit computers:

wget http://packages.osrfoundation.org/gazebo/ubuntu/pool/main/b/bullet/libbullet_2.81-2ppa1~precise_i386.deb

sudo dpkg -i libbullet_2.81-2ppa1~precise_i386.deb

For 64-bit computers:

wget http://packages.osrfoundation.org/gazebo/ubuntu/pool/main/b/bullet/libbullet_2.81-2ppa1~precise_amd64.deb

sudo dpkg -i libbullet_2.81-2ppa1~precise_amd64.deb

Installation

Gazebo 1.8.6 installation

cd ~/Downloads

In case of a 32-bit computer:

wget http://www.gazebosim.org/assets/distributions/gazebo_1.8.6-1~precise_i386.deb

otherwise, for 64-bit computers:

wget http://www.gazebosim.org/assets/distributions/gazebo_1.8.6-1~precise_amd64.deb

Install the debian:

for 32 bits:

sudo dpkg -i gazebo_1.8.6-1~precise_i386.deb

for 64 bits:

sudo dpkg -i gazebo_1.8.6-1~precise_amd64.deb

In order to install the missing dependencies which dpkg does not install:

sudo apt-get -f install

Installation

reem-education stacks installation

Install rosinstall:

```
sudo apt-get install python-rosinstall
```

Create the workspace were PAL ROS packages will be installed:

```
cd
mkdir reem-education
cd reem-education
```

Once in the selected workspace proceed with the installation of the PAL ROS packages:

```
rosinstall . https://raw.github.com/reem-education/stacks/fuerte/reem-education-ros-fuerte-gazebo-1.8.6-standalone.rosinstall
```

Installation

Setup environment

Edit ~/.bashrc and append the following lines:

```
. /opt/ros/fuerte/setup.bash
. /usr/share/gazebo-1.8/setup.sh

export ROS_PACKAGE_PATH=$HOME/reem-education/stacks:/opt/ros/fuerte/share:/opt/ros/fuerte/stacks

export GAZEBO_PLUGIN_PATH=`rospack find ros_control_gazebo_plugin`/lib:$GAZEBO_PLUGIN_PATH
export GAZEBO_PLUGIN_PATH=`rospack find atlas_msgs`/lib:$GAZEBO_PLUGIN_PATH
export GAZEBO_PLUGIN_PATH=`rospack find pal_gazebo_plugins`/lib:$GAZEBO_PLUGIN_PATH
export GAZEBO_MODEL_PATH=`rospack find reem_gazebo`/models:$GAZEBO_MODEL_PATH
export GAZEBO_RESOURCE_PATH=`rospack find reem_gazebo`/reem_gazebo/worlds:
$GAZEBO_RESOURCE_PATH
```

Install third party dependencies

Open a new terminal and execute:

```
rosdep install siftgpu
```

Build the reem-education packages

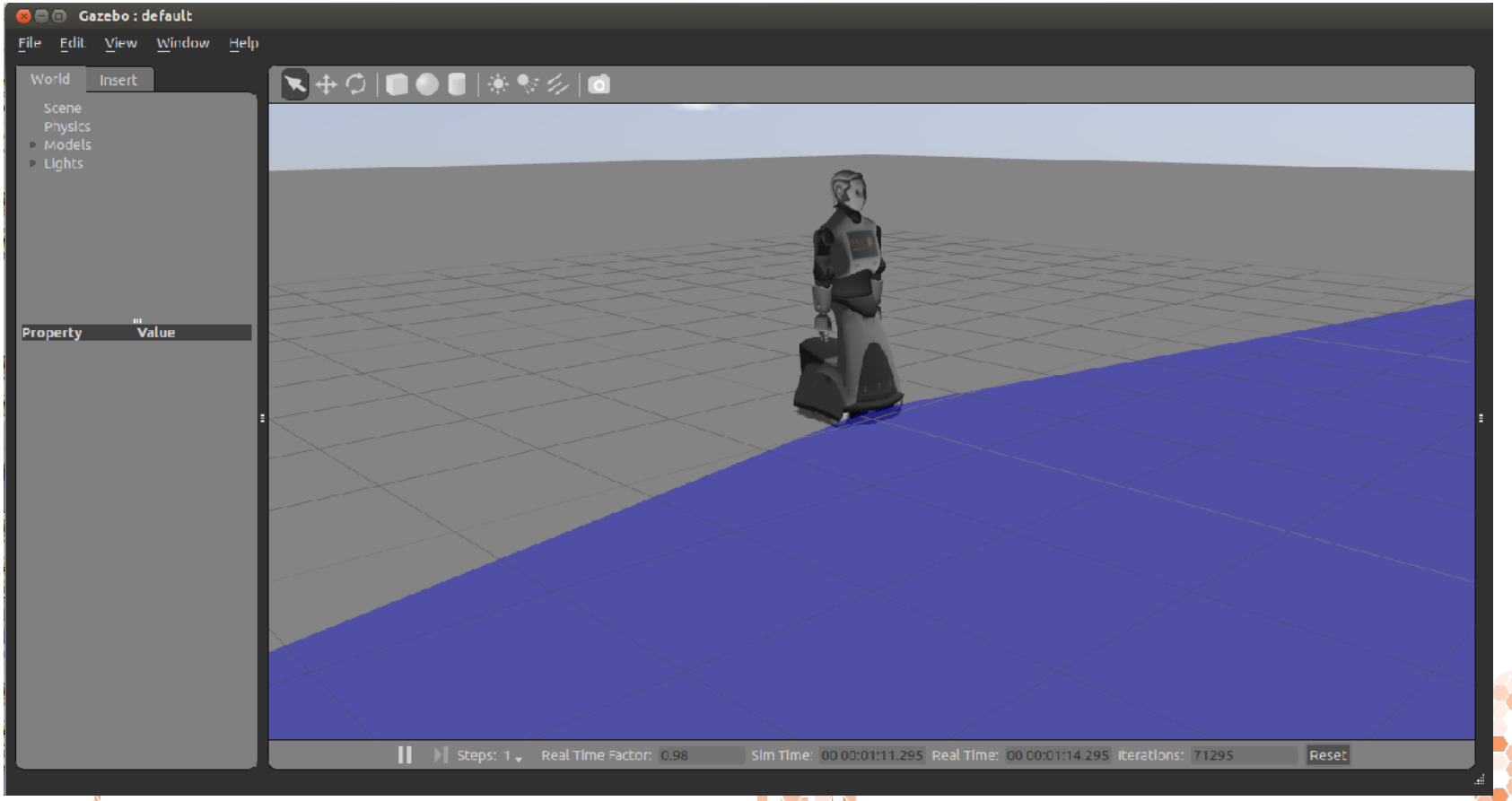
```
rosmake -a --pre-clean
```

Overview

1. Introduction
2. Installation
3. **REEM in Gazebo**
    1. Launch REEM
    2. Create world
    3. Building editor
    4. Save and load world
    5. Pre-defined worlds
    6. Load pre-defined worlds
4. Visualization in rviz
5. Motions
6. Perception
7. Autonomous navigation
8. Remote lab demo

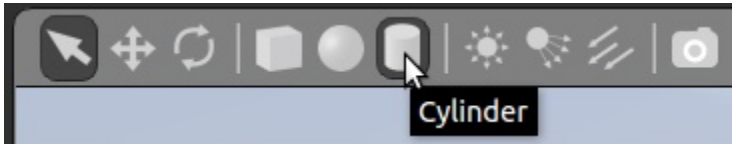
Launch REEM

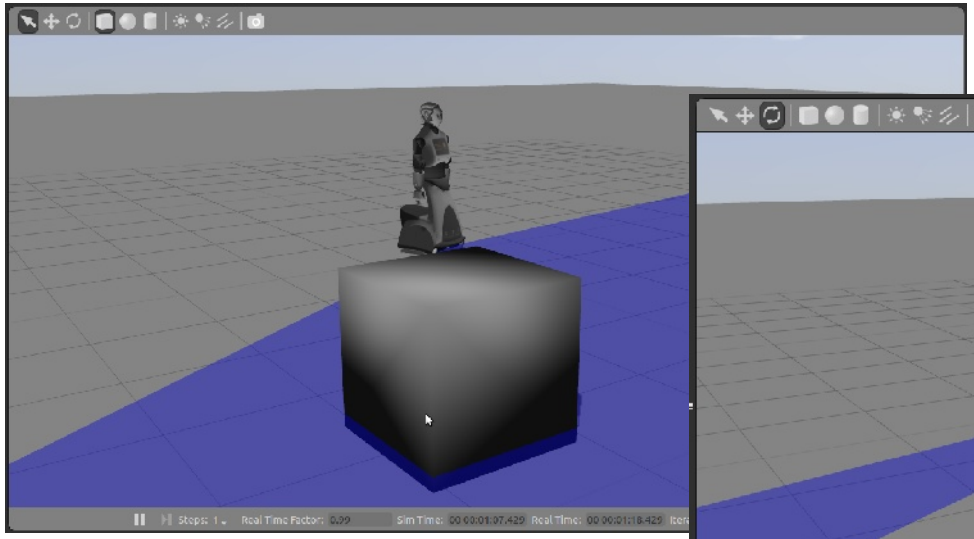
```
roslaunch reem_gazebo reem_gazebo.launch
```

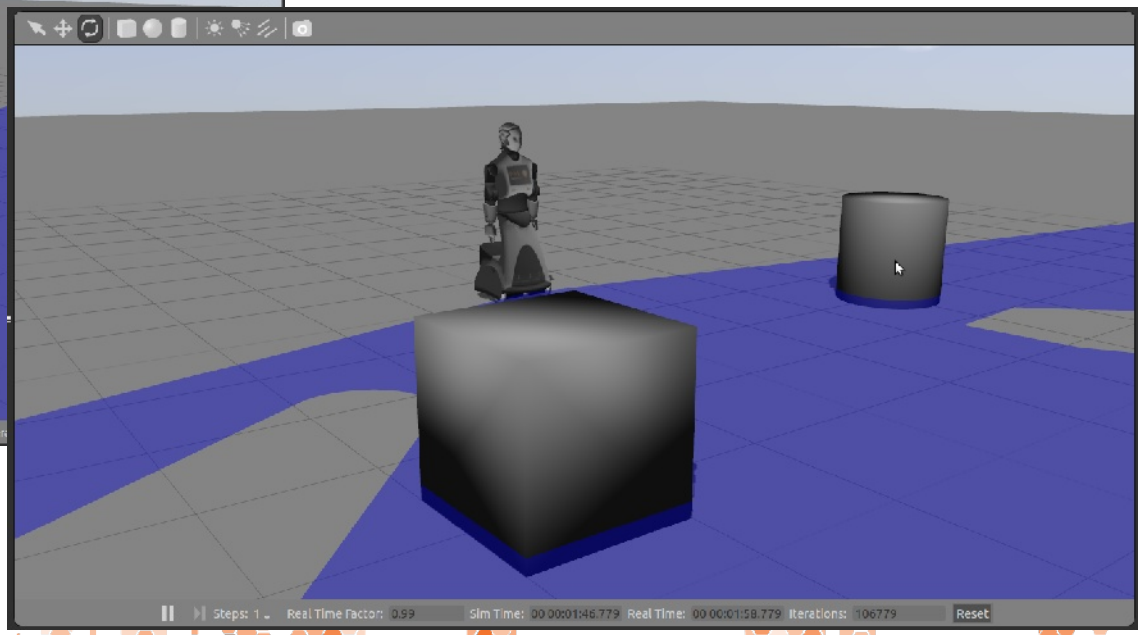


Create world

Add simple shapes

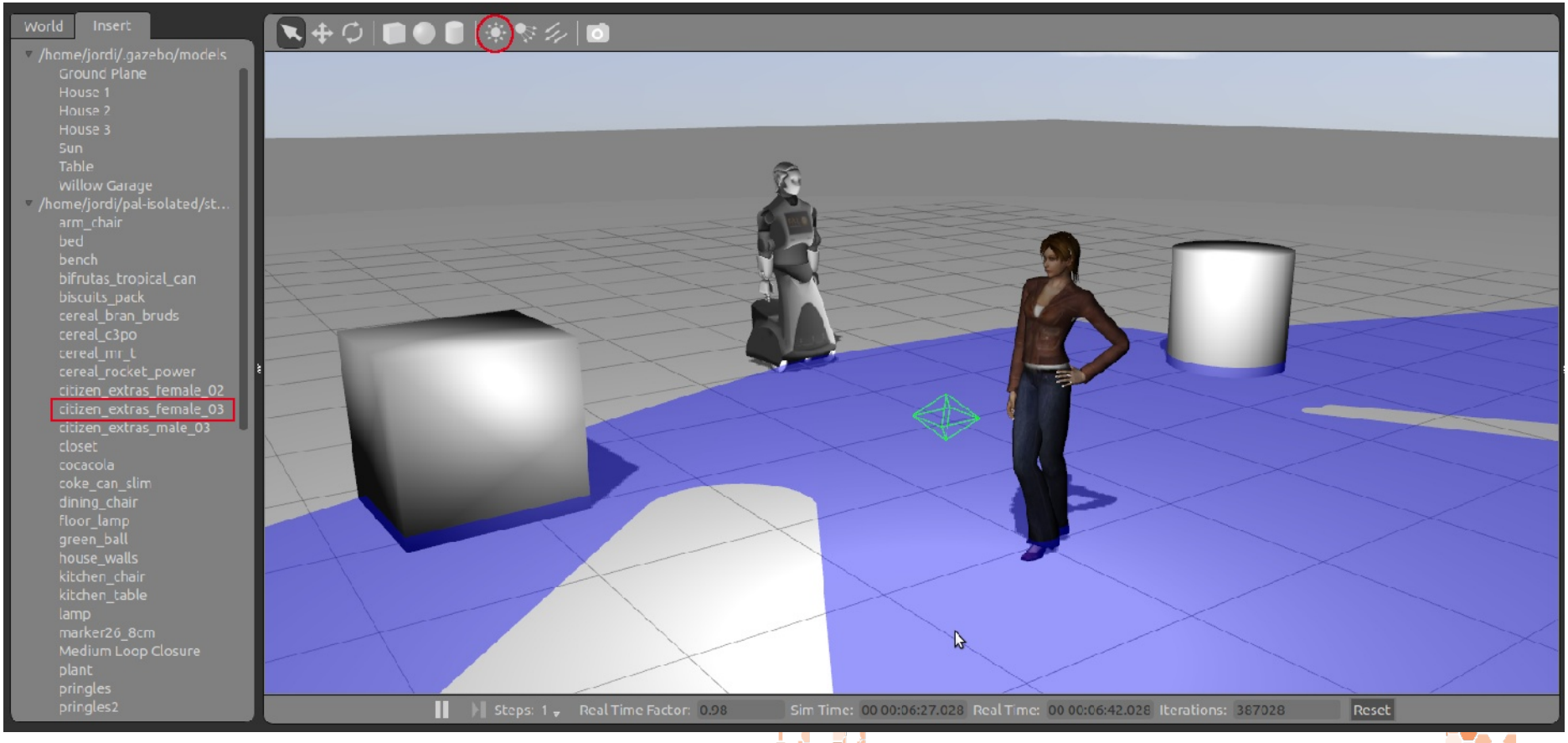






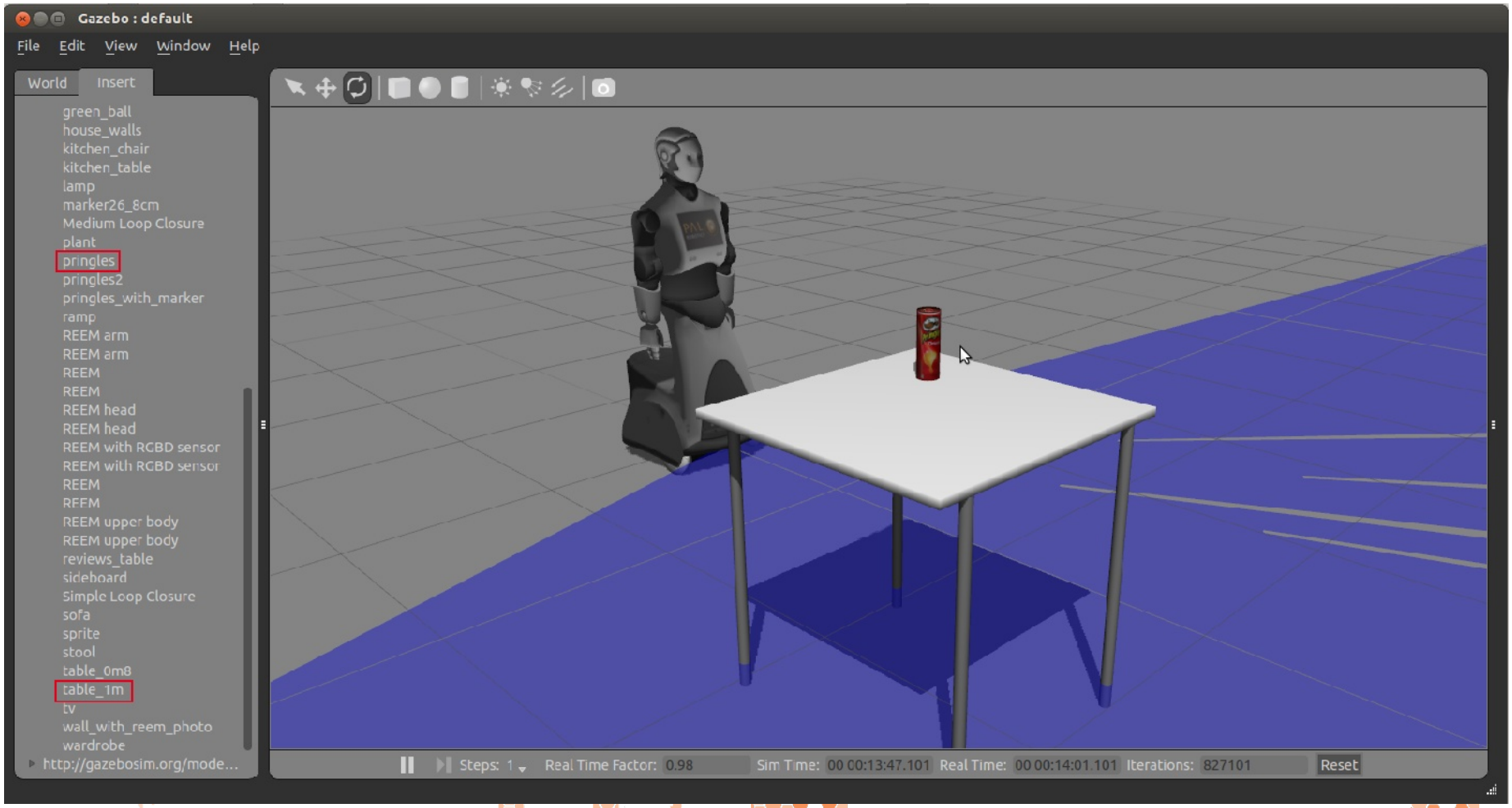
Create world

Add complex models and lights

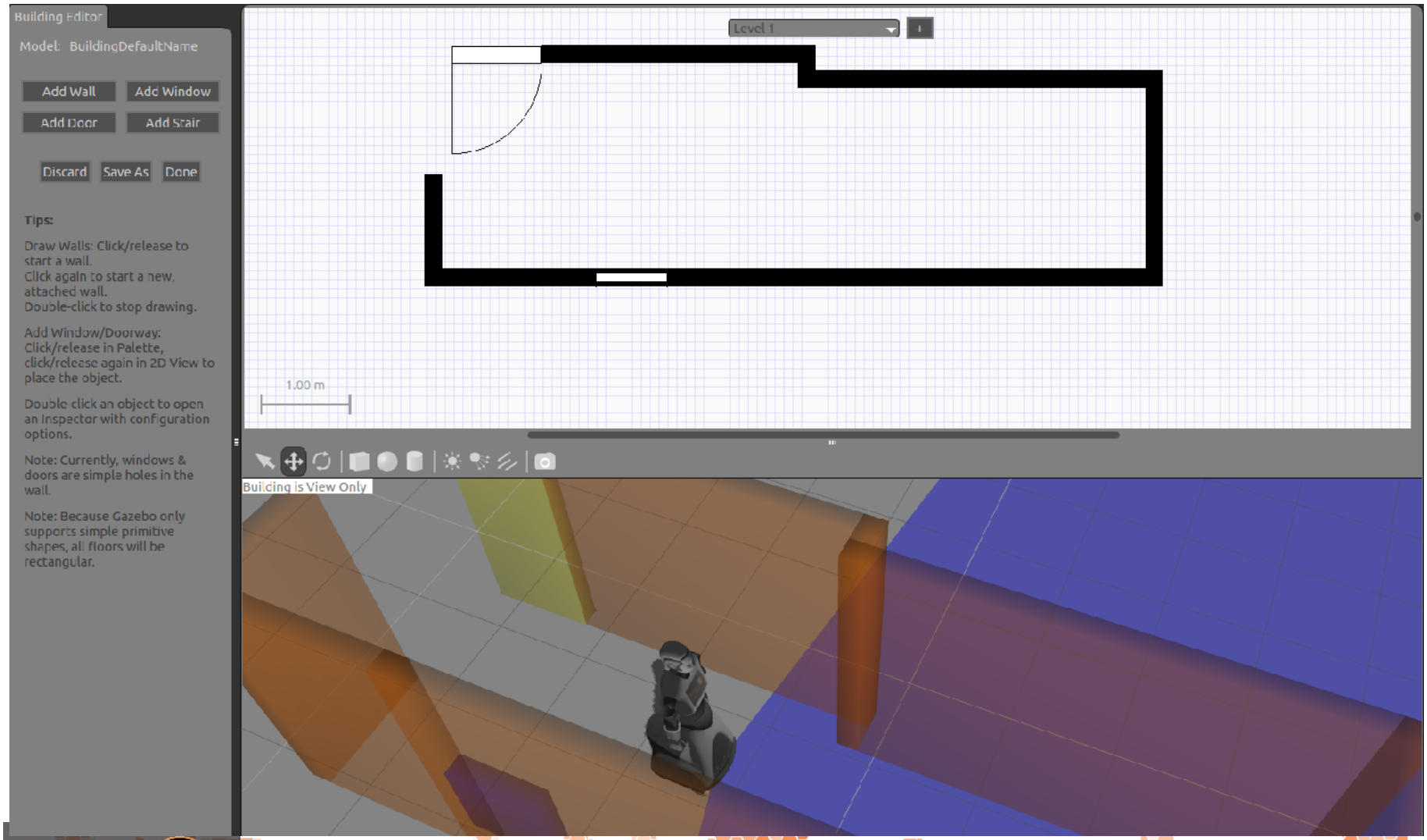


Create world

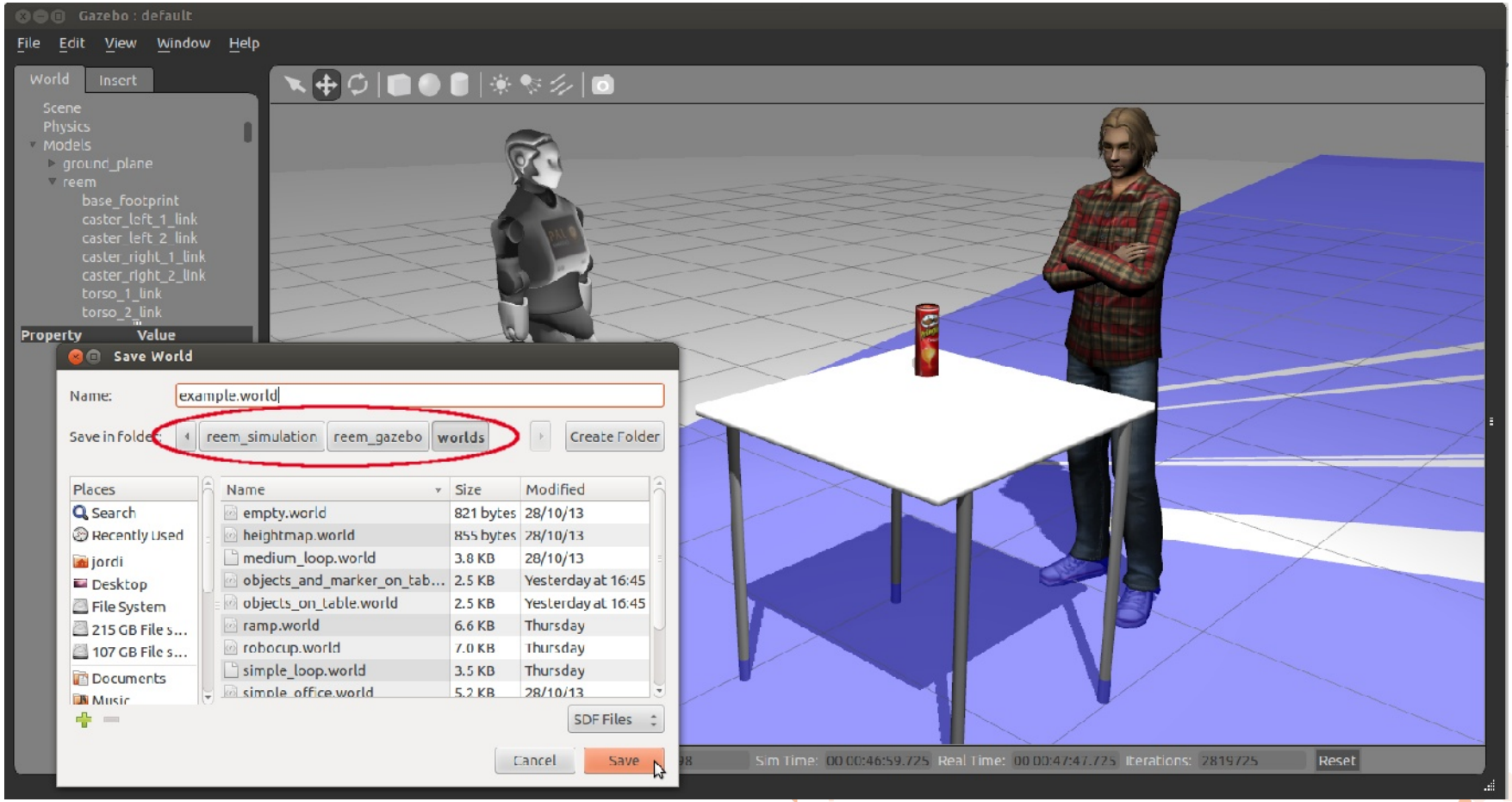
Stack models



Building editor



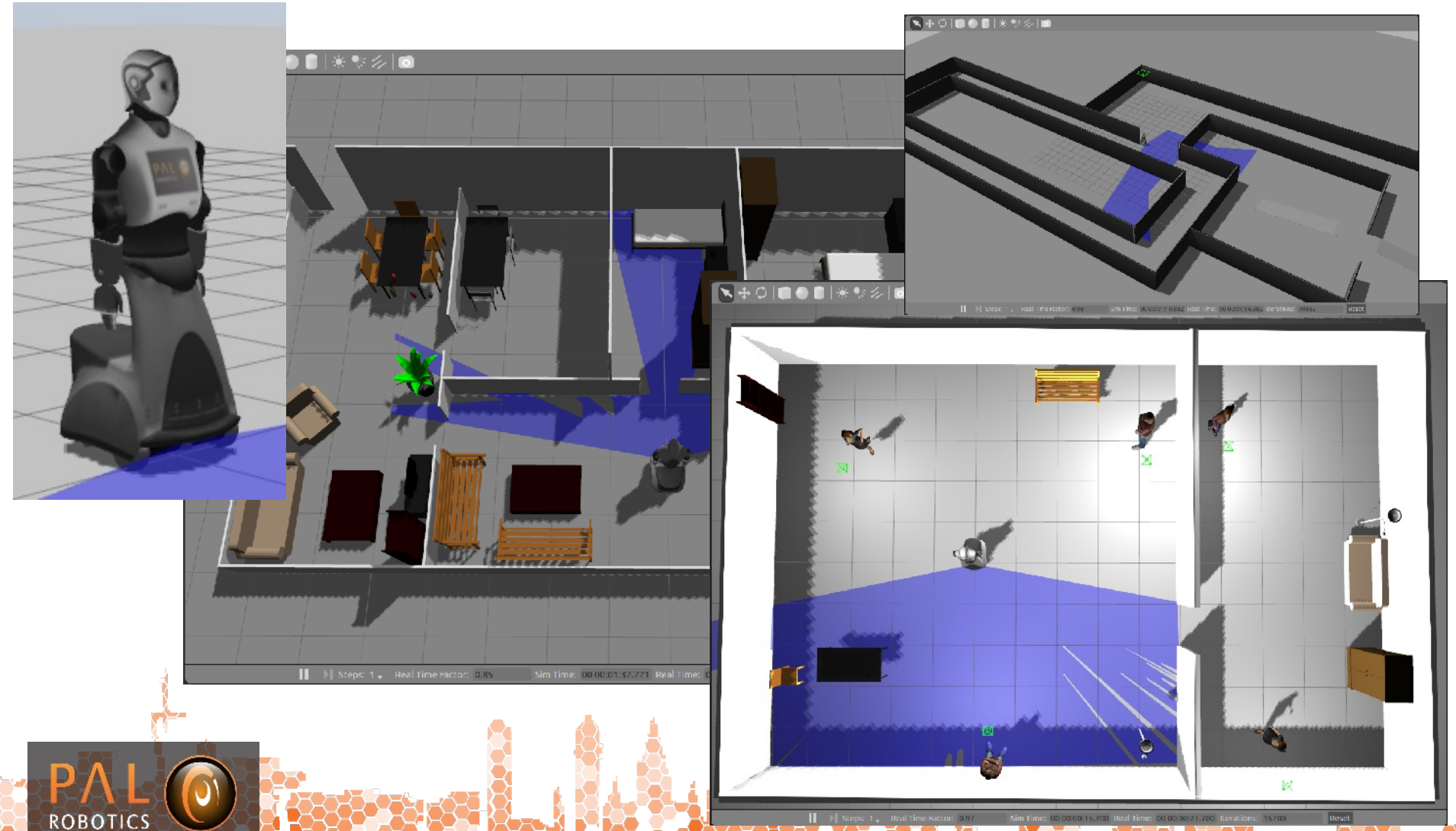
Save and load world




```
roslaunch reem_gazebo load_saved_world.launch world:=example
```

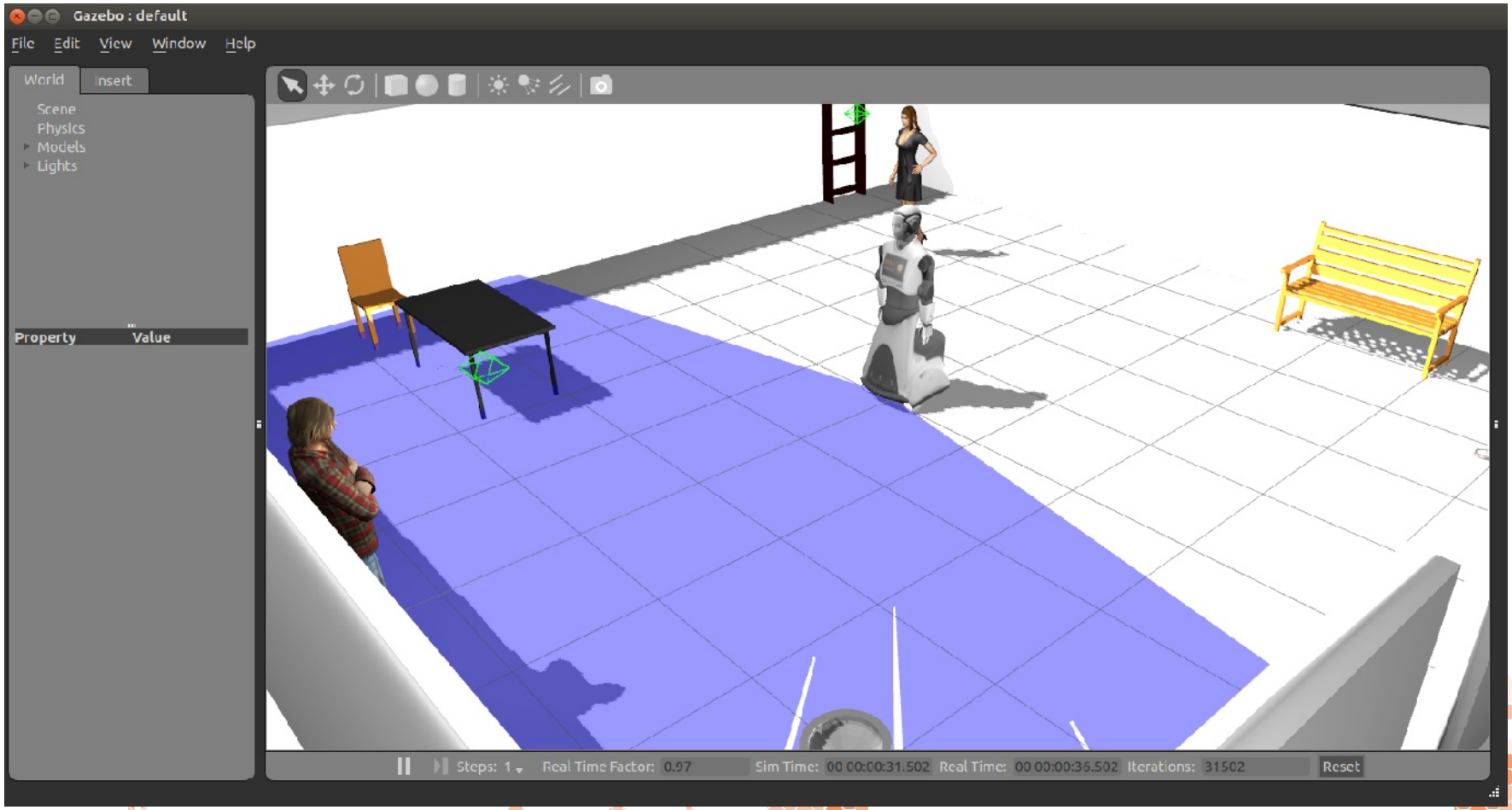
Pre-defined worlds

Several pre-defined worlds are provided in **reem_gazebo/worlds**



Load pre-defined worlds

```
roslaunch reem_gazebo reem_gazebo.launch world:=simple_office_people
```



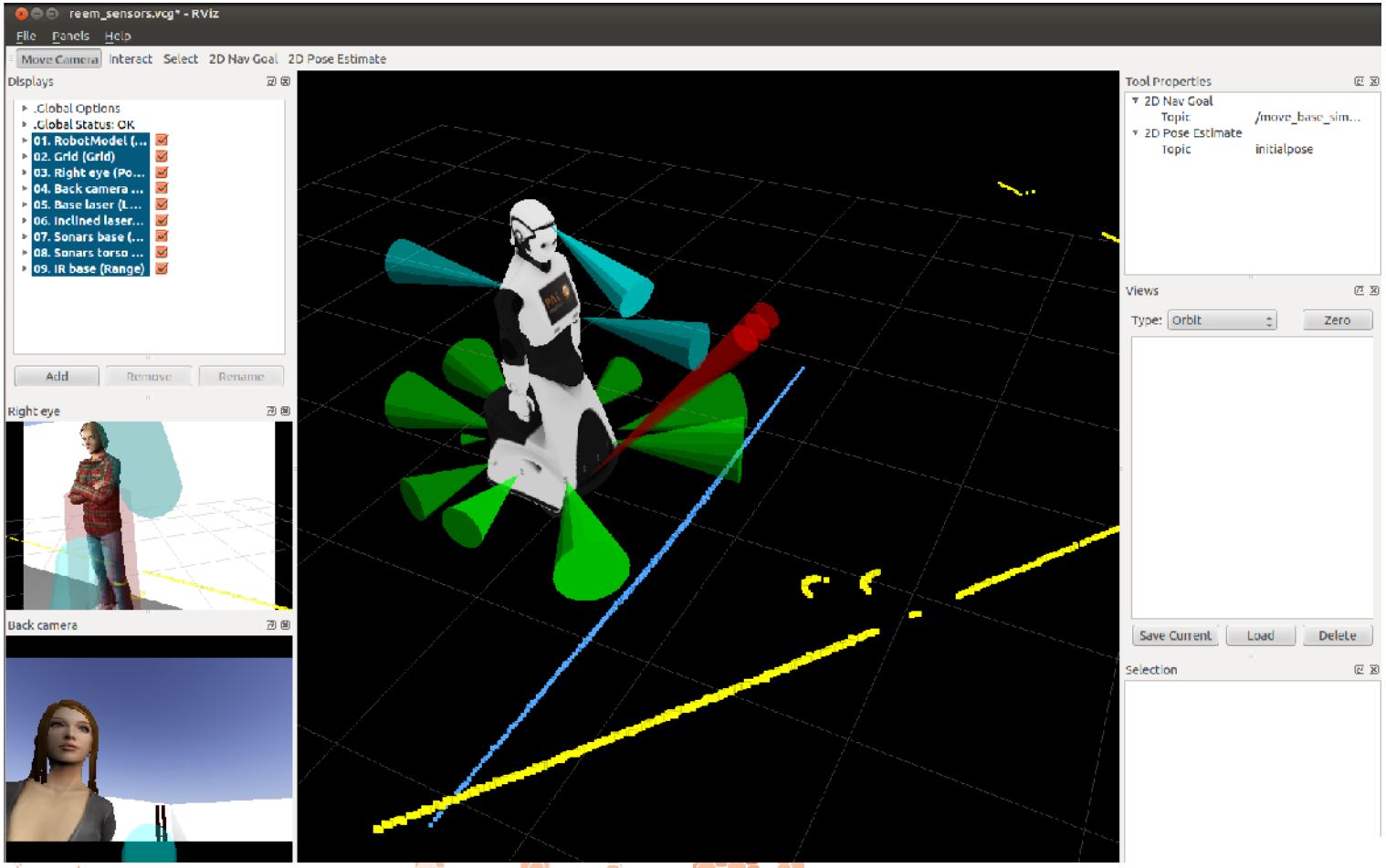
Overview

1. Introduction
2. Installation
3. REEM in Gazebo
4. **Visualization in rviz**
    1. Sensors
    2. Stereo point cloud
    3. PointHead
    4. RGB-D camera
5. Motions
6. Perception
7. Autonomous navigation
8. Remote lab demo

Sensors

Sensors visualization

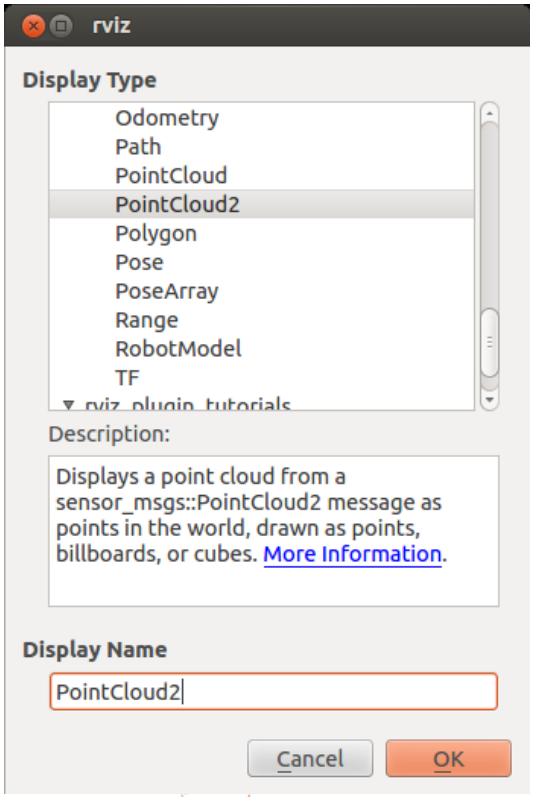
```
rosrun rviz rviz -d `rospack find reem_gazebo`/config/reem_sensors.vcg
```

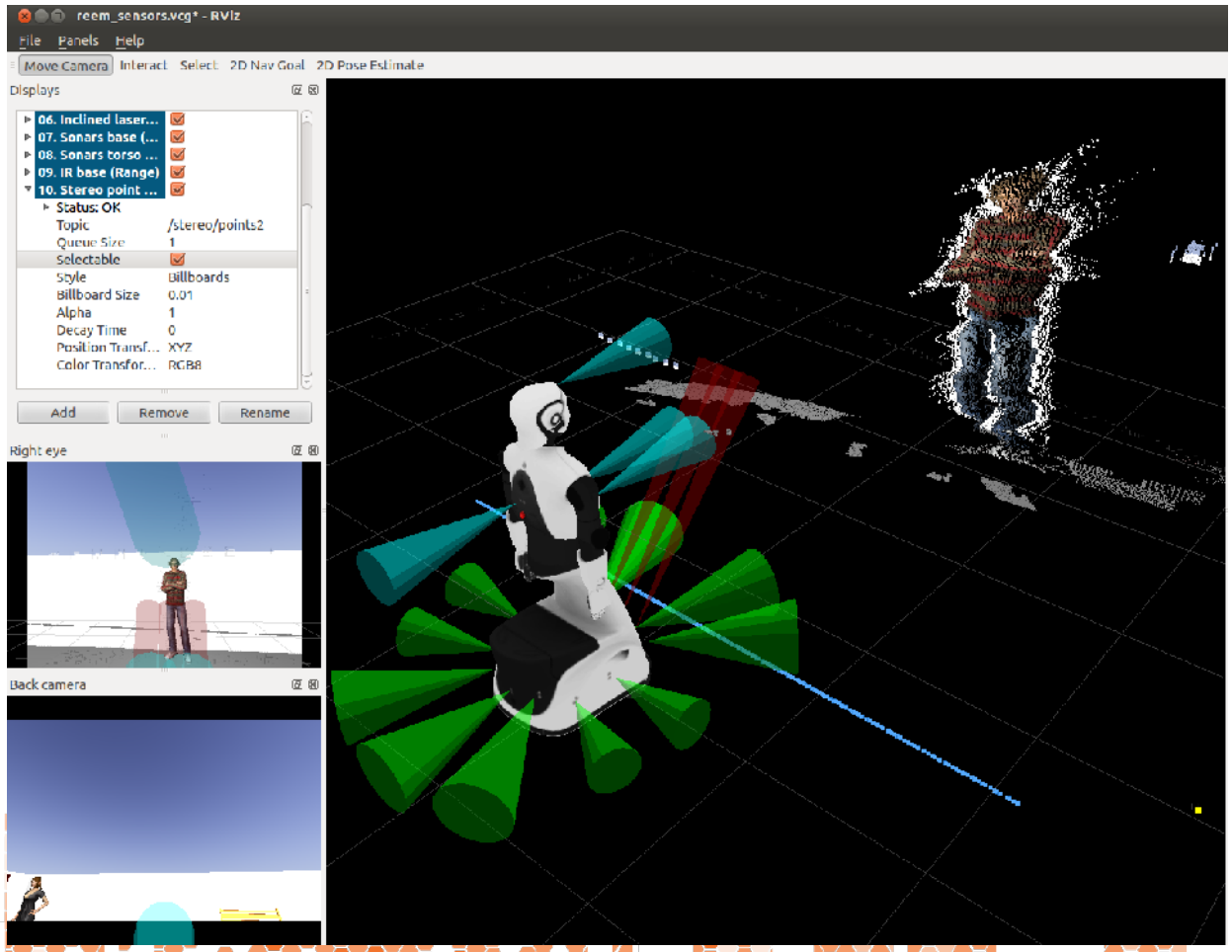


Stereo point cloud

Point cloud visualization

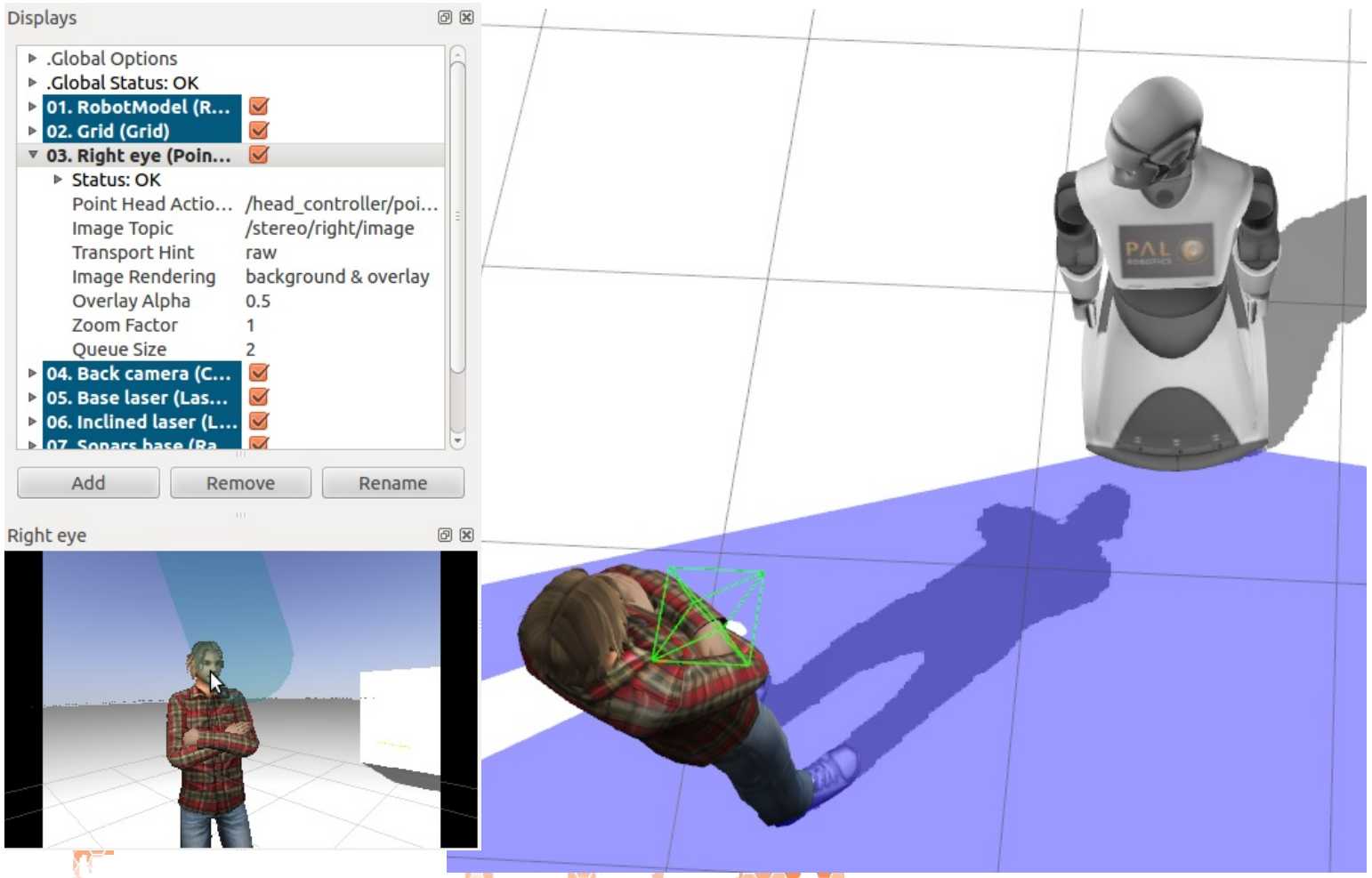
```
ROS_NAMESPACE=/stereo rosrun stereo_image_proc stereo_image_proc
/stereo/right/image_raw:=/stereo/right/image /stereo/left/image_raw:=/stereo/left/image
_approximate_sync:=True
```





PointHead

move REEM's head by clicking on image

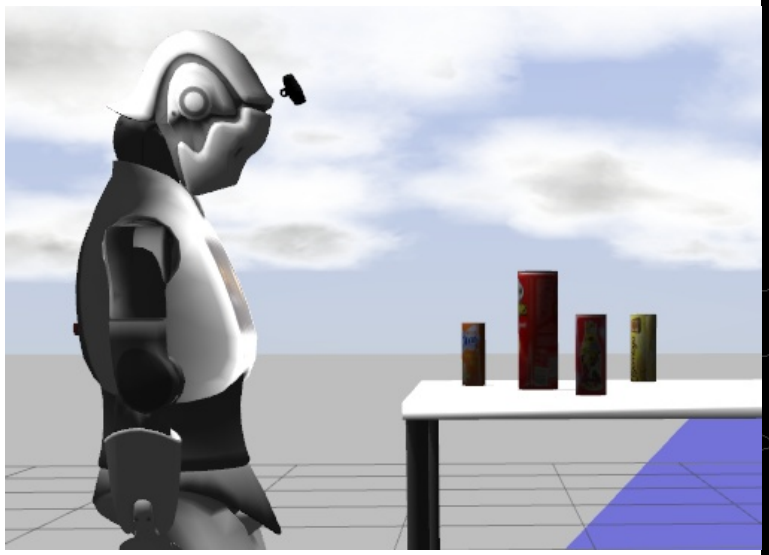


RGB-D camera

Launch REEM with a RGB-D sensor on top of its head

```
USE_RGBD_SENSOR=true roslaunch reem_gazebo reem_gazebo.launch world:=objects_on_table
```

```
rosrun rviz rviz -d `rospack find reem_gazebo`/config/reem_asus_xtion.vcg
```



Overview

1. Introduction
2. Installation
3. REEM in Gazebo
4. Visualization in rviz
5. **Motions**
    1. Inspect kinematic workspace
    2. Moving the wheels
    3. Play library motions
6. Perception
7. Autonomous navigation
8. Remote lab demo

Inspect kinematic workspace

roslaunch reem_tutorials reem_kinematic_sim.launch

Joint State Publisher

wheel_right_joint	0.00
wheel_left_joint	0.00
caster_right_1_joint	0.00
caster_right_2_joint	0.00
caster_left_1_joint	0.02
caster_left_2_joint	3.14
torso_1_joint	0.31
torso_2_joint	0.67
head_1_joint	0.49
head_2_joint	-0.26
arm_right_1_joint	1.27
arm_right_2_joint	0.72
arm_right_3_joint	0.88
arm_right_4_joint	0.54
arm_right_5_joint	-0.53
arm_right_6_joint	0.00
arm_right_7_joint	0.00
arm_left_1_joint	2.61
arm_left_2_joint	0.39
arm_left_3_joint	-0.56
arm_left_4_joint	2.27
arm_left_5_joint	0.00
arm_left_6_joint	0.00
arm_left_7_joint	0.00
hand_right_thumb_joint	0.75
hand_right_index_joint	2.97

Moving the wheels

Keyboard

```
roslaunch turtlebot_teleop keyboard_teleop.launch
```

```
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
anything else : stop

CTRL-C to quit

currently:      speed 0.3       turn 1
```

Joystick

```
roslaunch reem_bringup joystick_teleop.launch cmd_vel:=/cmd_vel
```

Play library motions

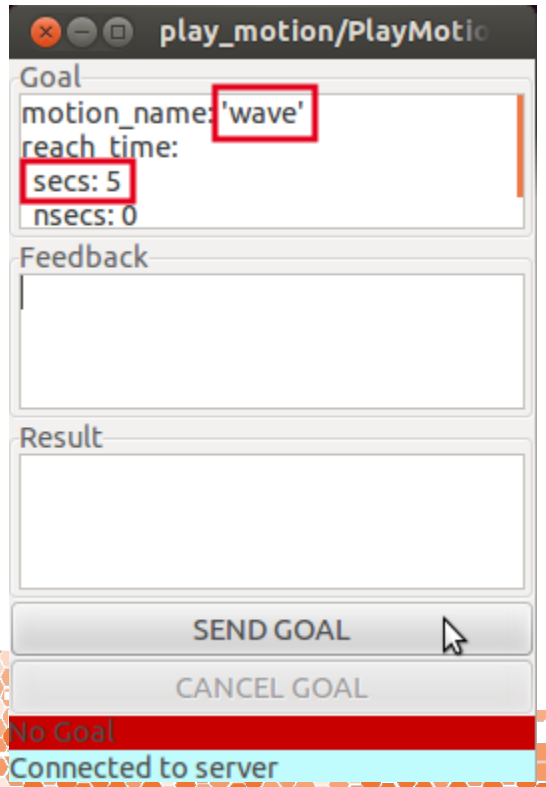
Launch the action server

```
roslaunch reem_bringup play_motion.launch
```

The motions defined in **reem_bringup/config/reem_motions.yaml** are loaded:
bow, car_drive_ferrari, open_arms, wave, etc.

Launch an action client interface

```
rosrun actionlib axclient.py /play_motion
```



REEM in Gazebo

1. Introduction
2. Installation
3. REEM in Gazebo
4. Visualization in rviz
5. Motions
6. **Perception**
    1. Image undistortion
    2. Image rectification
    3. Person detection
    4. Face detection
    5. Segmentation
    6. Model-basec object tracking
    7. Visual servo
7. Autonomous navigation
8. Remote lab demo

Image undistortion

```
rosrun image_view image_view image:=/stereo/left/image
```

```
ROS_NAMESPACE=/stereo/left rosrun image_proc image_proc image_raw:=image
```

```
rosrun image_view image_view image:=/stereo/left/image_rect_color
```

IMPORTANT: Gazebo does not simulate distortion

Image rectification

```
roscd reem_tutorials/etc
rosbag play reem_stereo --loop
```

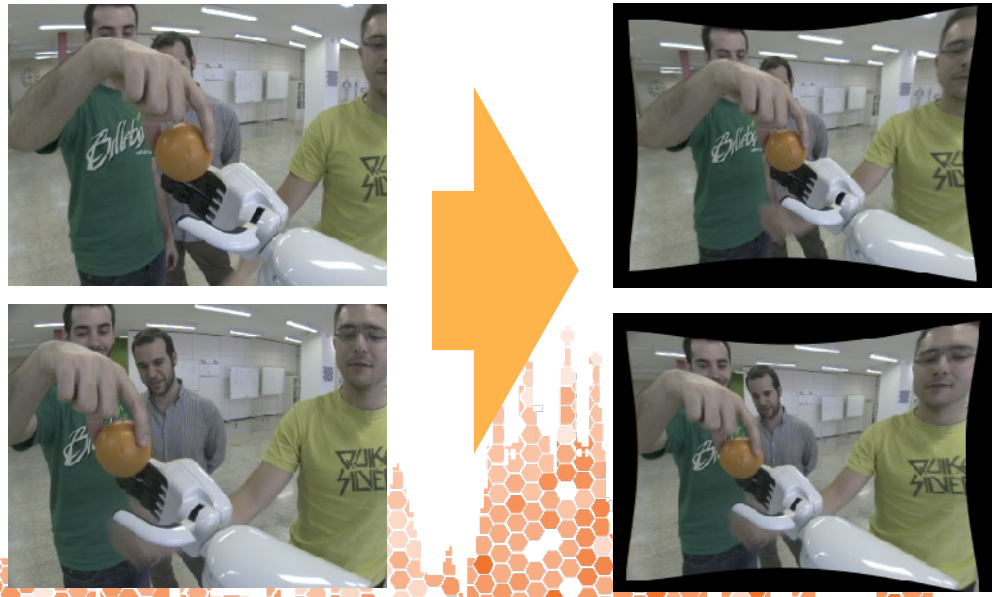
```
rosrun image_view image_view image:=/stereo/left/image
```

```
rosrun image_view image_view image:=/stereo/right/image
```

```
ROS_NAMESPACE=/stereo rosrun stereo_image_proc stereo_image_proc
/stereo/left/image_raw:=/stereo/left/image /stereo/right/image_raw:=/stereo/right/image
```

```
rosrun image_view image_view image:=/stereo/left/image_rect_color
```

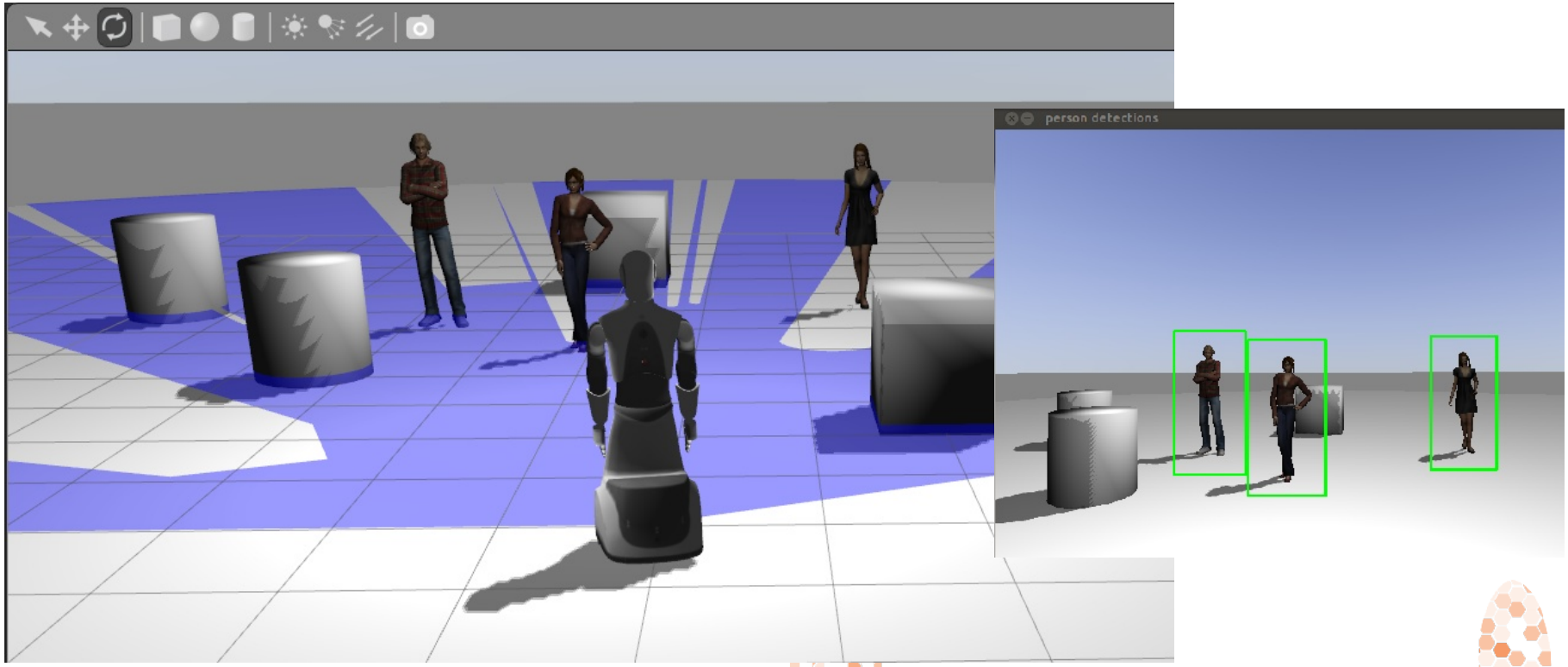
```
rosrun image_view image_view image:=/stereo/right/image_rect_color
```



Person detection

Fullbody standing person detection using OpenCV

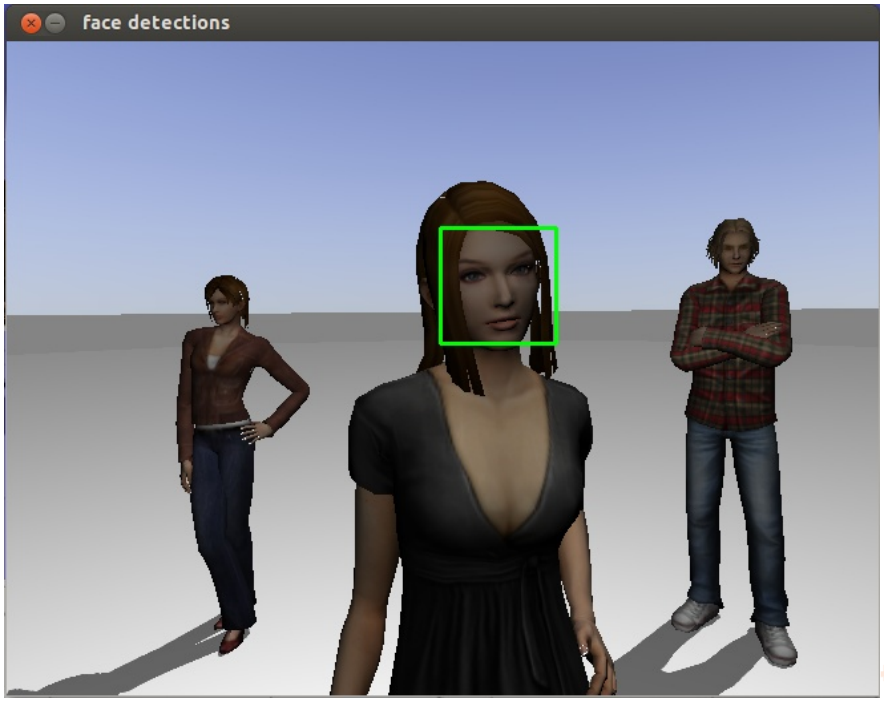
```
roslaunch person_detector_opencv person_detector.launch
```



Face detection

Face detection using OpenCV

```
roslaunch face_detector_opencv face_detector.launch
```



Histogram segmentation

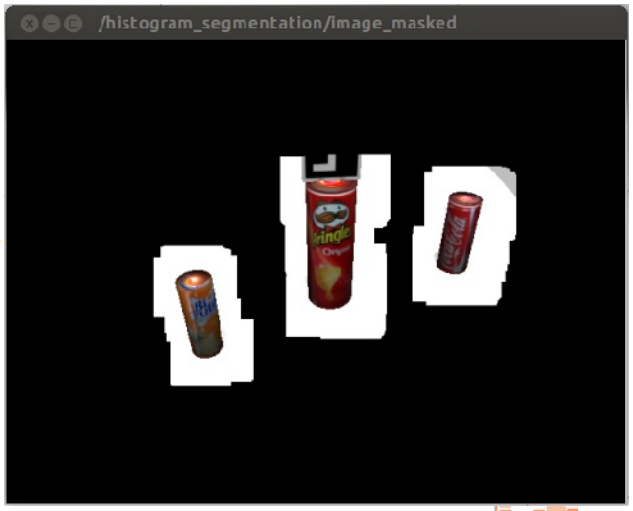
Hue-Saturation histogram segmentation

```
roslaunch reem_gazebo reem_gazebo.launch world:=objects_on_table
```

```
roslaunch pal_vision_segmentation histogram_pringles_segment.launch
```

```
rosrun image_view image_view image:=/histogram_segmentation/image_masked
```

Histogram back-projection + morphological operations



Disparity segmentation

Segmentation based on disparity range

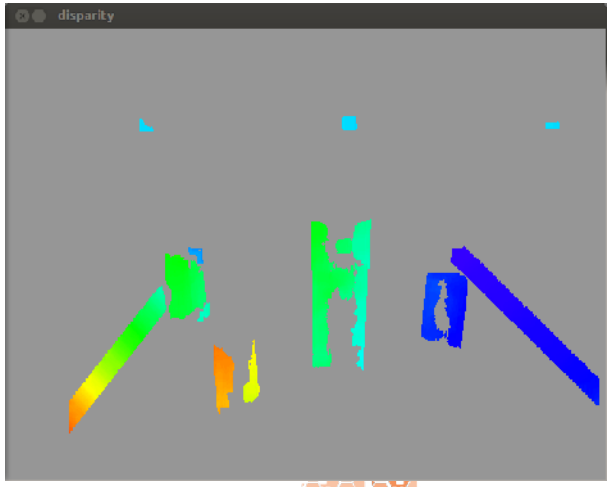
```
roslaunch reem_gazebo reem_gazebo.launch world:=objects_on_table
```

```
ROS_NAMESPACE=/stereo rosrun stereo_image_proc stereo_image_proc
/stereo/right/image_raw:=/stereo/right/image /stereo/left/image_raw:=/stereo/left/image
_approximate_sync:=True
```

```
roslaunch pal_vision_segmentation disparity_segment.launch
```

```
rosrun image_view image_view image:=/disparity_segmentation/image_masked
```

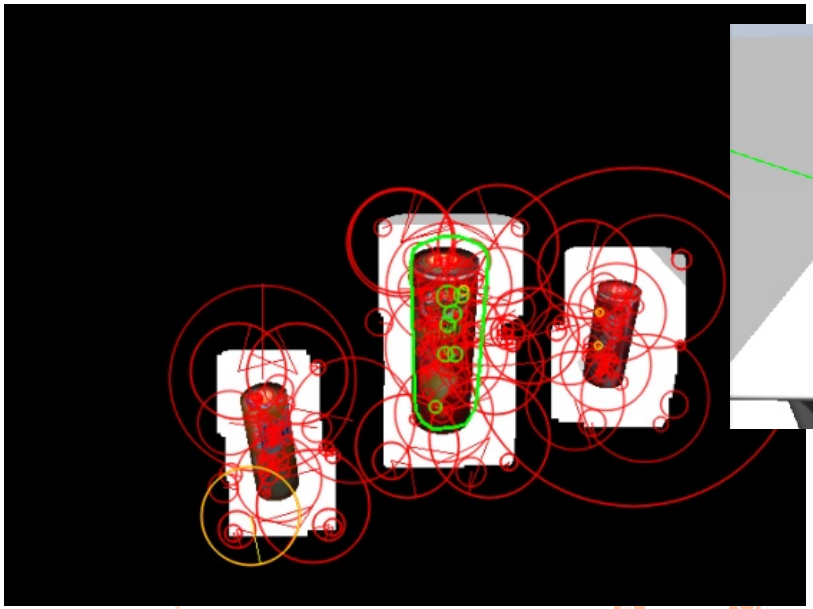
```
rosrun dynamic_reconfigure reconfigure_gui /disparity_segmentation
```

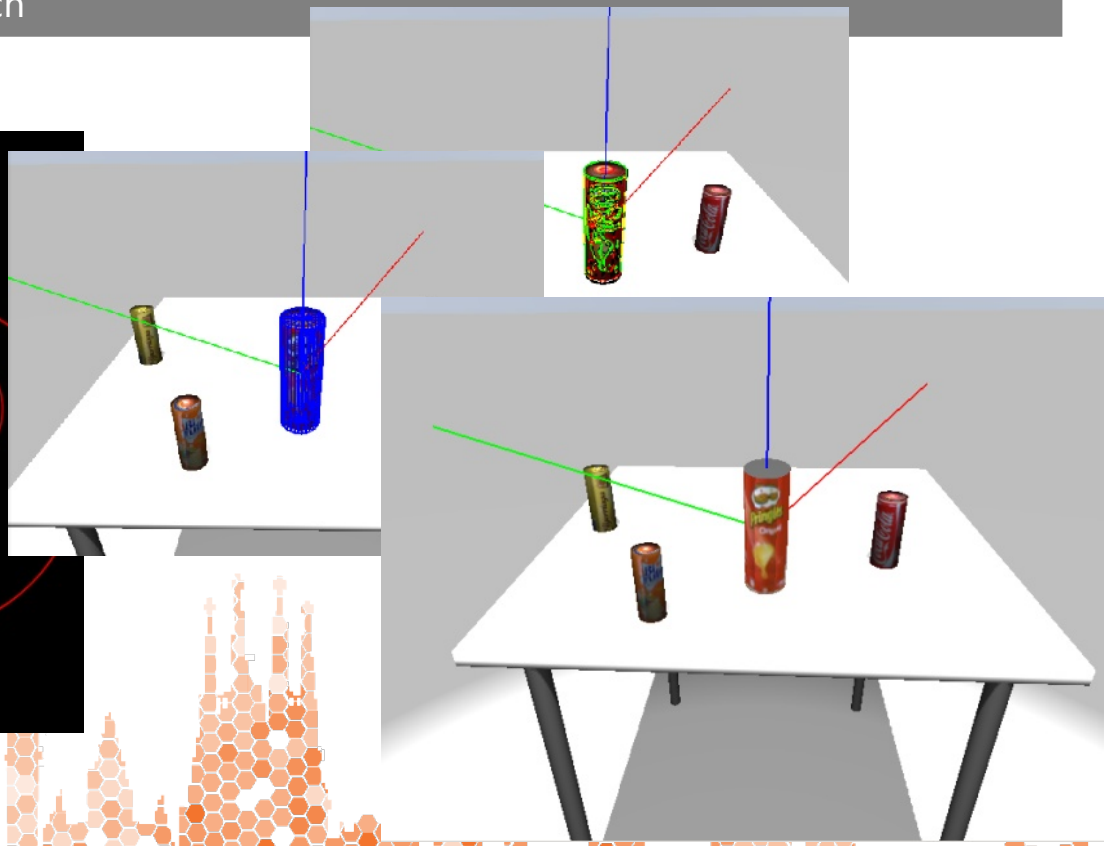


Model based object tracking

BLORT: The Blocks World Robotic Vision Toolbox

```
roslaunch reem_gazebo reem_gazebo.launch world:=objects_on_table
roslaunch pal_vision_segmentation histogram_pringles_segment.launch
roslaunch blort_ros tracking_histogram.launch
roslaunch blort_ros view_control.launch
```





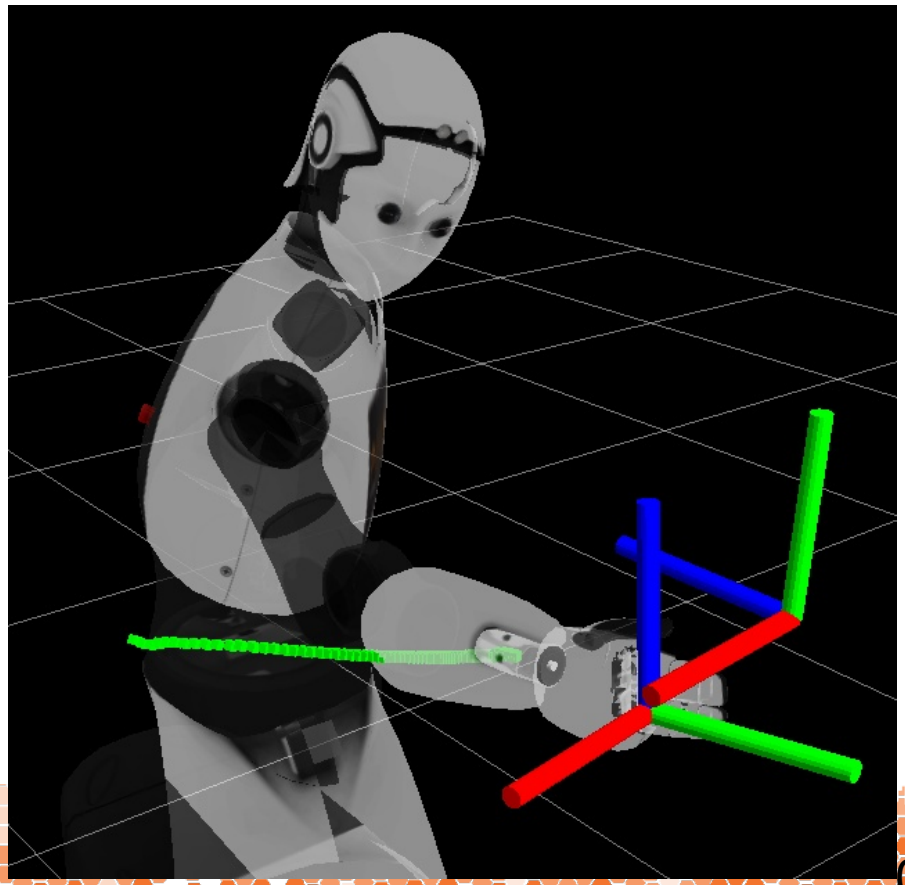
Visual servo

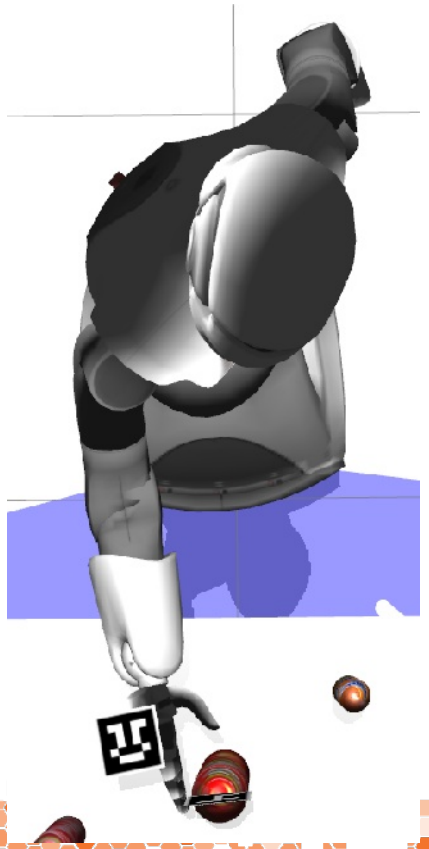
roslaunch reem_upperbody_visual_servo simulation.launch

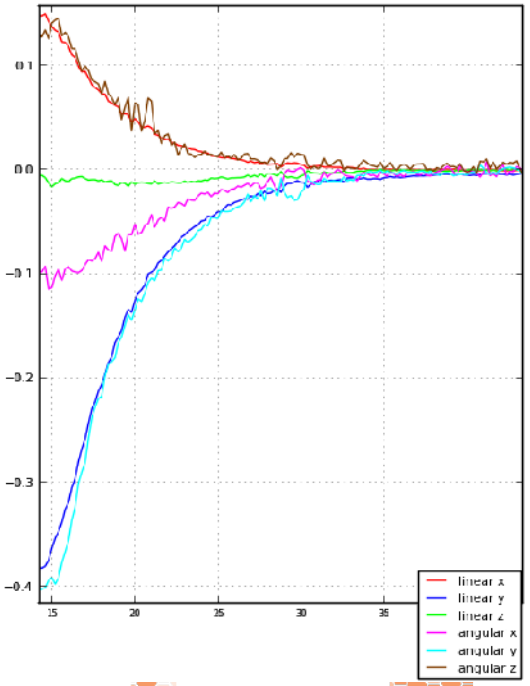
roscd reem_upperbody_visual_servo/scripts/simulation
./start_perception.sh

roslaunch reem_upperbody_visual_servo rviz.launch

roslaunch reem_upperbody_visual_servo reem_upperbody_visual_servo.launch







Overview

1. Introduction
2. Installation
3. REEM in Gazebo
4. Visualization in rviz
5. Motions
6. Perception
7. **Autonomous navigation**
    1. Mapping
    2. Choosing a map
    3. Autonomous navigation
8. Remote lab demo

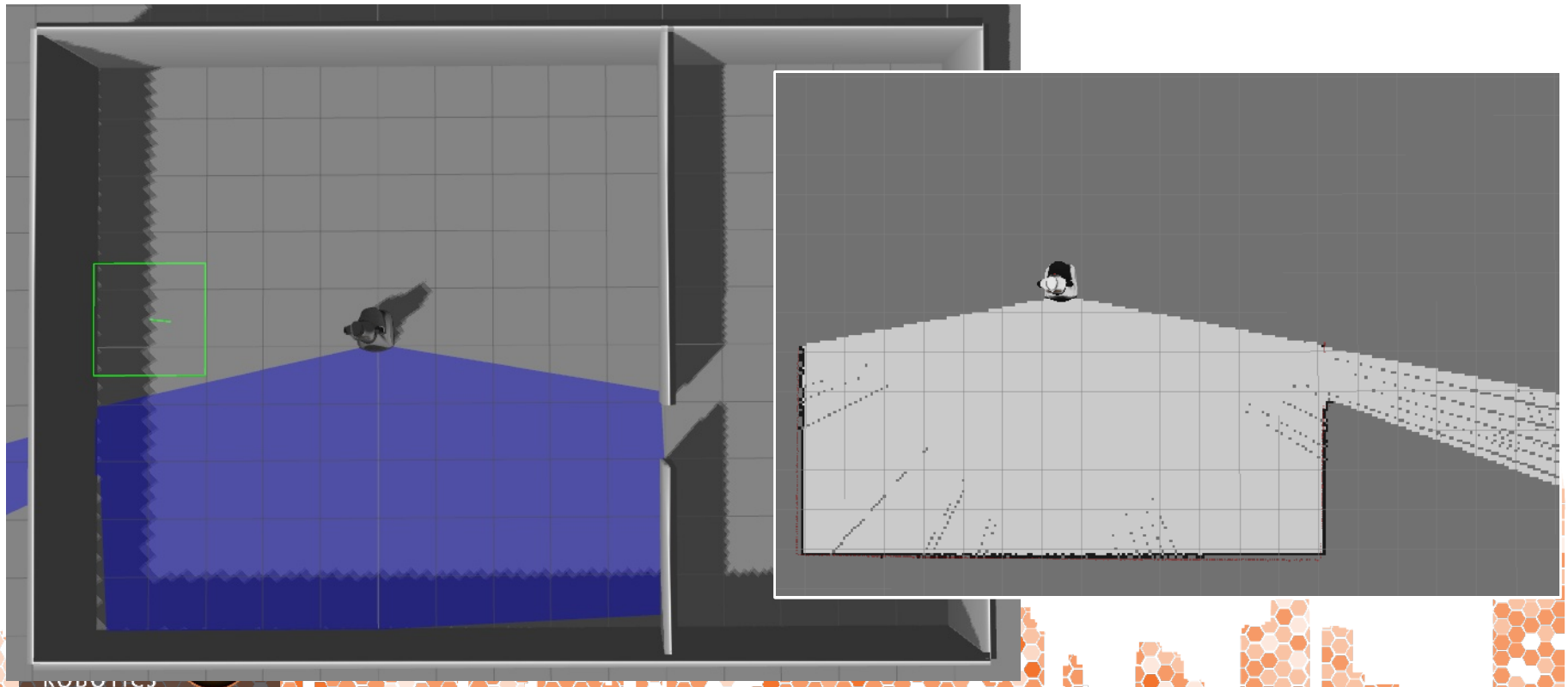
Mapping

Launch mapping

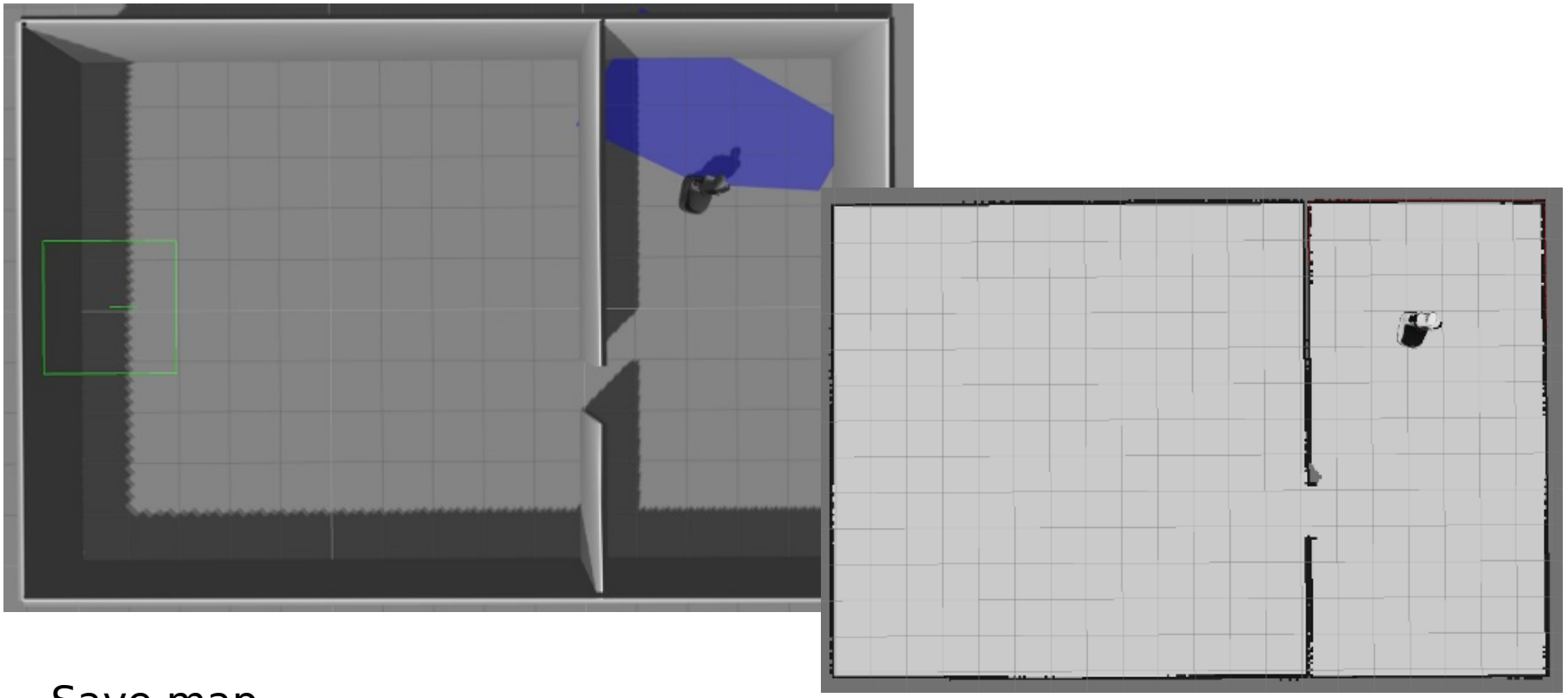
```
roslaunch reem_tutorials mapping.launch world:=simple_office
```

Keyboard teleoperation

```
roslaunch turtlebot_teleop keyboard_teleop.launch
```



Mapping



Save map

```
roscd reem_gazebo_maps/config/maps
```

```
rosrun map_server map_saver -f simple_office
```

Choosing the map

Create a soft link to the map to be used

```
roscd reem_gazebo_maps/config/maps
```

```
unlink map.yaml
```

```
ln -s simple_office.yaml map.yaml
```

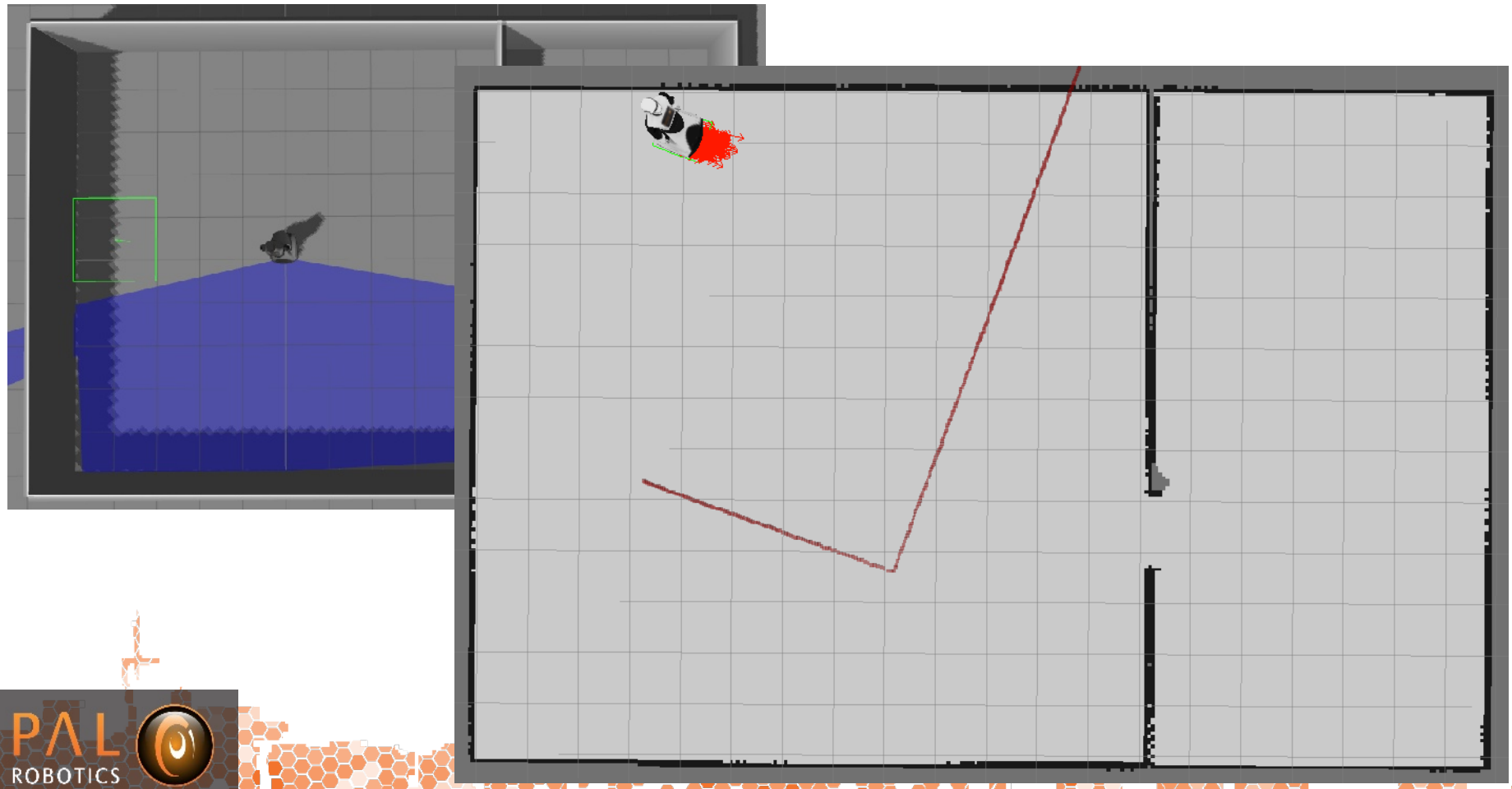
```
-rw-rw-r-- 1 jordi users 60K Nov 12 15:11 default.pgm
-rw-rw-r-- 1 jordi users 129 Nov 12 15:11 default.yaml
lrwxrwxrwx 1 jordi users  18 Nov 13 09:30 map.yaml -> simple_office.yaml
-rw-rw-r-- 1 jordi users 16M Nov 12 17:06 simple_office.pgm
-rw-rw-r-- 1 jordi users 143 Nov 12 17:06 simple_office.yaml
-rw-rw-r-- 1 jordi users 16M Nov 13 08:40 small_office.pgm
-rw-rw-r-- 1 jordi users 142 Nov 13 08:40 small_office.yaml
jordi@france:~/pal-isolated/stacks/reem_gazebo_navigation/reem_gazebo_maps/config/maps$
```

Autonomous navigation

Launch gazebo + rviz

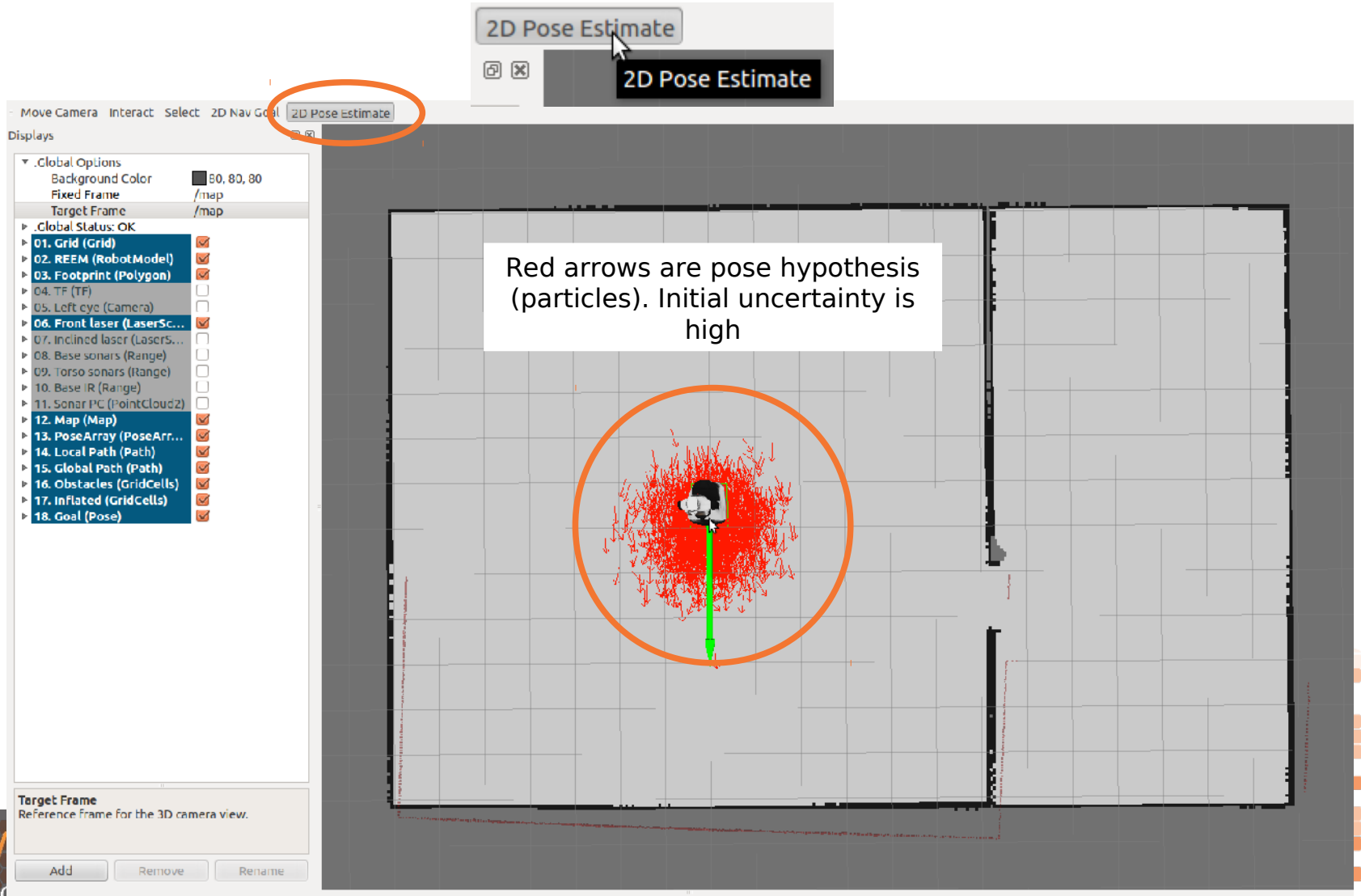
```
roslaunch reem_tutorials autonomous_2dnav.launch
```

The initial position of REEM in the map may be wrong → **wake-up robot problem**



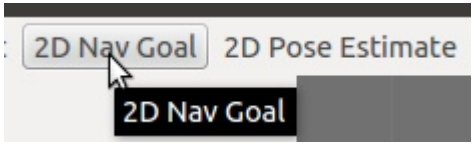
Autonomous navigation

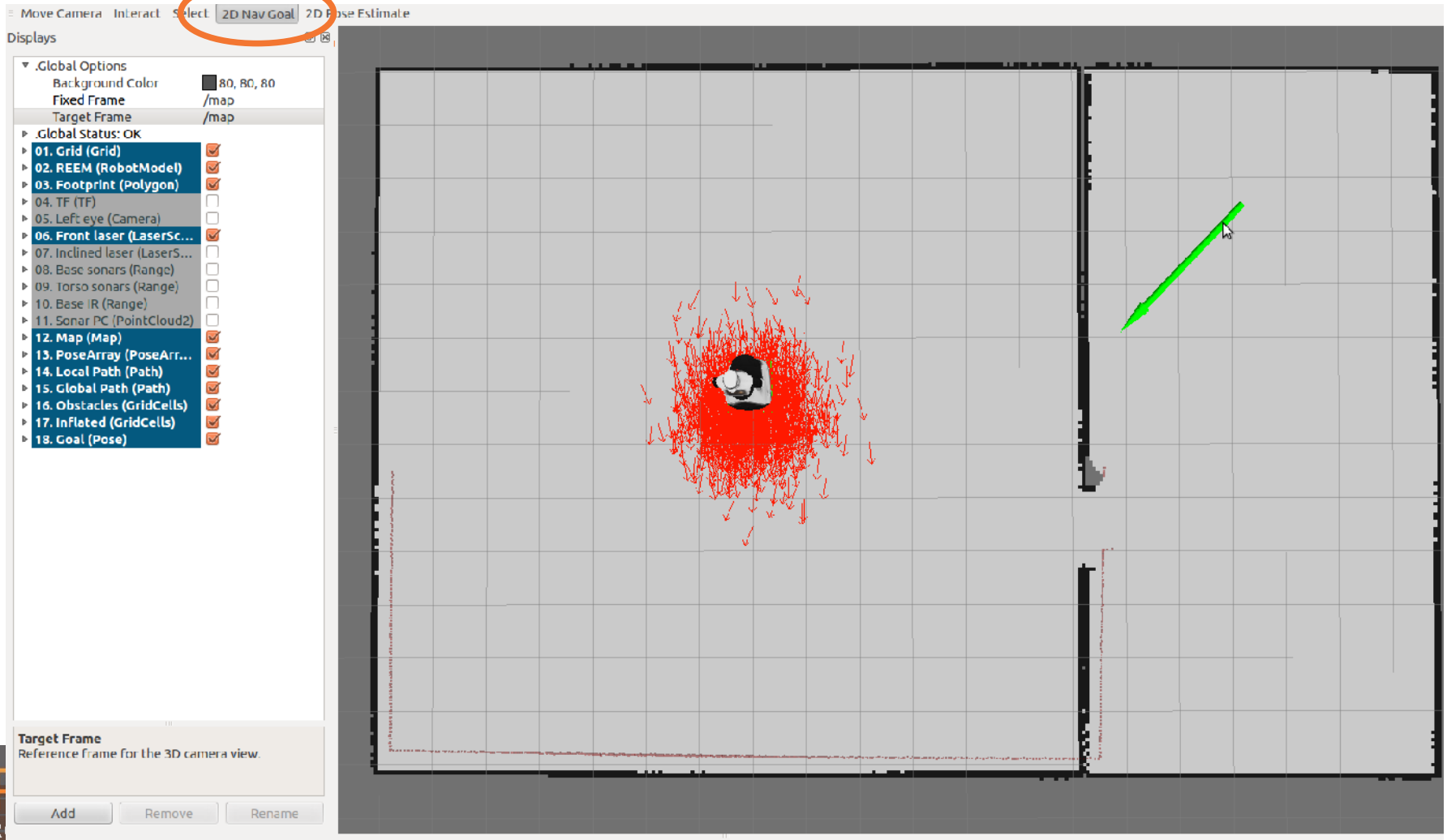
Localize the robot: provide an estimate with rviz



Autonomous navigation

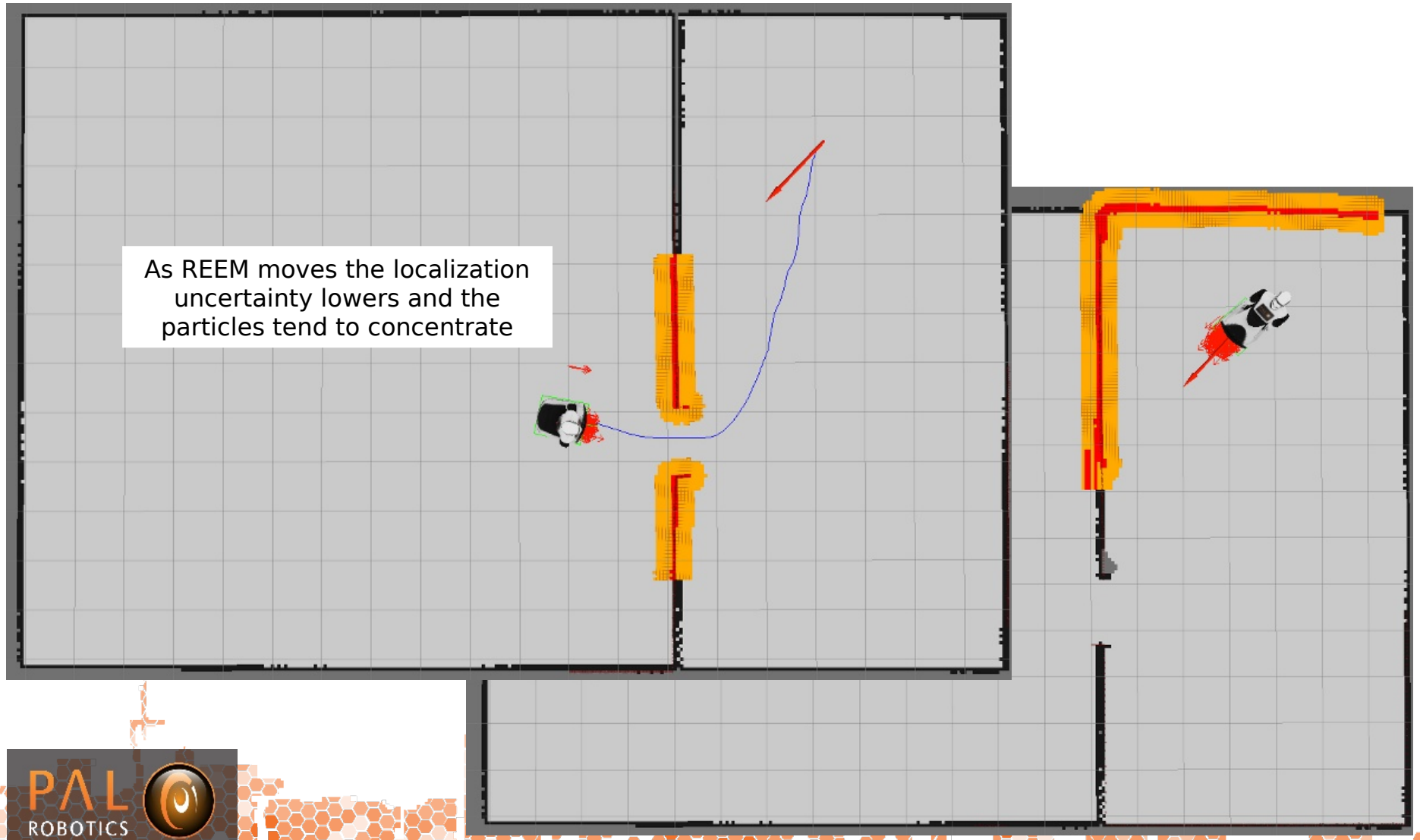
Sending REEM to a goal pose





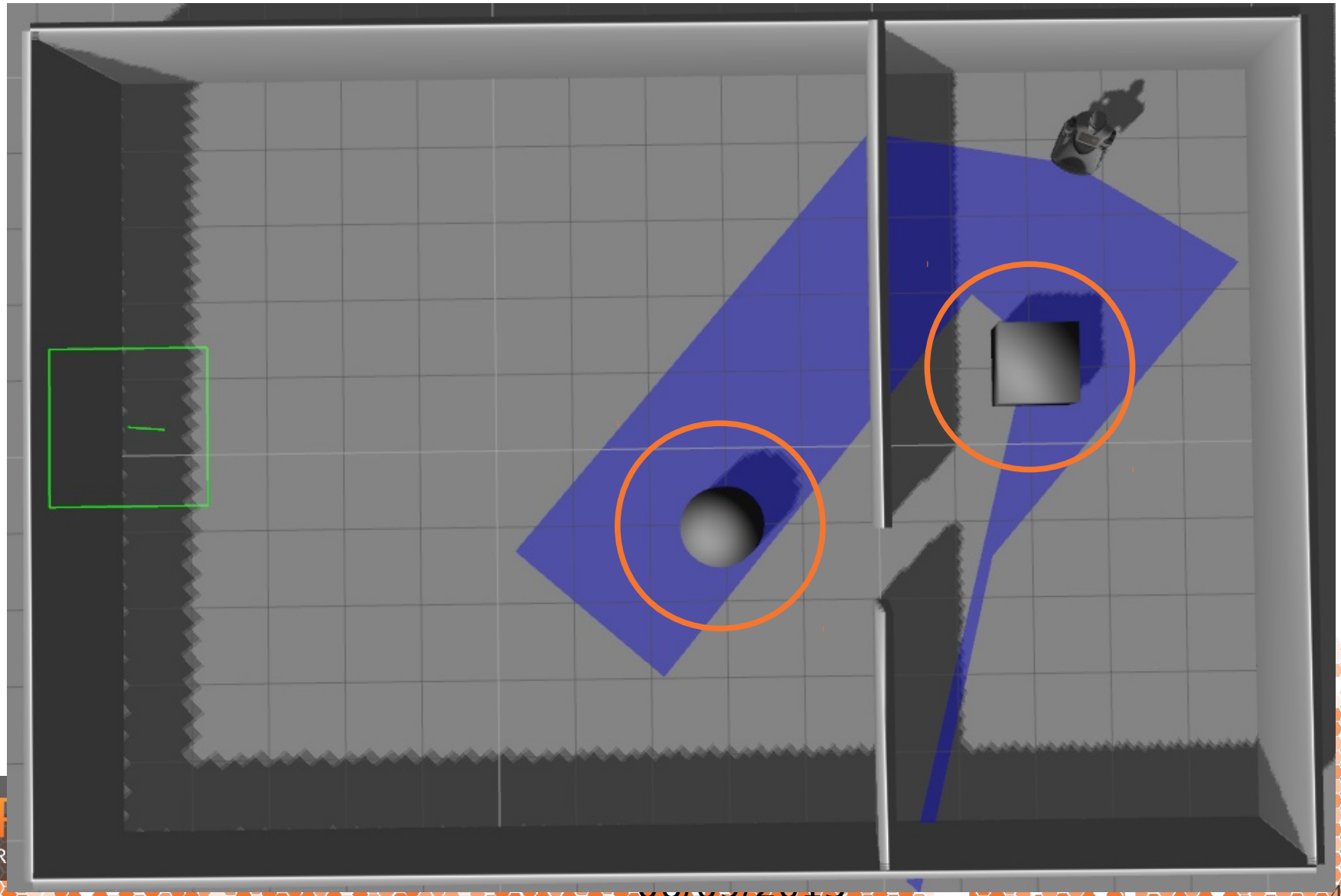
Autonomous navigation

Reaching the goal



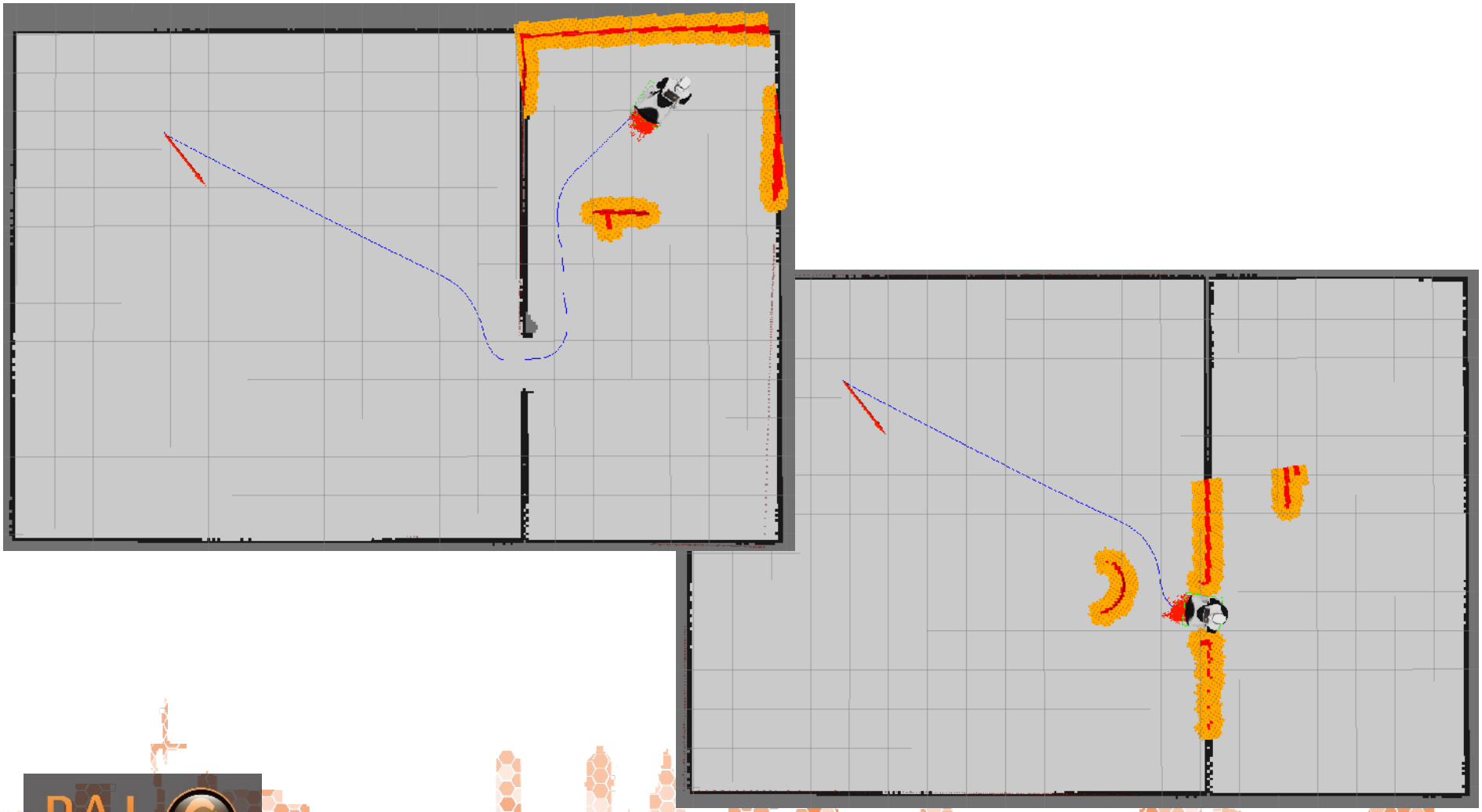
Autonomous navigation

Adding obstacles not represented in the map



Autonomous navigation

Navigating avoiding obstacles



Exercises

Hands-on session

Exercises

1. Modify *person_detector_opencv/src/person_detector.cpp* so that the node publishes the detections on a topic (you may use geometry_msgs::Polygon)

2. Create a C++ node to send navigation goals to /reem_sim/move_base_simple/goal (and/or to /reem_sim/move_base using the actionlib interface if time allows)

3. Check the documentation of the Monte Carlo Localization algorithm (http://wiki.ros.org/amcl) in order to find how to spread particles all over the map. Then move REEM with the keyboard (ROS_NAMESPACE=reem_sim roslaunch turtlebot_teleop keyboard_teleop.launch) and after a short time a good localization hypothesis should come up.

4. Launch REEM in an empty world and create a new world of your own. Then save it and use it to perform a map and test localization and planning on it.

5. Set up a stereo odomer for REEM using http://wiki.ros.org/viso2_ros or http://wiki.ros.org/fovis

Thank you for your attention!

PAL ROBOTICS S.L.

Pujades 77-79, 4º 4ª
08005 Barcelona, Spain

T. +34 934 145 347
F. +34 932 091 109

info@pal-robotics.com
www.pal-robotics.com