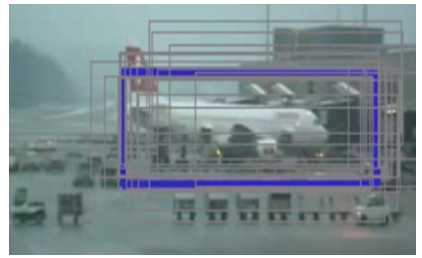
UNIVERSITÀ di VERONA

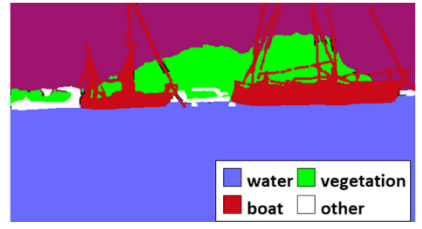
Dipartimento di INFORMATICA

Corso di Laboratorio Ciberfisico
Modulo di Robot Programming with ROS









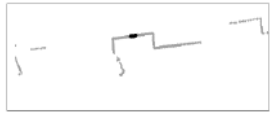
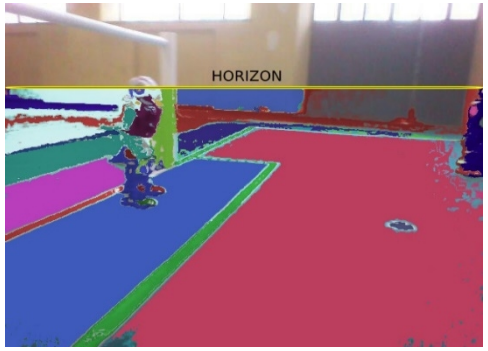
actionlib

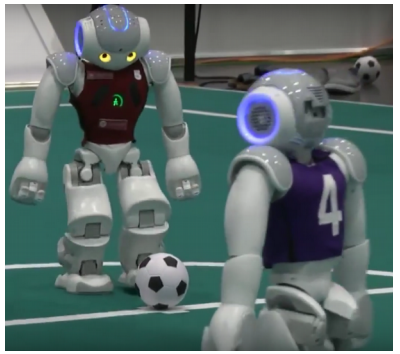
ROS

Docente:

Domenico Daniele Bloisi





Giugno 2018

References and Credits

Questo materiale deriva da:

Luca Iocchi – Sapienza Università di Roma
Actions and Plans

https://www.dis.uniroma1.it/~nardi/Didattica/CAI/matdid/1-ROS-ActionLib.pdf

ROS actionlib

- Node A sends a request to node B to perform some task
- Services are suitable if task is "instantaneous"
- Actions are more adequate when task takes time and we want to monitor, have continuous feedback and possibly cancel the request during execution

actionlib package

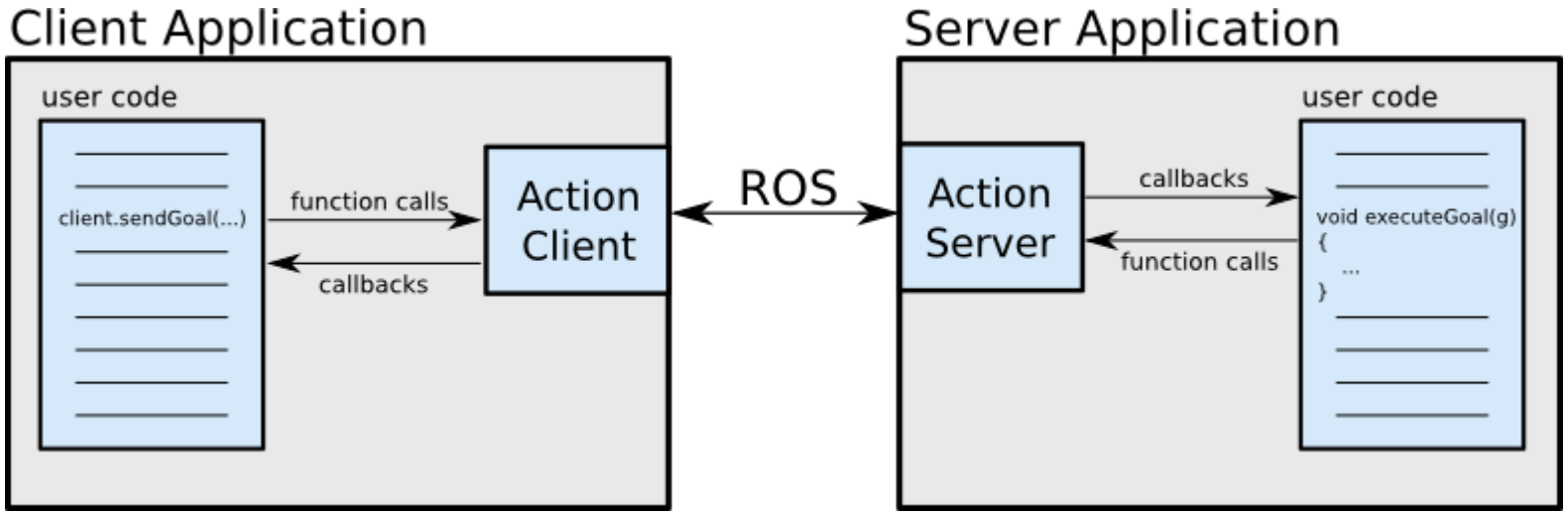
actionlib package provides tools to

- create servers that execute long-running tasks (that can be preempted)
- create clients that interact with servers

References

- http://wiki.ros.org/actionlib
- http://wiki.ros.org/actionlib/DetailedDescription
- http://wiki.ros.org/actionlib/Tutorials

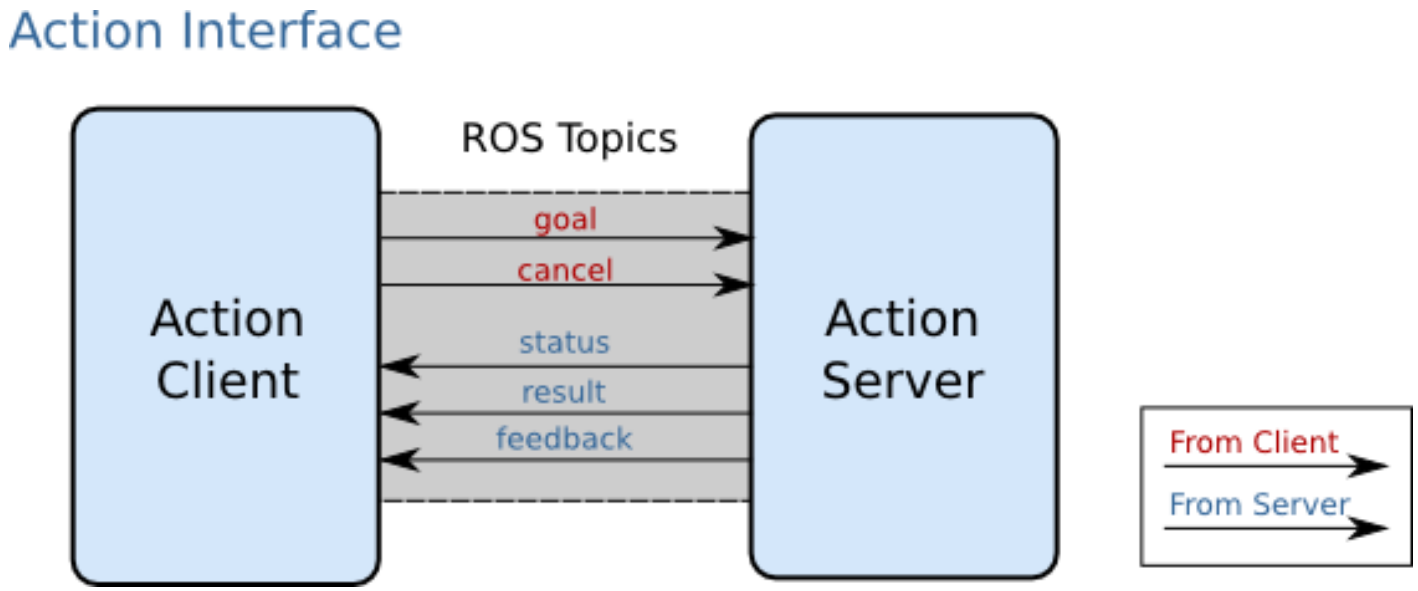
Client-Server interaction



- The ActionClient and ActionServer communicate via a "ROS Action Protocol", which is built on top of ROS messages

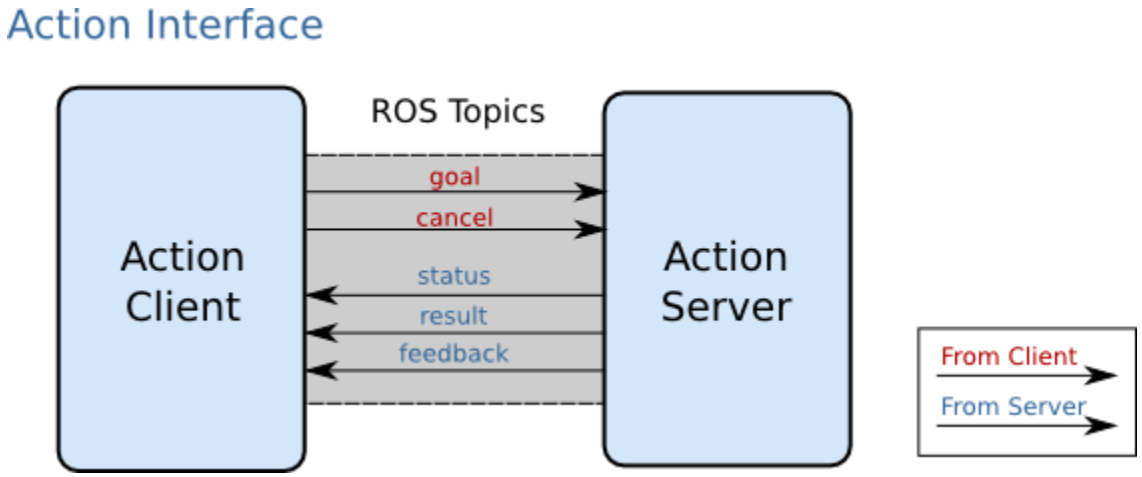
- The client and server then provide a simple API for users to request goals (on the client side) or to execute goals (on the server side) via function calls and callbacks

Action Interface & Transport Layer



- The action client and server communicate with each other using a predefined action protocol
- This action protocol relies on ROS topics in a specified ROS namespace in order to transport messages

ROS Messages



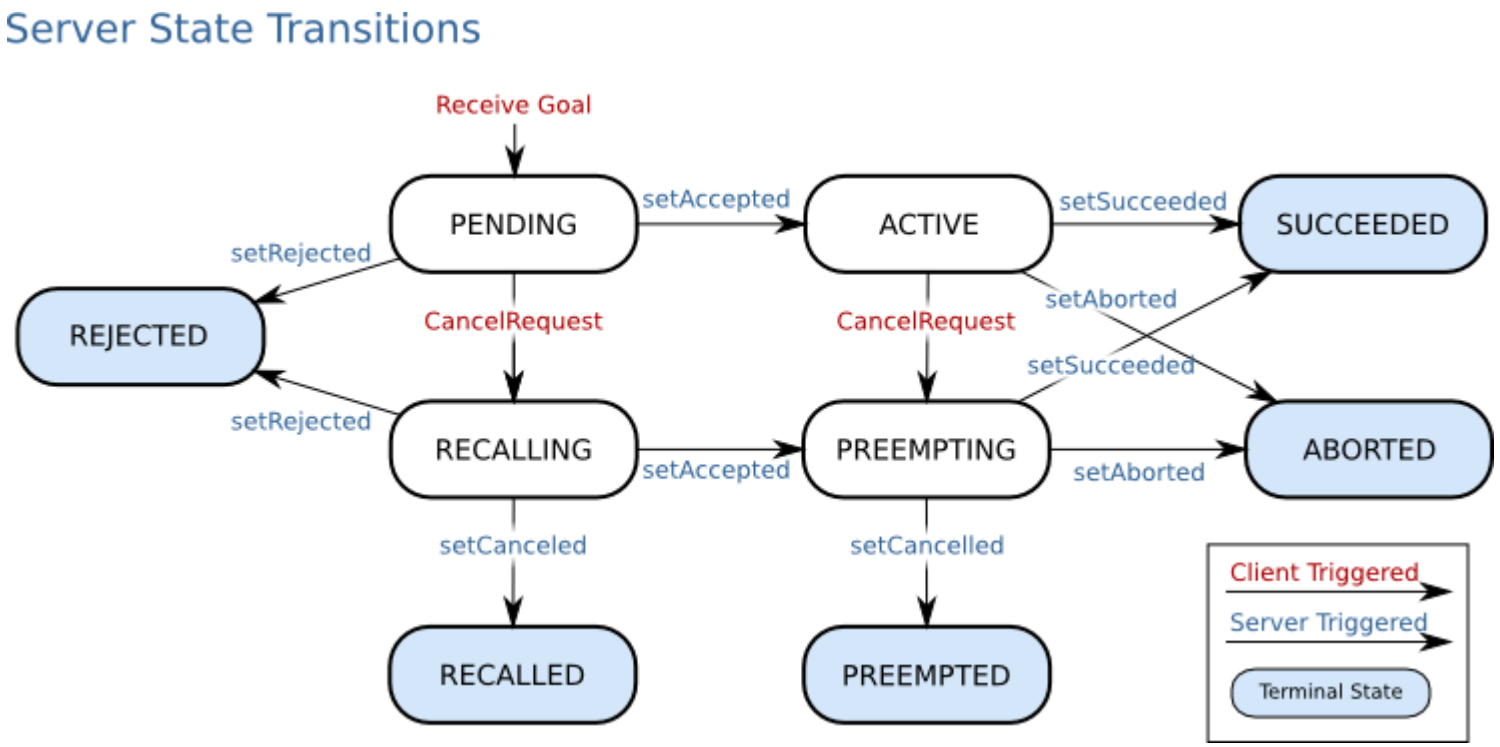
- goal - Used to send new goals to servers
- cancel - Used to send cancel requests to servers
- status - Used to notify clients on the current state of every goal in the system
- feedback - Used to send clients periodic auxiliary information for a goal
- result - Used to send clients one-time auxiliary information upon completion of a goal

Actions and Goal ID

- Action templates are defined by a name and some additional properties through an .action structure defined in ROS
- Each instance of an action has a unique Goal ID
- Goal ID provides the action server and the action client with a robust way to monitor the execution of a particular instance of an action

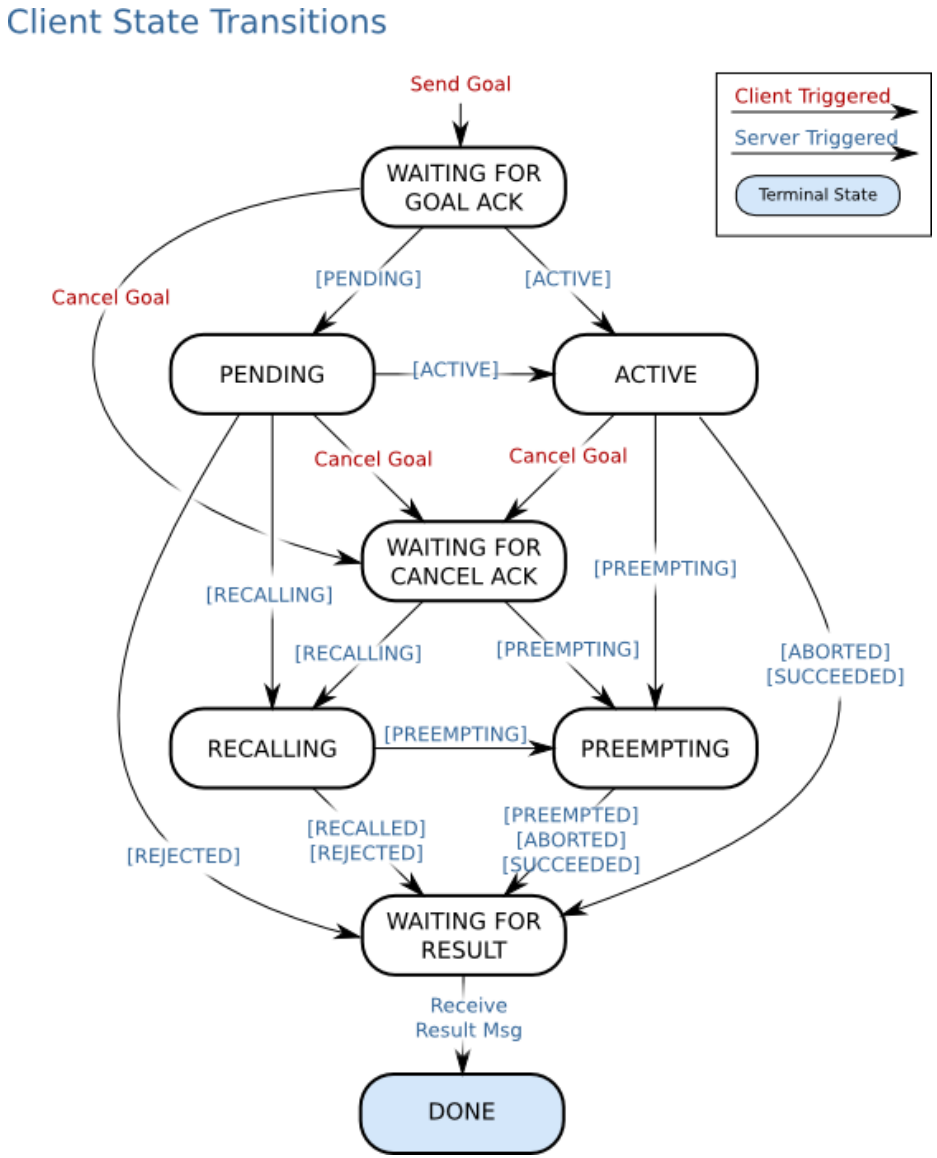
Server State Machine

- Goals are initiated by an ActionClient
- Once a goal is received by an ActionServer, the ActionServer creates a state machine to track the status of the goal



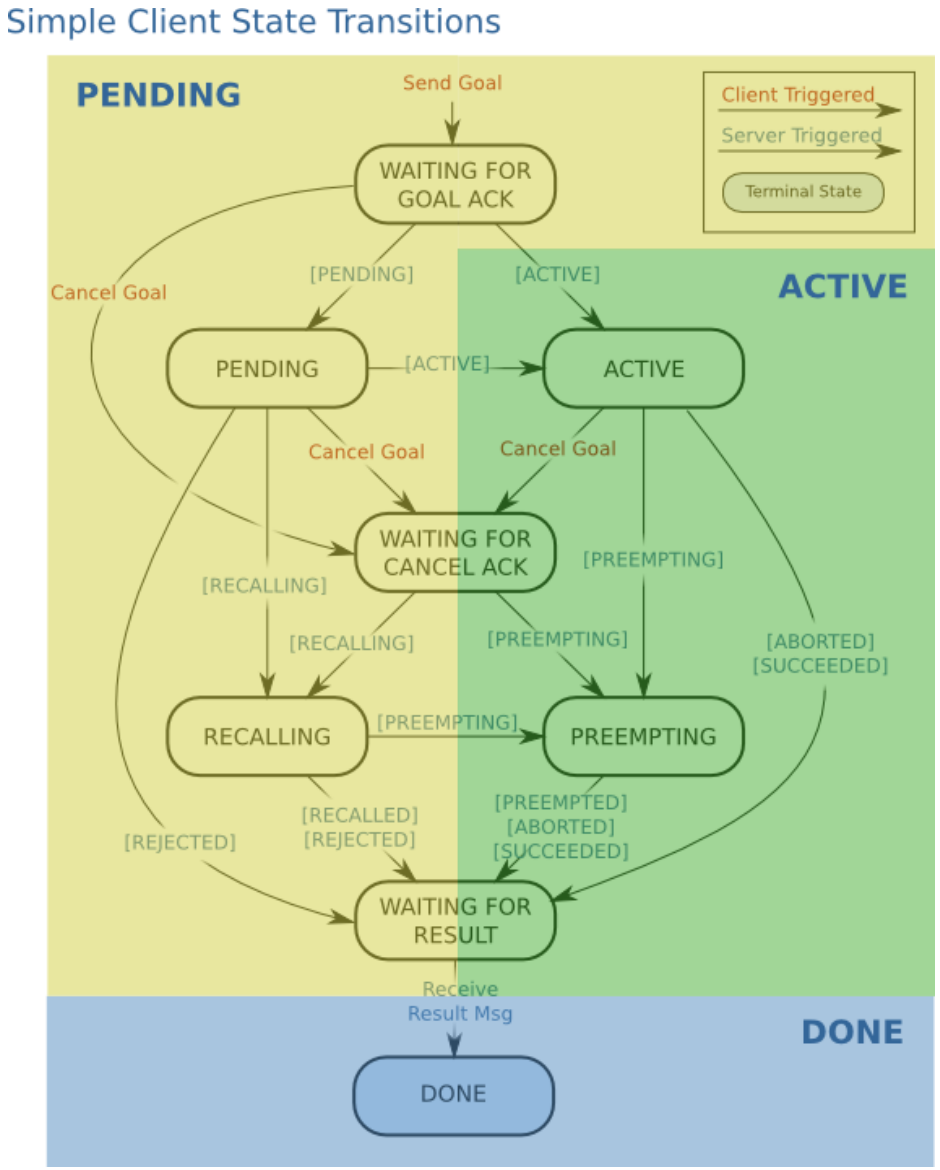
Client State Machine

- in actionlib, the server state machine is the primary machine
- the client state machine is the secondary/coupled state machine that tries to track the server's state



SimpleActionServer/Client

- SimpleActionServer: implements a single goal policy
- Only one goal can have an active status at a time
- New goals preempt previous goals based on the stamp in their GoalID field
- SimpleActionClient: implements a simplified ActionClient



Example: move_base action server

- **Action Subscribed Topics**
  - move_base/goal (move_base_msgs/MoveBaseActionGoal): A goal for move_base to pursue in the world
  - move_base/cancel (actionlib_msgs/GoalID): A request to cancel a specific goal
- **Action Published Topics**
  - move_base/feedback (move_base_msgs/MoveBaseActionFeedback): Feedback contains the current position of the base in the world
  - move_base/status (actionlib_msgs/GoalStatusArray): Provides status information on the goals that are sent to the move_base action
  - move_base/result (move_base_msgs/MoveBaseActionResult): Result is empty for the move_base action

Sending a goal with move_base

```
typedef actionlib::SimpleActionClient<move_base_msgs::MoveBaseAction>
    MoveBaseClient;

//tell the action client that we want to spin a thread by default
MoveBaseClient ac("move_base", true);

//wait for the action server to come up
while(!ac.waitForServer(ros::Duration(5.0))){
    ROS_INFO("Waiting for the move_base action server to come up");
}

// setting the goal
move_base_msgs::MoveBaseGoal goal;
goal.target_pose.header.frame_id = "base_link";
goal.target_pose.header.stamp = ros::Time::now();
goal.target_pose.pose.position.x = 1.0;
goal.target_pose.pose.orientation.w = 1.0;
```

Sending a goal with move_base

```
// sending the goal
ROS_INFO("Sending goal");
ac.sendGoal(goal);

// wait until finish
ac.waitForResult();

// print result
if(ac.getState() == actionlib::SimpleClientGoalState::SUCCEEDED)
    ROS_INFO("Hooray, the base moved 1 meter forward");
else
    ROS_INFO("The base failed to move forward 1 meter for some reason");
```

Cancelling a goal with move_base

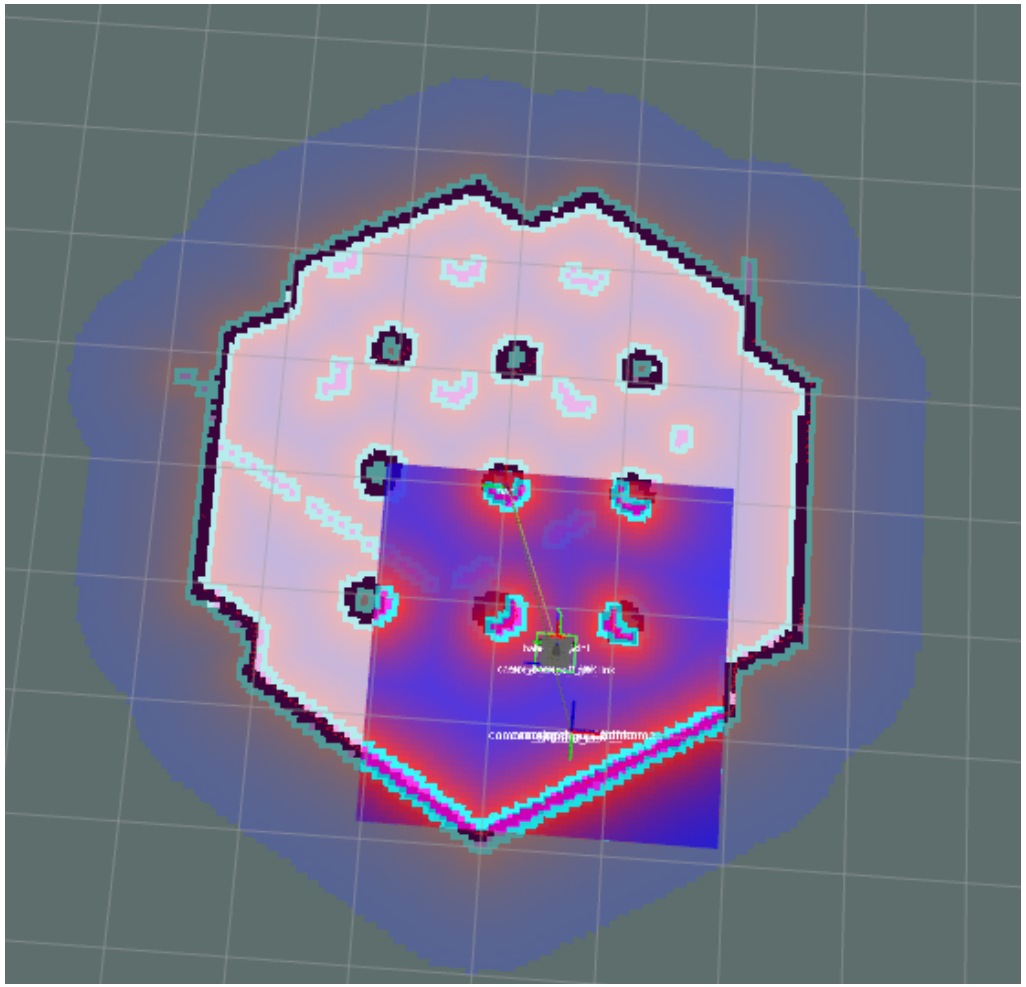
```
typedef
actionlib::SimpleActionClient<move_base_msgs::MoveBaseAction>
MoveBaseClient;

MoveBaseClient ac("move_base", true);

...

// Cancel all active goals
ac.cancelAllGoals();
```

ActionServer/Client

- ActionServer and ActionClient use the complete set of states and transitions
- More difficult to program
- Needed when it is necessary to execute multiple instances of an action at the same time (parallel actions)

Example Turtlebot3

- Use the SimpleActionClient to send a navigation goal to the Turtlebot3
- Tell the base to move 1 meter forward in the "base_link" coordinate frame



http://wiki.ros.org/navigation/Tutorials/SendingSimpleGoals

Package creation

```
$ catkin_create_pkg turtlebot3_navigation_goals
  move_base_msgs actionlib roscpp
```

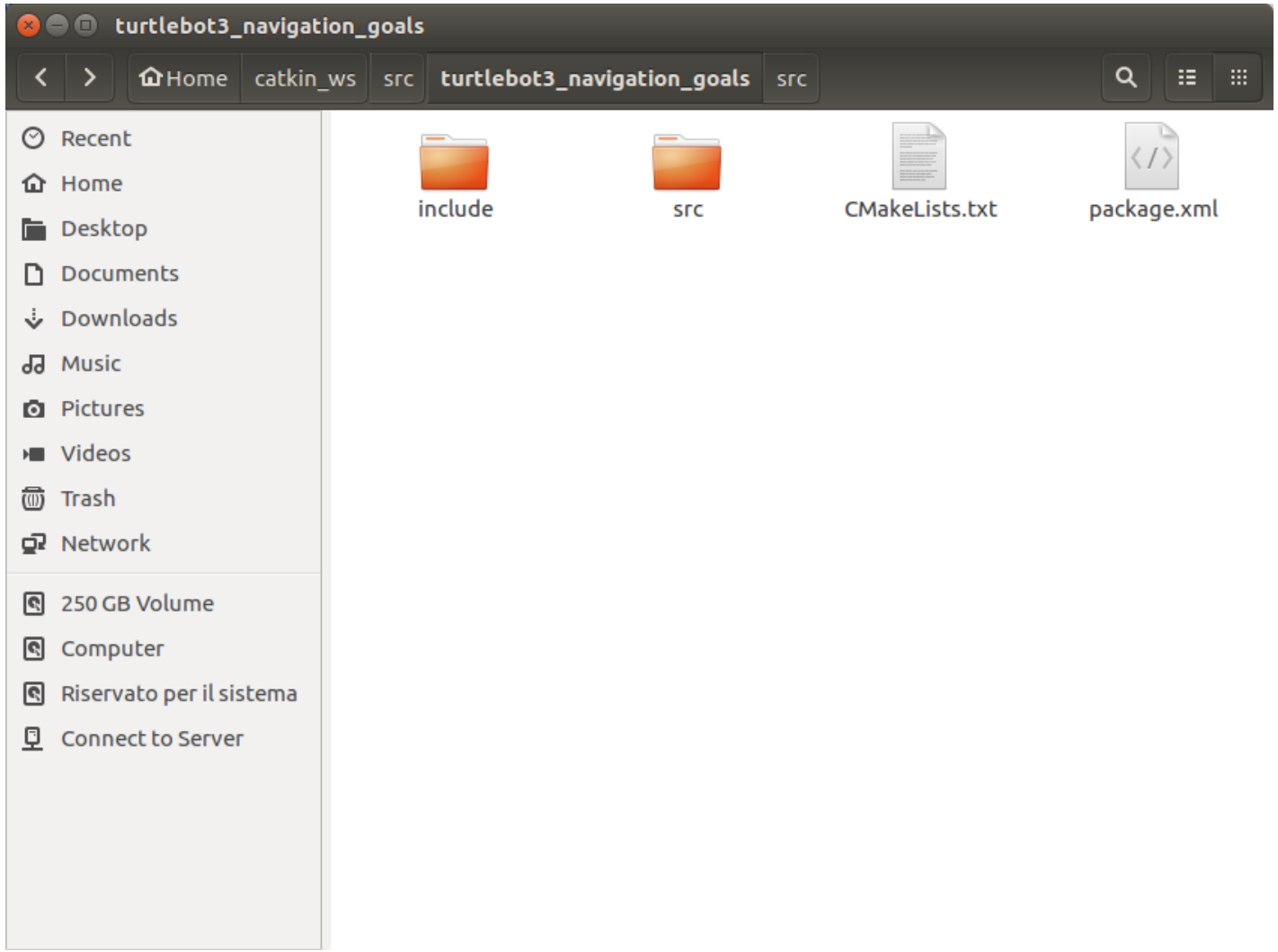
```
bloisi@bloisi-U36SG:~/catkin_ws/src$ catkin_create_pkg turtlebot3_navigation_goa
ls move_base_msgs actionlib roscpp
Created file turtlebot3_navigation_goals/CMakeLists.txt
Created file turtlebot3_navigation_goals/package.xml
Created folder turtlebot3_navigation_goals/include/turtlebot3_navigation_goals
Created folder turtlebot3_navigation_goals/src
Successfully created files in /home/bloisi/catkin_ws/src/turtlebot3_navigation_g
oals. Please adjust the values in package.xml.
```

turtlebot3_navigation_goal

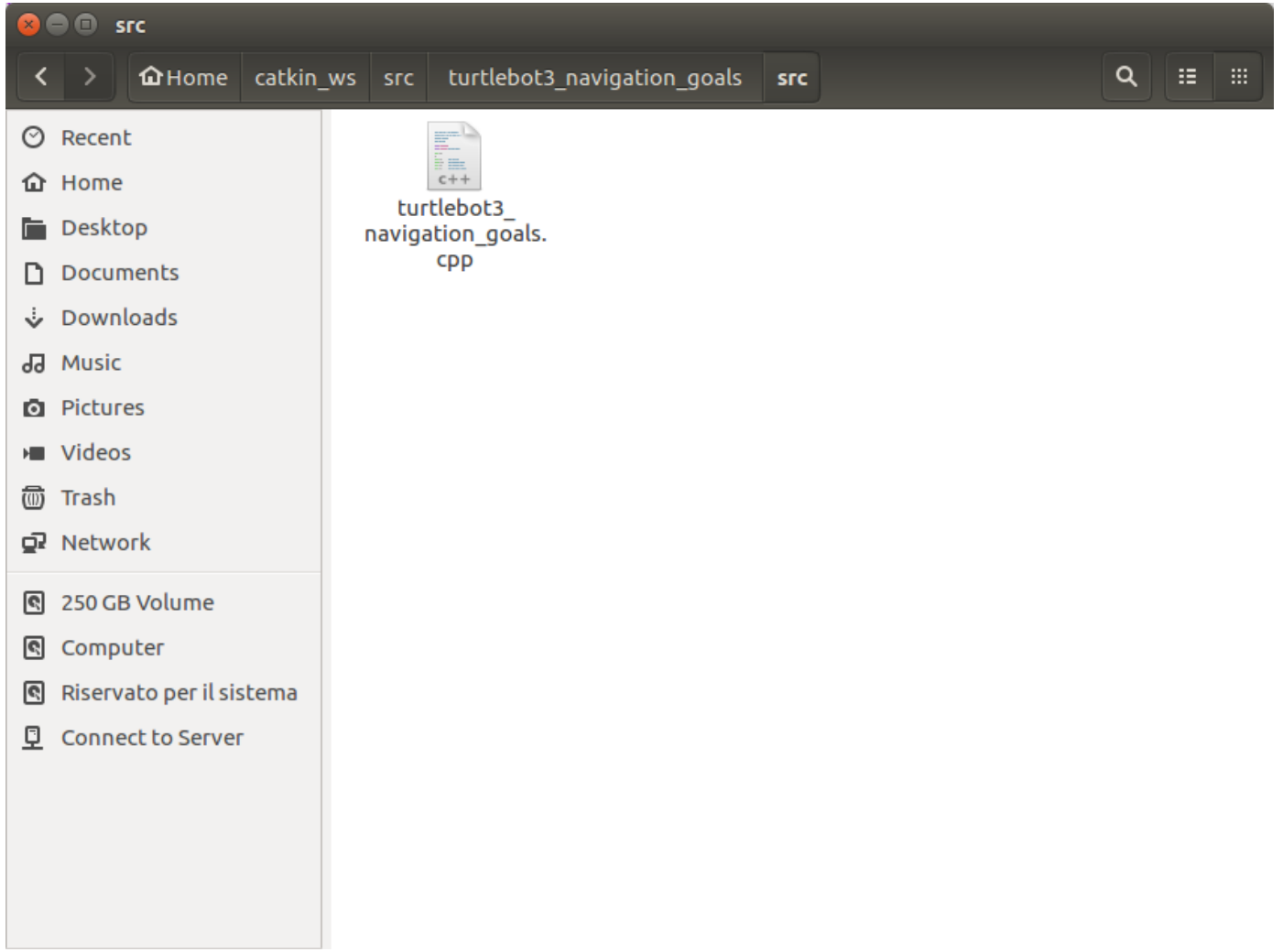
```
$ roscd turtlebot3_navigation_goals
```

```
bloisi@bloisi-U36SG:~/catkin_ws/src$ catkin_create_pkg turtlebot3_navigation_goa
ls move_base_msgs actionlib roscpp
Created file turtlebot3_navigation_goals/CMakeLists.txt
Created file turtlebot3_navigation_goals/package.xml
Created folder turtlebot3_navigation_goals/include/turtlebot3_navigation_goals
Created folder turtlebot3_navigation_goals/src
Successfully created files in /home/bloisi/catkin_ws/src/turtlebot3_navigation_g
oals. Please adjust the values in package.xml.
bloisi@bloisi-U36SG:~/catkin_ws/src$ roscd turtlebot3_navigation_goals
```

Files in the package



src folder


src
Home
catkin_ws
src
turtlebot3_navigation_goals
src
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Network
250 GB Volume
Computer
Riservato per il sistema
Connect to Server
c++
turtlebot3_
navigation_goals.
cpp

turtlebot3_navigation_goals.cpp

Open Save

```cpp
#include <ros/ros.h>
#include <move_base_msgs/MoveBaseAction.h>
#include <actionlib/client/simple_action_client.h>

typedef actionlib::SimpleActionClient<move_base_msgs::MoveBaseAction> MoveBaseClient;

int main(int argc, char** argv){
  ros::init(argc, argv, "simple_navigation_goals");

  //tell the action client that we want to spin a thread by default
  MoveBaseClient ac("move_base", true);

  //wait for the action server to come up
  while(!ac.waitForServer(ros::Duration(5.0))){
    ROS_INFO("Waiting for the move_base action server to come up");
  }

  move_base_msgs::MoveBaseGoal goal;

  //we'll send a goal to the robot to move 1 meter forward
  goal.target_pose.header.frame_id = "base_link";
  goal.target_pose.header.stamp = ros::Time::now();

  goal.target_pose.pose.position.x = 1.0;
  goal.target_pose.pose.orientation.w = 1.0;

  ROS_INFO("Sending goal");
  ac.sendGoal(goal);

  ac.waitForResult();

  if(ac.getState() == actionlib::SimpleClientGoalState::SUCCEEDED)
    ROS_INFO("Hooray, the base moved 1 meter forward");
  else
    ROS_INFO("The base failed to move forward 1 meter for some reason");

  return 0;
}

```

C++ Tab Width: 8 Ln 39, Col 1 INS

CmakeLists.txt

Open Save

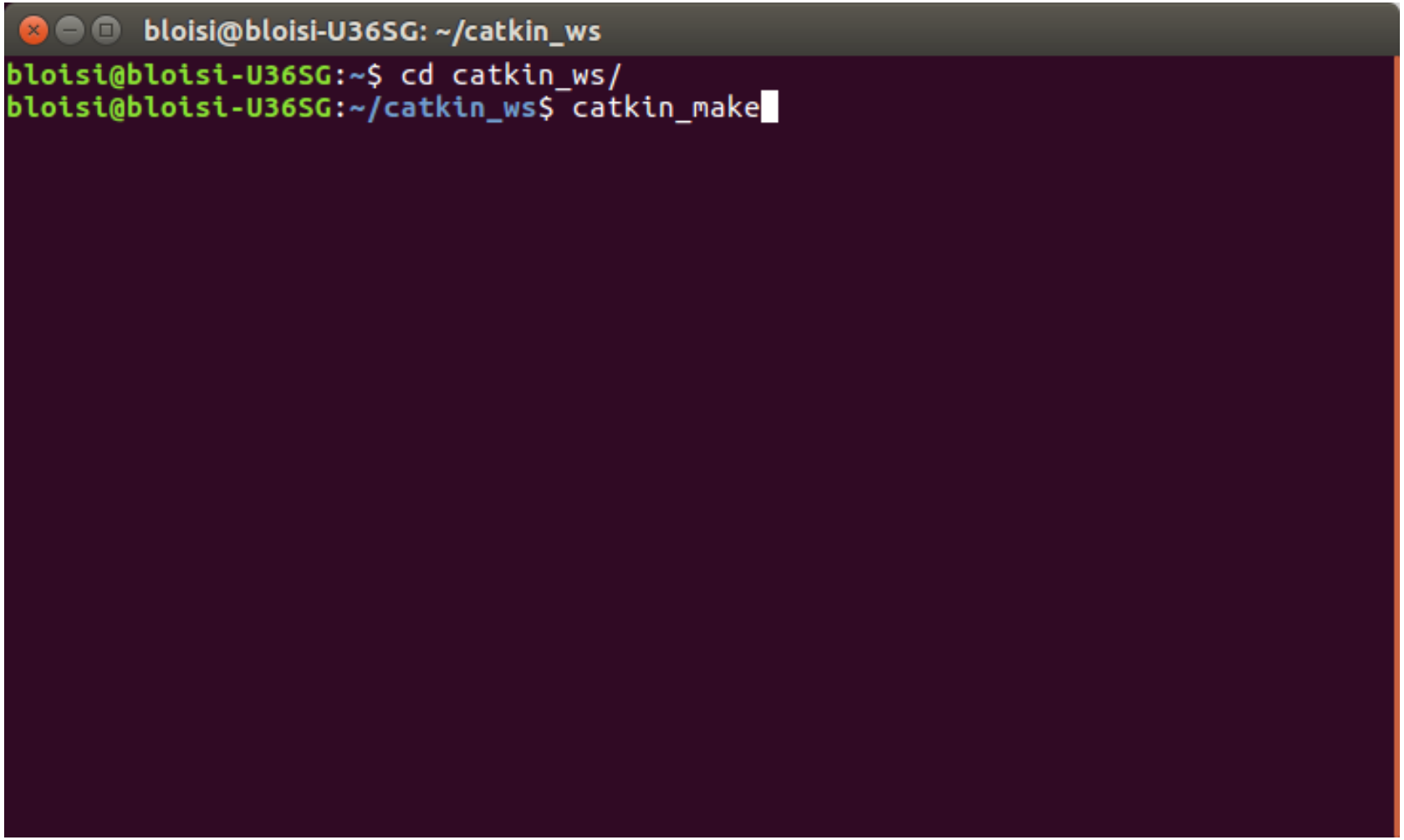
turtlebot3_navigation_goals.cpp × CMakeLists.txt ×

```
166
167 ## Mark executables and/or libraries for installation
168 # install(TARGETS ${PROJECT_NAME} ${PROJECT_NAME}_node
169 #   ARCHIVE DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
170 #   LIBRARY DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
171 #   RUNTIME DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
172 # )
173
174 ## Mark cpp header files for installation
175 # install(DIRECTORY include/${PROJECT_NAME}/
176 #   DESTINATION ${CATKIN_PACKAGE_INCLUDE_DESTINATION}
177 #   FILES_MATCHING PATTERN "*.h"
178 #   PATTERN ".svn" EXCLUDE
179 # )
180
181 ## Mark other files for installation (e.g. launch and bag files, etc.)
182 # install(FILES
183 #   # myfile1
184 #   # myfile2
185 #   DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION}
186 # )
187
188 ##############
189 ## Testing ##
190 ##############
191
192 ## Add gtest based cpp test target and link libraries
193 # catkin_add_gtest(${PROJECT_NAME}-test test/test_turtlebot3_navigation_goals.cpp)
194 # if(TARGET ${PROJECT_NAME}-test)
195 #   target_link_libraries(${PROJECT_NAME}-test ${PROJECT_NAME})
196 # endif()
197
198 ## Add folders to be run by python nosetests
199 # catkin_add_nosetests(test)
200
201 add_executable(turtlebot3_navigation_goals src/turtlebot3_navigation_goals.cpp)
202 target_link_libraries(turtlebot3_navigation_goals ${catkin_LIBRARIES})
203
```

CMake Tab Width: 8 Ln 203, Col 1 INS

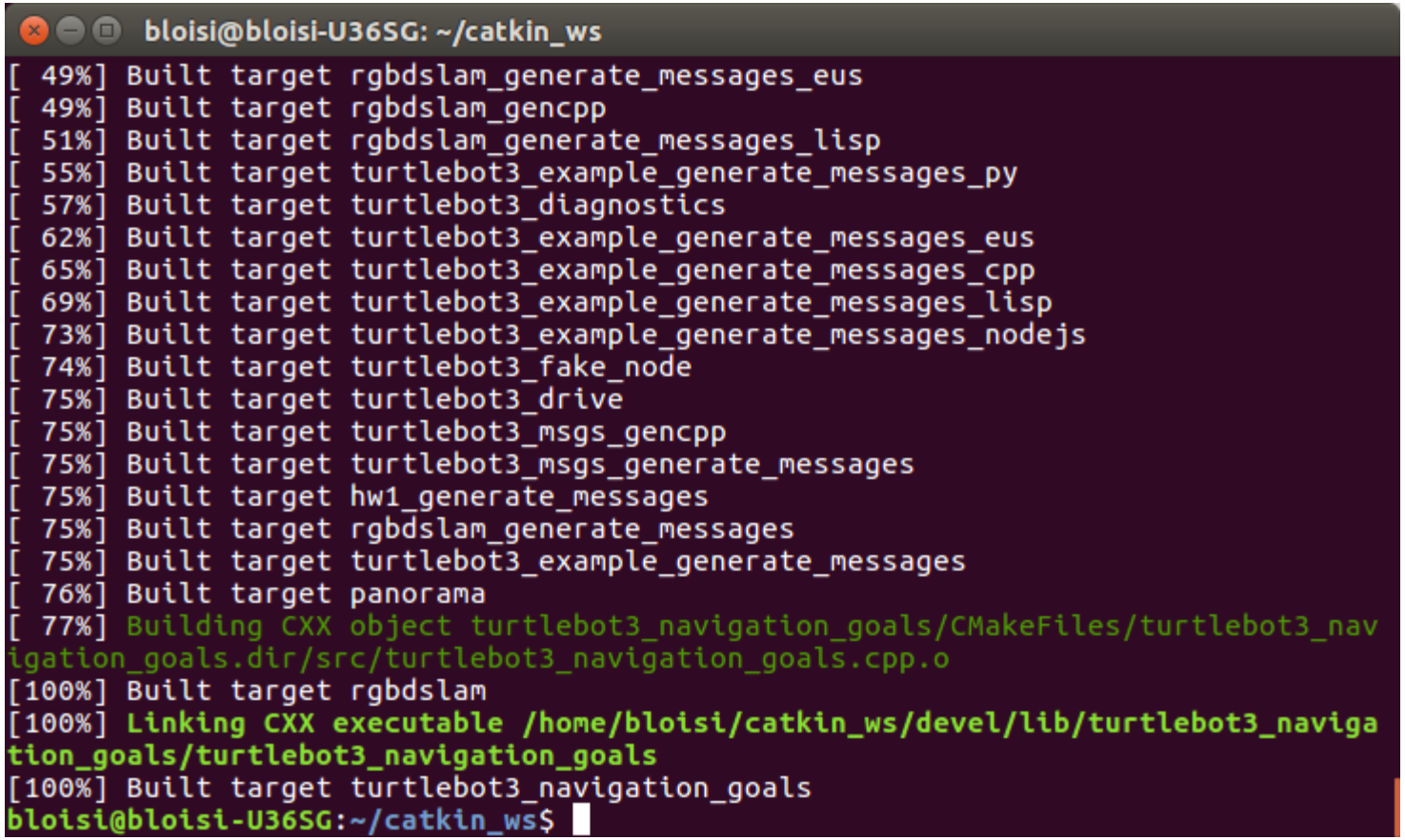
catkin_make

```
$ cd ~/catkin_ws
$ catkin_make
```

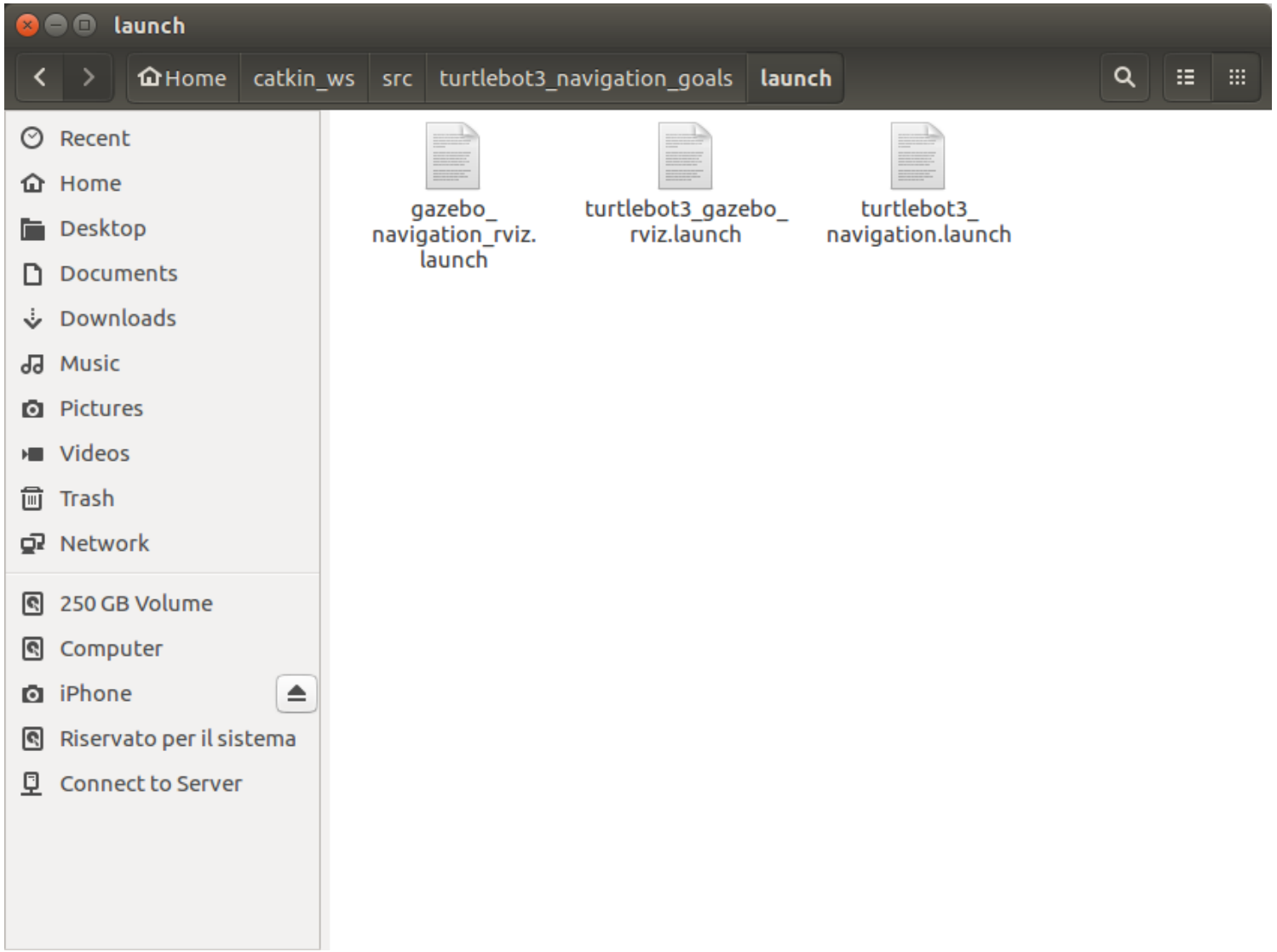


catkin_make - execution

```
$ cd ~/catkin_ws
$ catkin_make
```



launch files



gazebo_navigation_rviz.launch

```
<launch>

    <include file="$(find turtlebot3_gazebo)/launch/turtlebot3_world.launch"/>
    <include file="$(find turtlebot3_navigation_goals)/launch/turtlebot3_navigation.launch"/>
    <include file="$(find turtlebot3_navigation_goals)/launch/turtlebot3_gazebo_rviz.launch"/>

</launch>
```

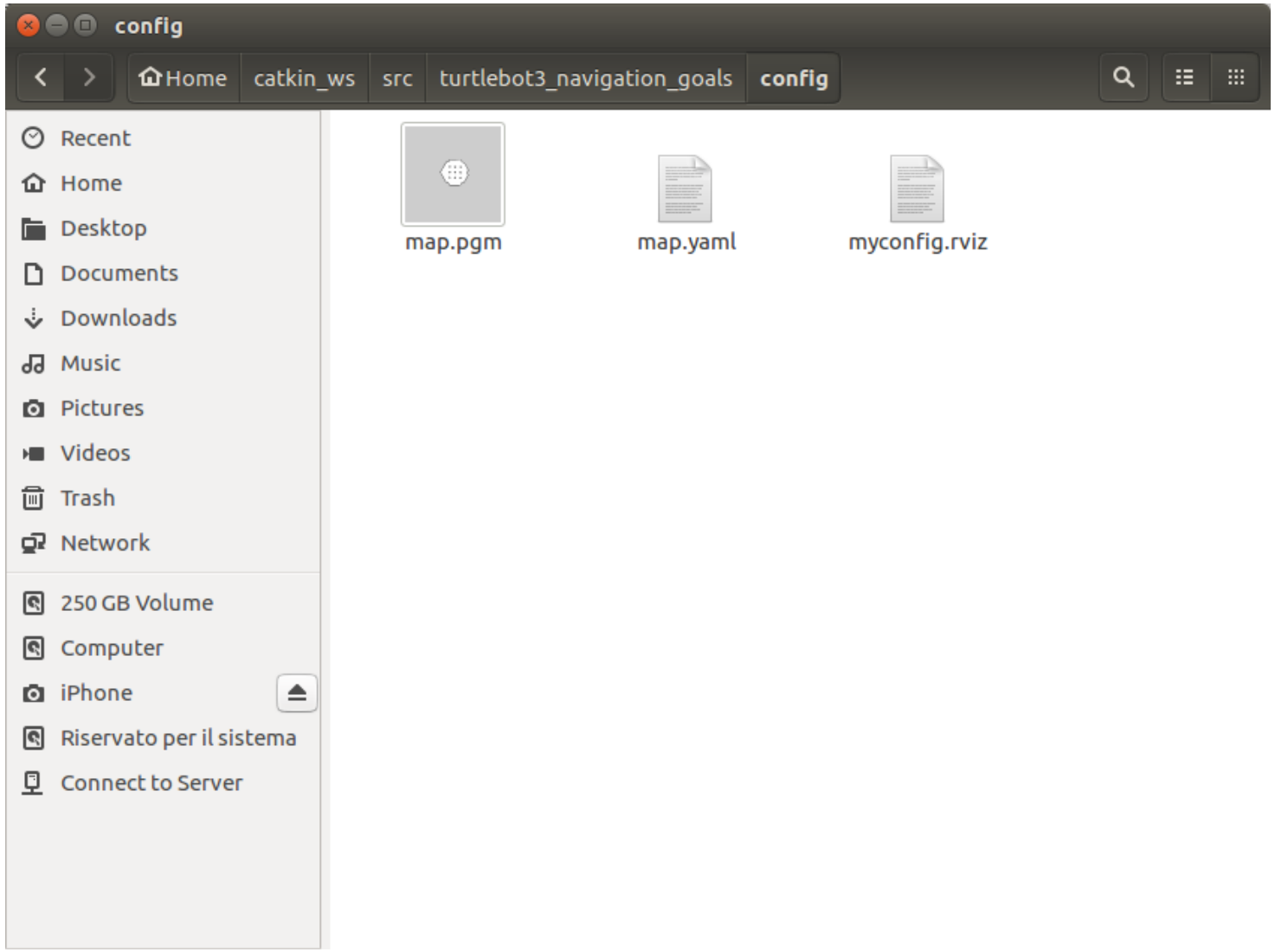
turtlebot3_navigation.launch

```
<launch>
  <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle]"/>

  <!-- Turtlebot3 -->
  <include file="$(find turtlebot3_bringup)/launch/turtlebot3_remote.launch" />

  <!-- Map server -->
  <arg name="map_file" default="$(find turtlebot3_navigation_goals)/config/map.yaml"/>
  <node name="map_server" pkg="map_server" type="map_server" args="$(arg map_file)">
  </node>

  <!-- AMCL -->
  <include file="$(find turtlebot3_navigation)/launch/amcl.launch"/>

  <!-- move_base -->
  <arg name="cmd_vel_topic" default="/cmd_vel" />
  <arg name="odom_topic" default="odom" />
  <node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen">
    <param name="base_local_planner" value="dwa_local_planner/DWAPlannerROS" />

    <rosparam file="$(find turtlebot3_navigation)/param/costmap_common_params_$(arg model).yaml" command="load" ns="global_costmap" />
    <rosparam file="$(find turtlebot3_navigation)/param/costmap_common_params_$(arg model).yaml" command="load" ns="local_costmap" />
    <rosparam file="$(find turtlebot3_navigation)/param/local_costmap_params.yaml" command="load" />
    <rosparam file="$(find turtlebot3_navigation)/param/global_costmap_params.yaml" command="load" />
    <rosparam file="$(find turtlebot3_navigation)/param/move_base_params.yaml" command="load" />
    <rosparam file="$(find turtlebot3_navigation)/param/dwa_local_planner_params_waffle.yaml" command="load" />

    <remap from="cmd_vel" to="$(arg cmd_vel_topic)"/>
    <remap from="odom" to="$(arg odom_topic)"/>
  </node>
</launch>
```

turtlebot3_gazebo_rviz.launch

```
<launch>
  <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle]"/>

  <include file="$(find turtlebot3_bringup)/launch/includes/description.launch.xml">
    <arg name="model" value="$(arg model)" />
  </include>

  <!-- Commented out this node since it is already launched in the navigation launch file or in included files>
  <node pkg="robot_state_publisher" type="robot_state_publisher" name="robot_state_publisher" output="screen">
    <param name="publish_frequency" type="double" value="50.0" />
  </node> -->

  <node name="rviz" pkg="rviz" type="rviz" args="-d $(find turtlebot3_navigation_goals)/config/myconfig.rviz"/>

</launch>
```

config folder

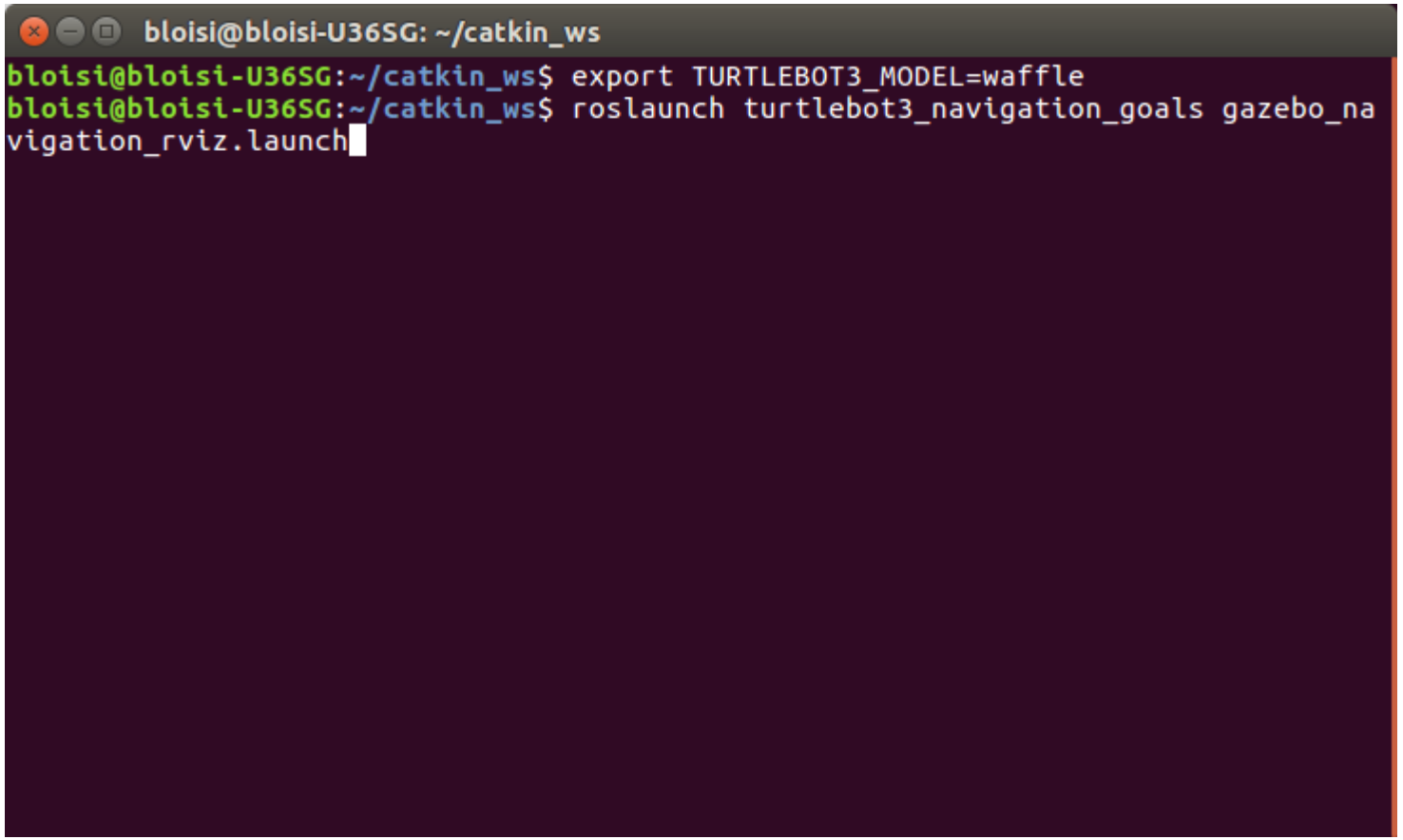


Setting Turtlebot3 model

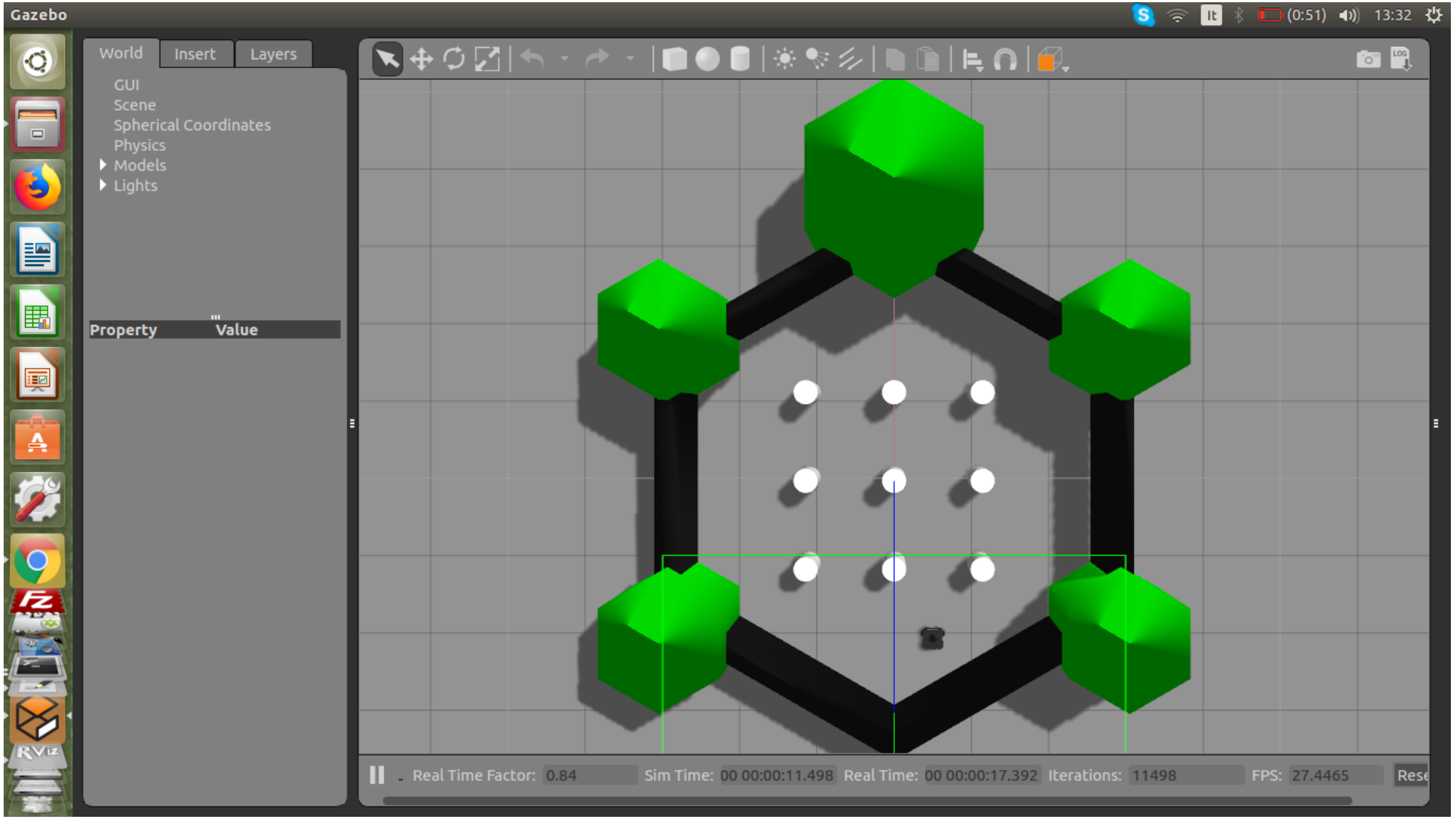
```
$ export TURTLEBOT3_MODEL=waffle
```

gazebo_navigation_rviz.launch

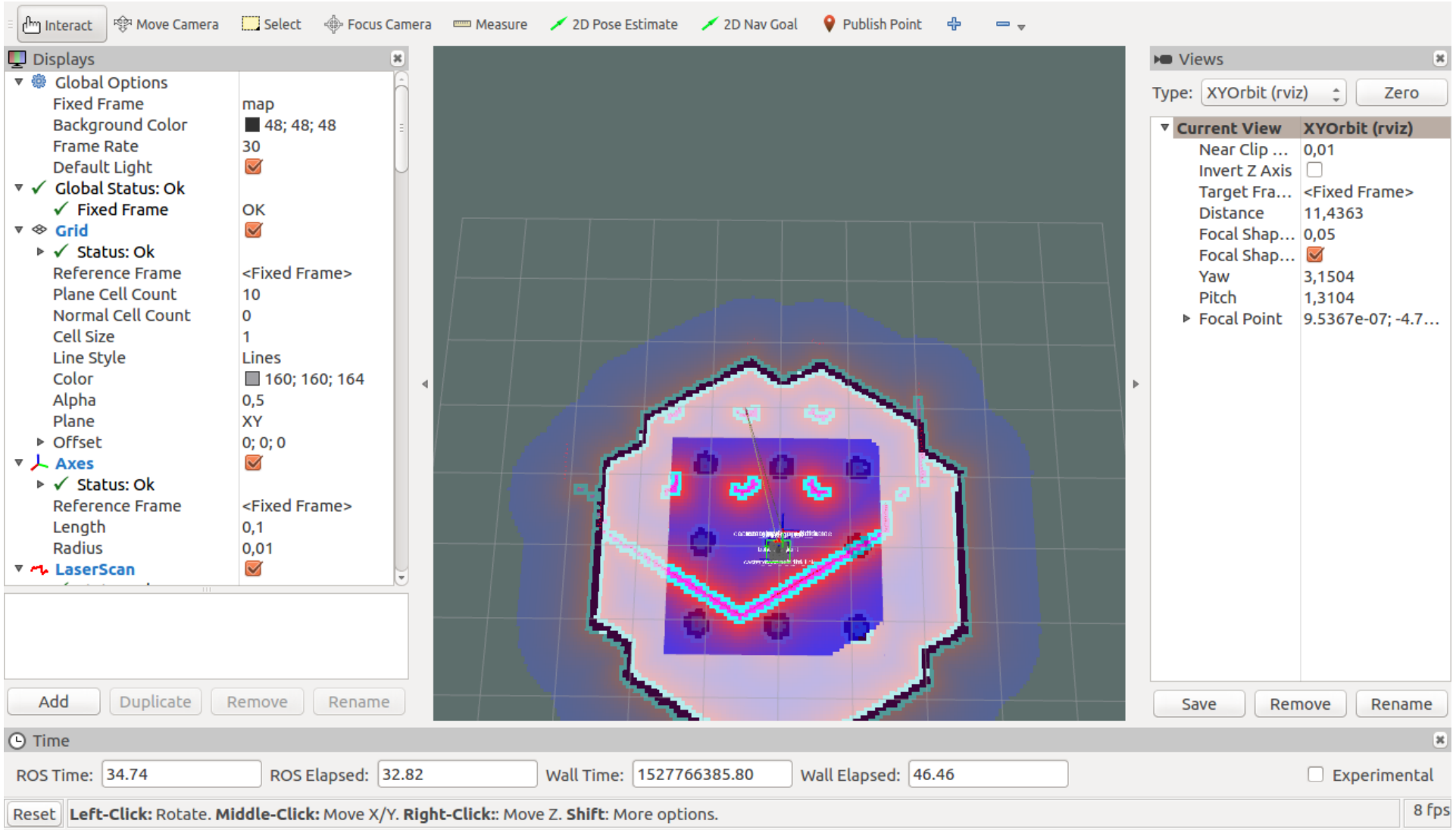
```
$ roslaunch turtlebot3_navigation_goals gazebo_navigation_rviz.launch
```



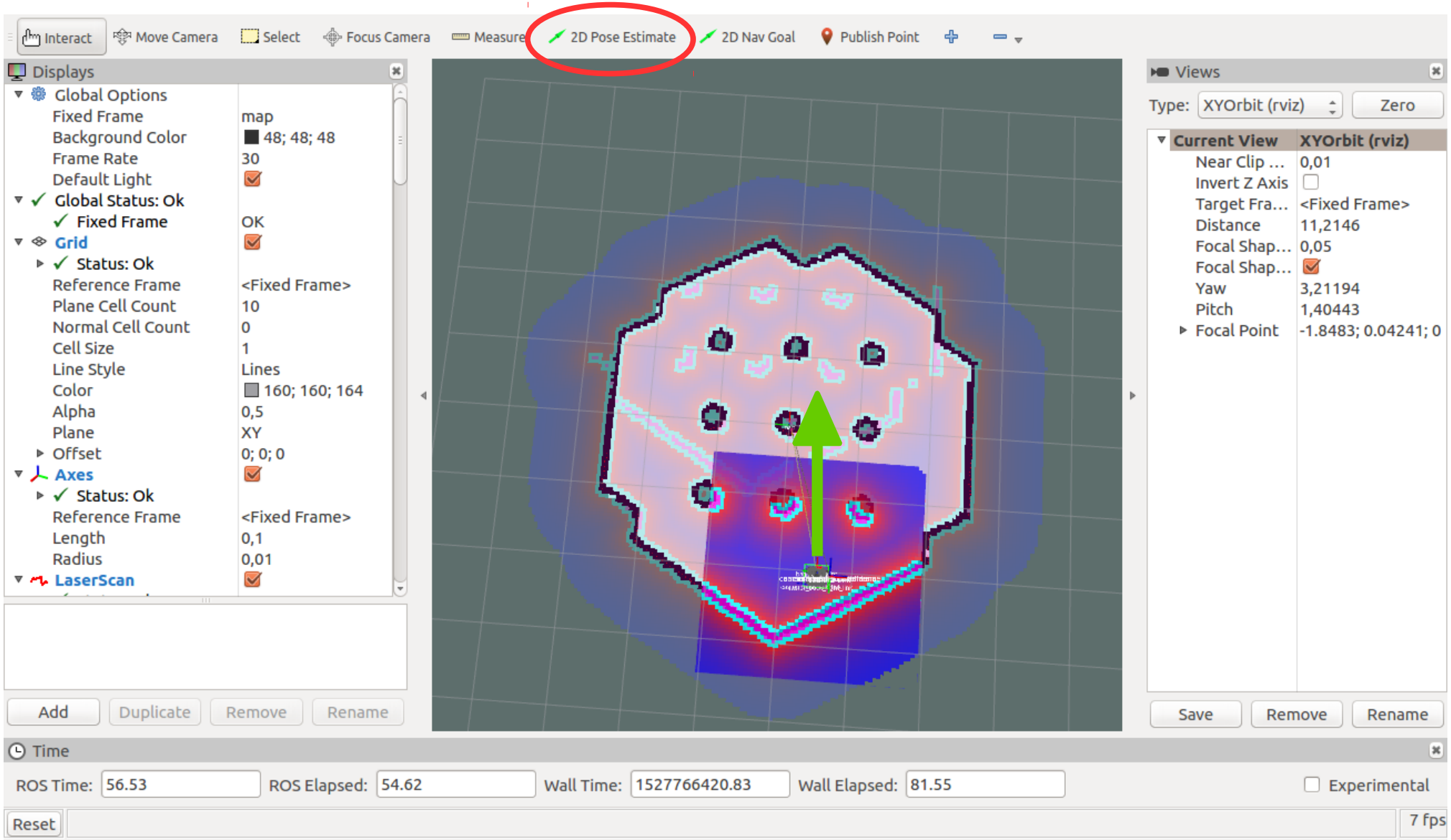
Gazebo


Gazebo
(0:51)
13:32
World
Insert
Layers
GUI
Scene
Spherical Coordinates
Physics
Models
Lights
Property
Value
Real Time Factor: 0.84
Sim Time: 00 00:00:11.498
Real Time: 00 00:00:17.392
Iterations: 11498
FPS: 27.4465

RViz

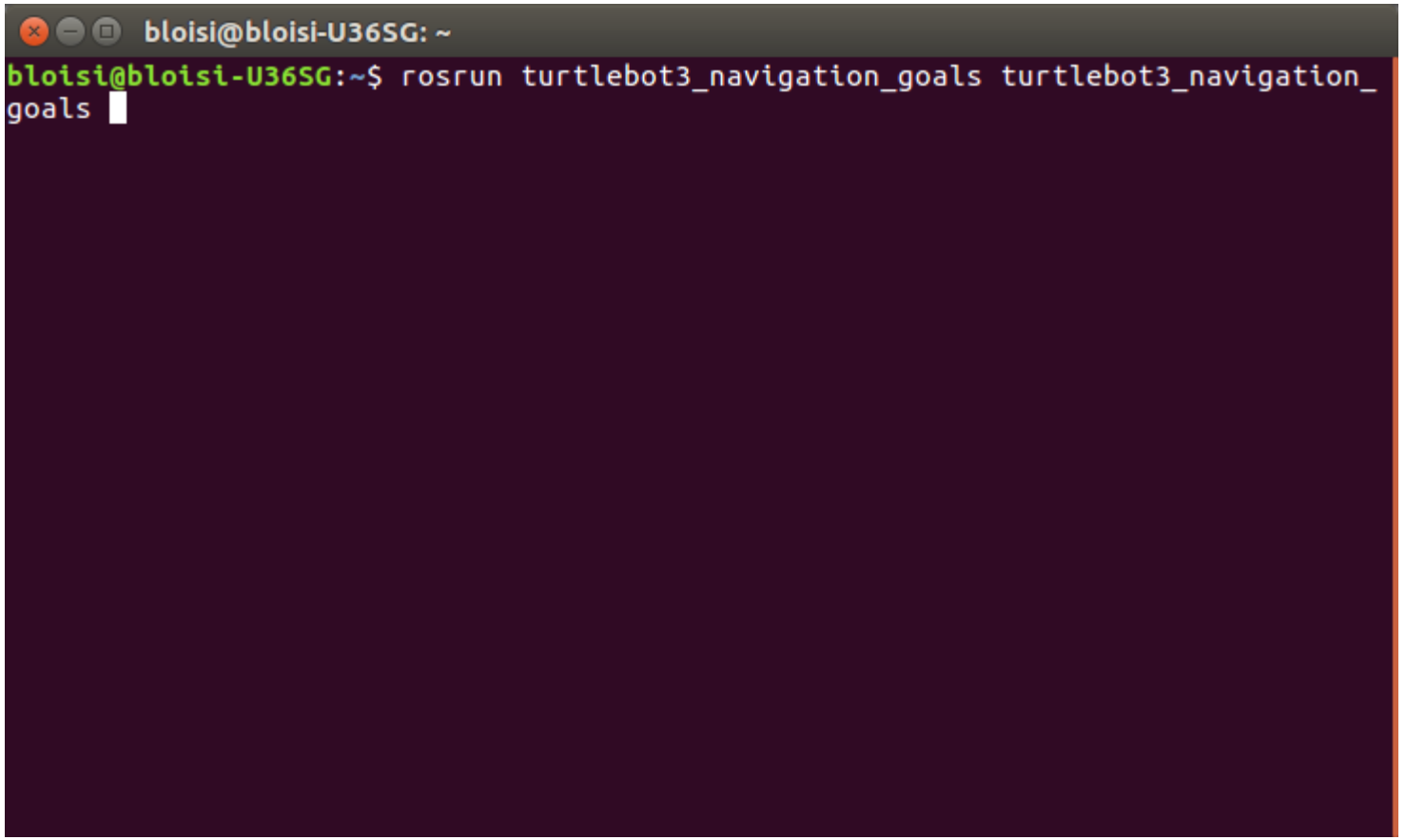

Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Global Options
Fixed Frame map
Background Color 48; 48; 48
Frame Rate 30
Default Light
Global Status: Ok
Fixed Frame OK
Grid
Status: Ok
Reference Frame <Fixed Frame>
Plane Cell Count 10
Normal Cell Count 0
Cell Size 1
Line Style Lines
Color 160; 160; 164
Alpha 0,5
Plane XY
Offset 0; 0; 0
Axes
Status: Ok
Reference Frame <Fixed Frame>
Length 0,1
Radius 0,01
LaserScan
Add
Duplicate
Remove
Rename
Views
Type: XYOrbit (rviz)
Zero
Current View XYOrbit (rviz)
Near Clip … 0,01
Invert Z Axis
Target Fra… <Fixed Frame>
Distance 11,4363
Focal Shap… 0,05
Focal Shap…
Yaw 3,1504
Pitch 1,3104
Focal Point 9.5367e-07; -4.7…
Save
Remove
Rename
Time
ROS Time: 34.74
ROS Elapsed: 32.82
Wall Time: 1527766385.80
Wall Elapsed: 46.46
Experimental
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Move Z. Shift: More options.
8 fps

Pose estimation

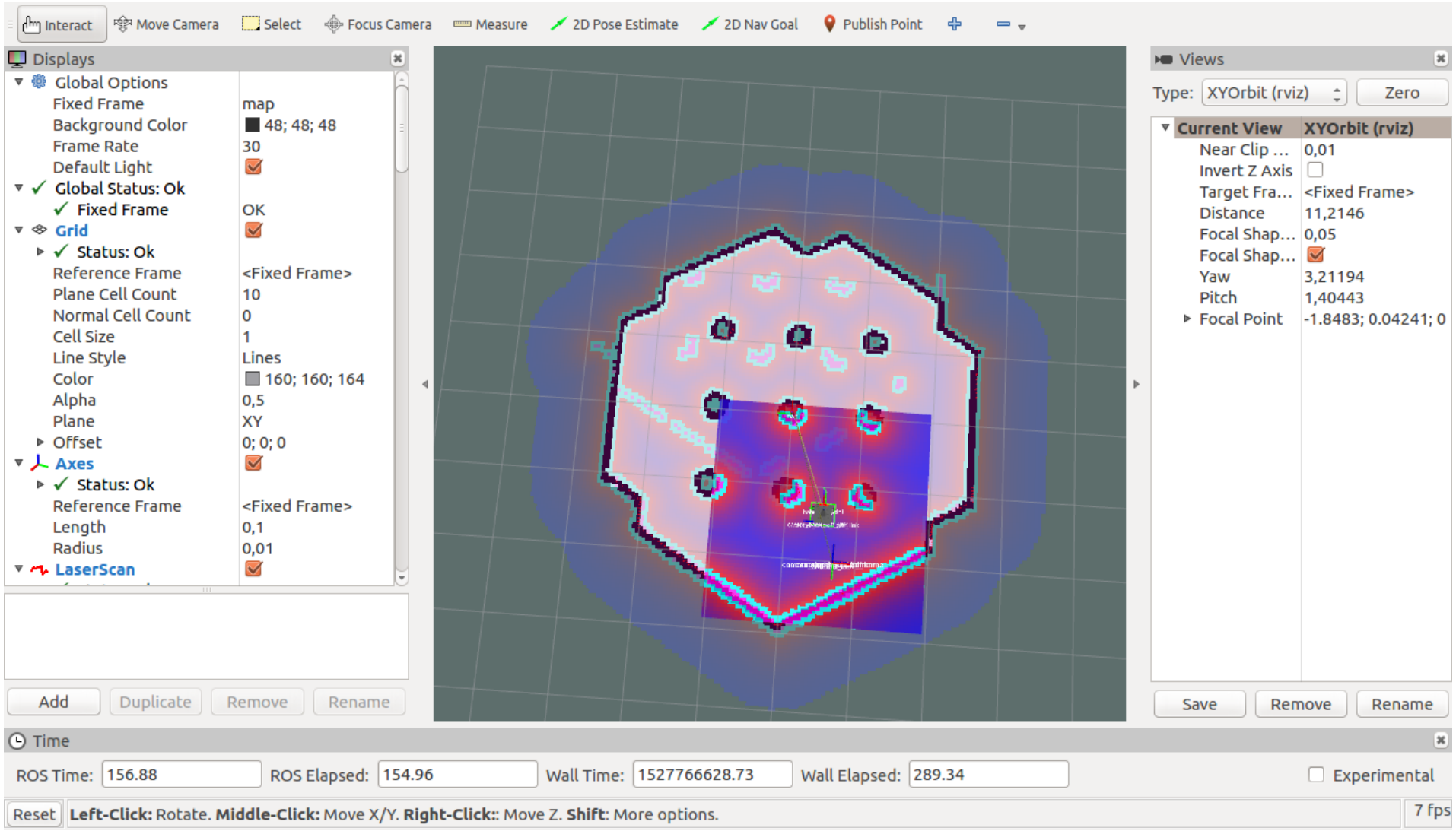


Sending the goal

```
$ rosrun turtlebot3_navigation_goals turtlebot3_navigation_goals
```



Reaching the goal

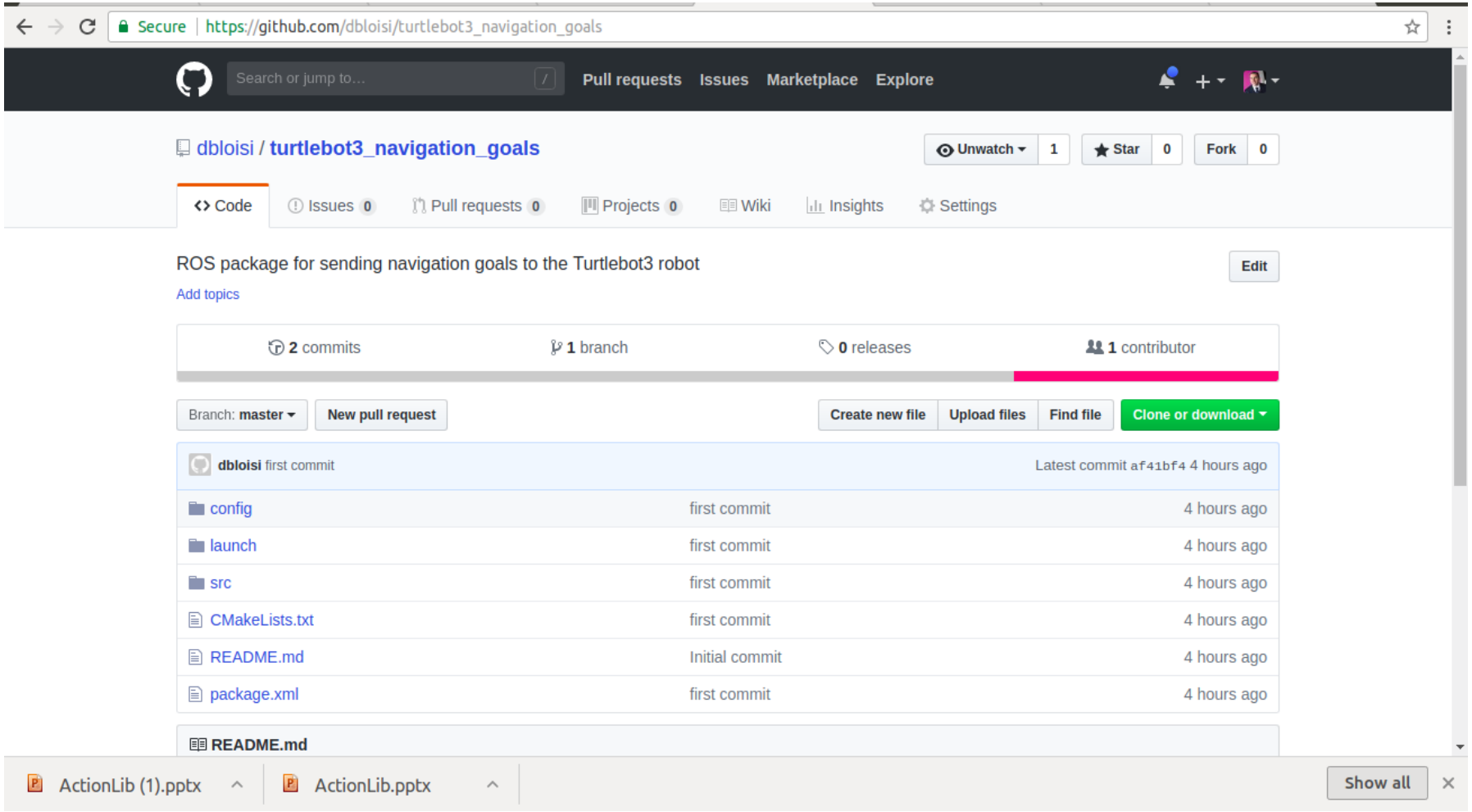


Goal reached

```
bloisi@bloisi-U36SG:~/catkin_ws/src/turtlebot3_navigation_goals/launch$ rosrun t
urtlebot3_navigation_goals turtlebot3_navigation_goals
[ INFO] [1528132840.163758075, 71.877000000]: Sending goal
[ INFO] [1528132840.164147895, 71.877000000]: Goal sent, waiting for results
[ INFO] [1528132877.513702526, 87.077000000]: Hooray, the base moved 1 meter for
ward
bloisi@bloisi-U36SG:~/catkin_ws/src/turtlebot3_navigation_goals/launch$
```

Git repo

https://github.com/dbloisi/turtlebot3_navigation_goals.git



Esercizio 1

Modificare il codice del package turtlebot3_navigation_goals in modo che siano inviati più task al robot invece di un singolo task

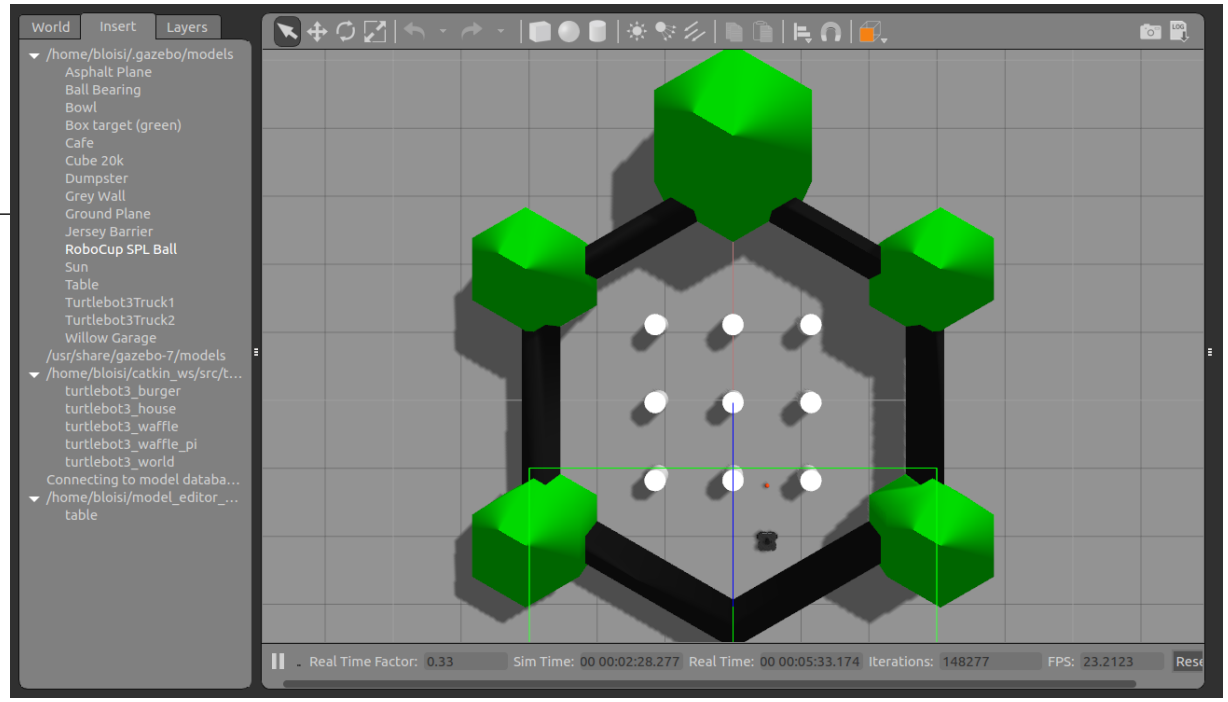
Esercizio 2

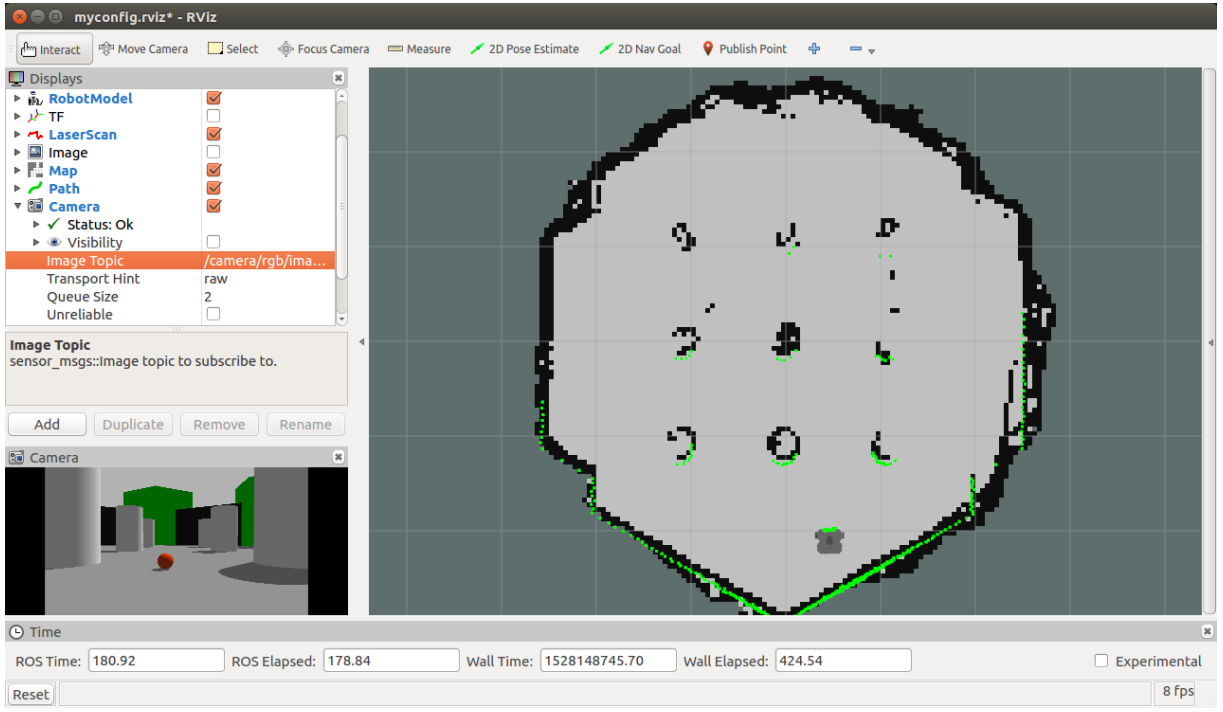
Eseguire il package turtlebot3_navigation_goal utilizzando un mondo virtuale Gazebo diverso da quello TurtleBot3 world

Esercizio 2

Scrivere un nuovo package denominato turtlebot3_visual_goal in grado di

1. individuare la RoboCup SPL ball nella scena attraverso la camera del robot
2. utilizzare la posizione della palla come goal facendo navigare il robot verso di essa





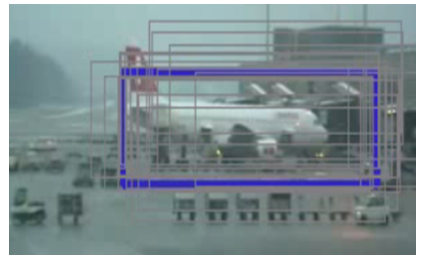
UNIVERSITÀ di VERONA

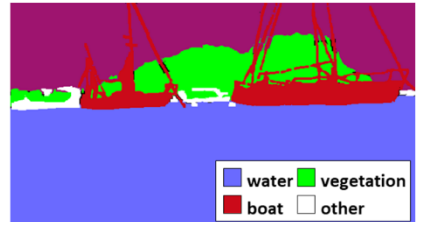
Dipartimento di INFORMATICA

Corso di Laboratorio Ciberfisico
Modulo di Robot Programming with ROS









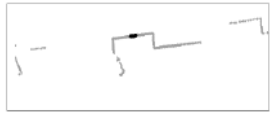
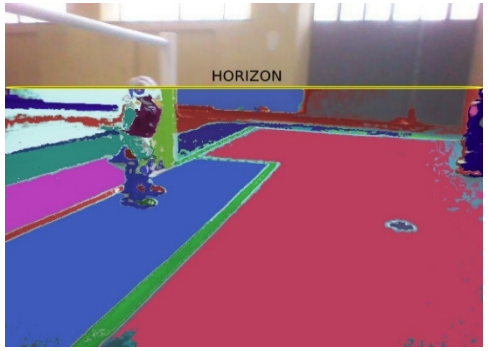
actionlib

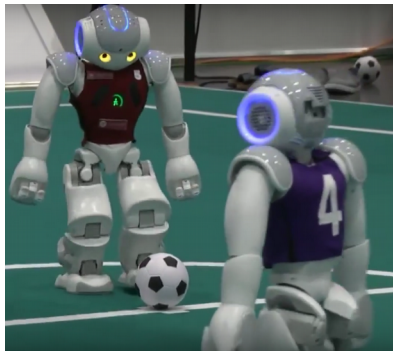
ROS

Docente:

Domenico Daniele Bloisi





Giugno 2018