Tutorial 3: planning and robot navigation

RoboCup@Home Practical Course WS18/19

Institute for Cognitive Systems

Department of Electrical and Computer Engineering

Technical University of Munich

Dr. Karinne Ramirez-Amaro

Dr. Pablo Lanillos

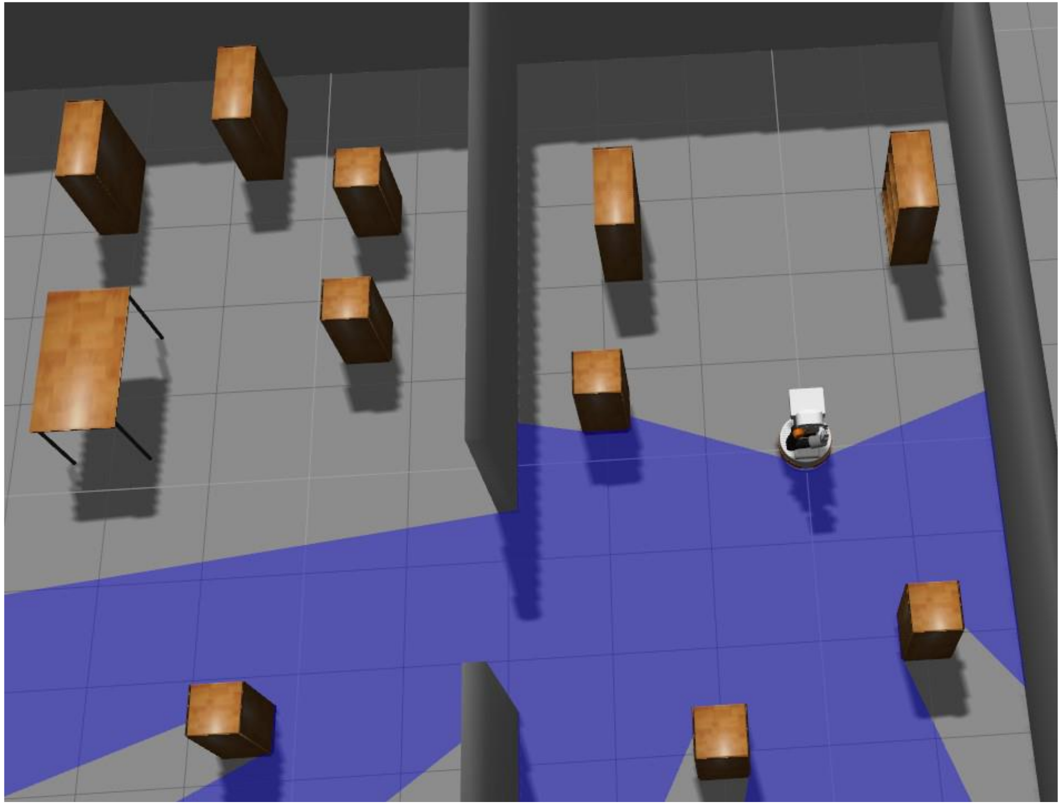
M.Sc. Rogelio Guadarrama

M.Sc. Constantin Uhde

Dr. Gordon Cheng

Munich, 2. May 2018

Navigation, Localization, Mapping



Three distinct problems:

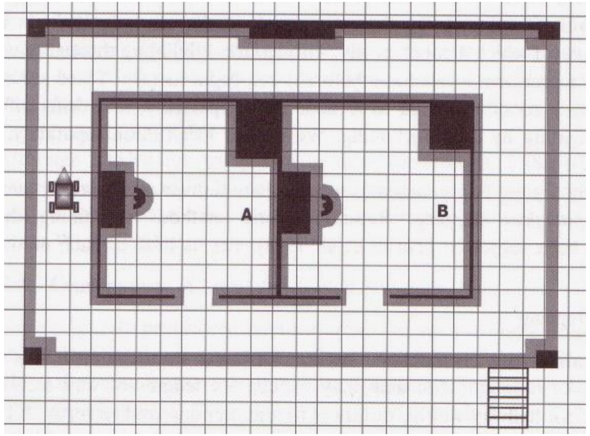
- Get from A to B
- Locate the Robot
- Represent the Environment

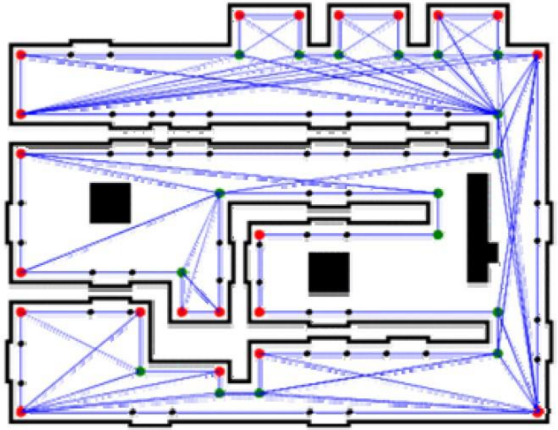
Mapping

Represent the robot environment by means of a Map.

Possible representations:

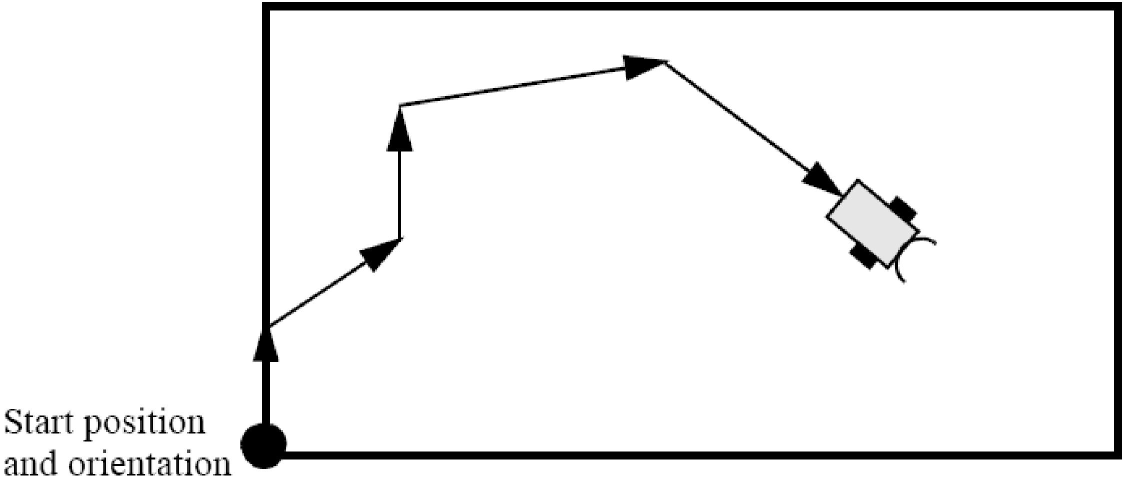
- Grid based / metric map: Environment is mapped with an evenly spaced grid, with each grid cell storing its probability of occupancy
- Topological map: depicts landmarks and their relations. The map is represented by a graph with nodes corresponding to landmarks and edges representing valid paths between these locations.





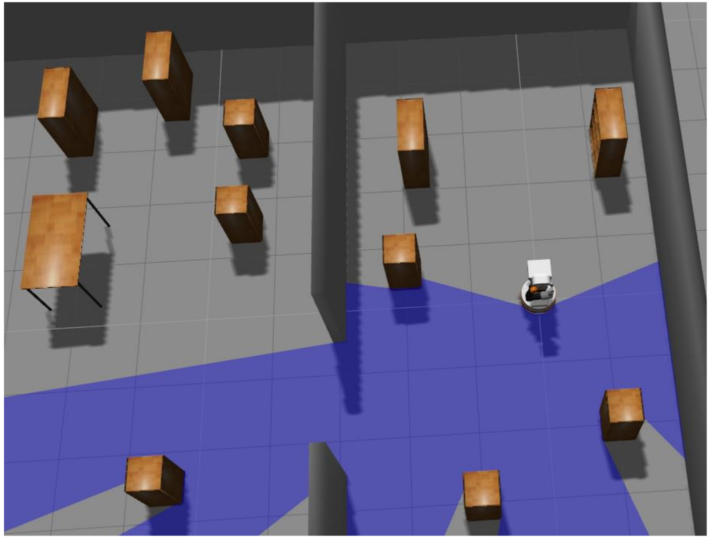
Localization

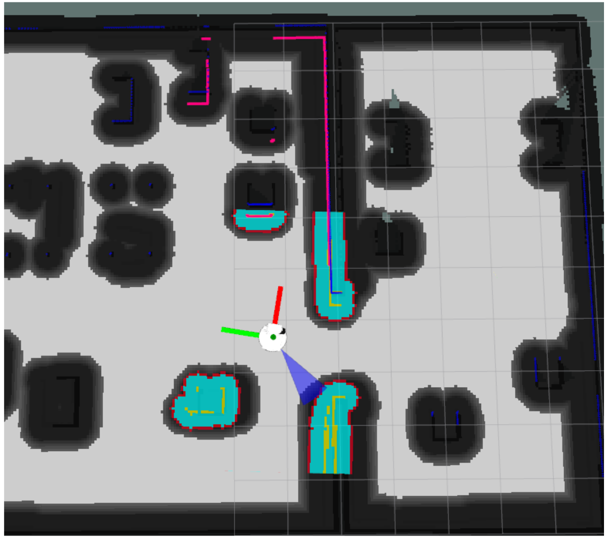
Robot position with respect to the environment



Navigation

Find a path to a goal location given a localized robot and a map of the environment



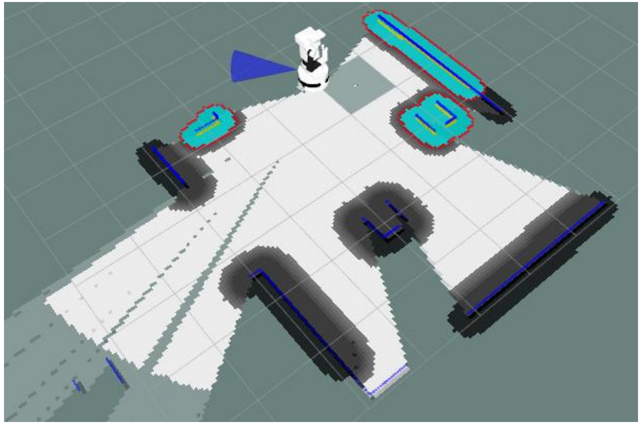


SLAM – Simultaneous Localization and Mapping

In practice, localization and mapping cannot be solved independently.

SLAM builds a map and localizes the robot in it at the same time.

Luckily “gmapping” provides this capability for us

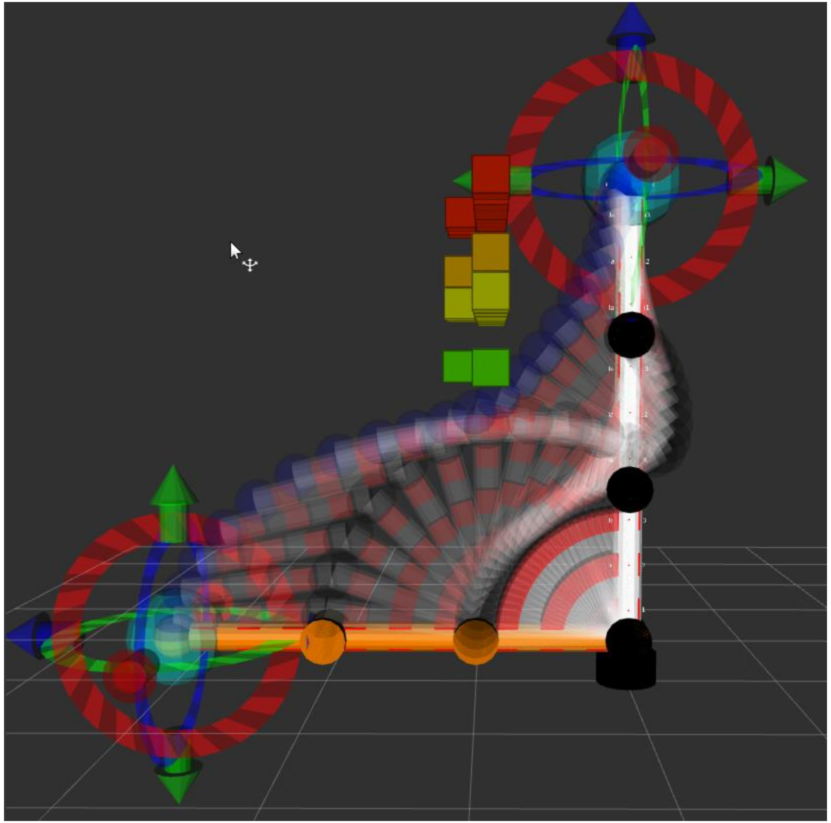


Motion Planning

Planning in typically high dimensional joint space

Can include velocity, acceleration and other constraints

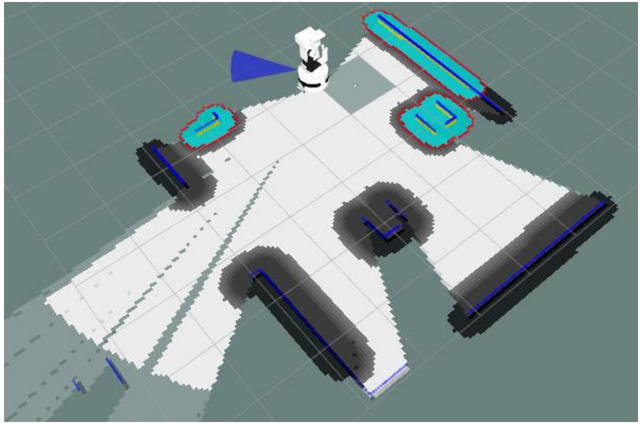
Used for robotic manipulators, humanoids, etc.



Mapping in ROS

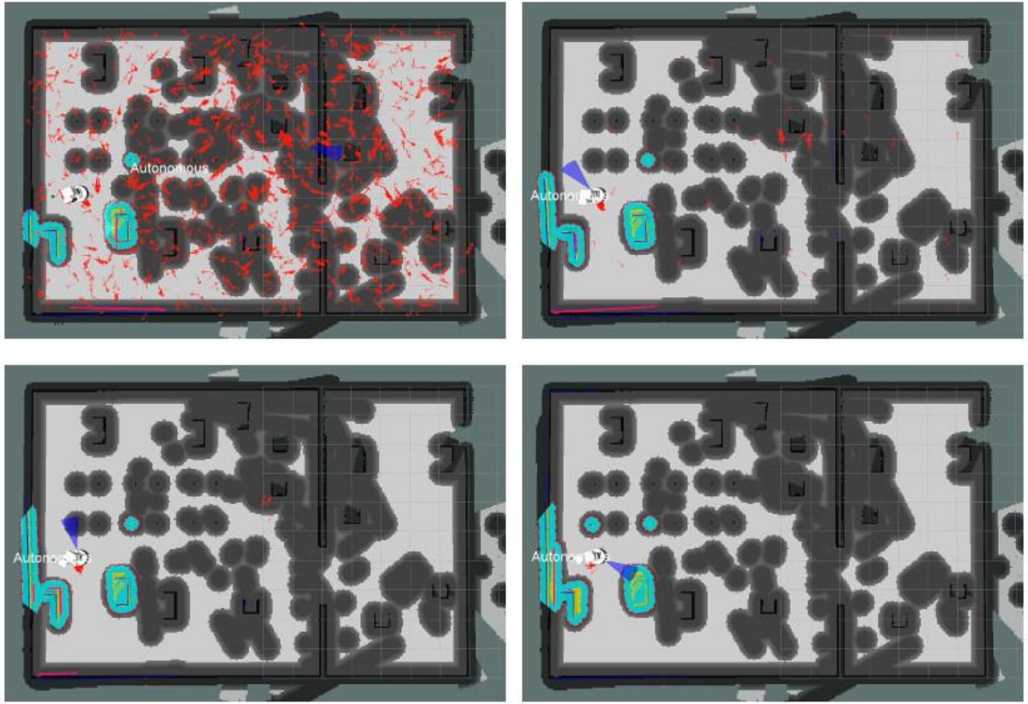
“gmapping” package allows to build 2D occupancy grid maps with laser sensor data and robot position
wiki.ros.org/gmapping

But:
If the position of the recording sensor is not precisely known,
Recorded data can’t be put into place on a map.
Therefore we need an approach that simultaneously localizes the robot (for the sensor position) and maps the environment

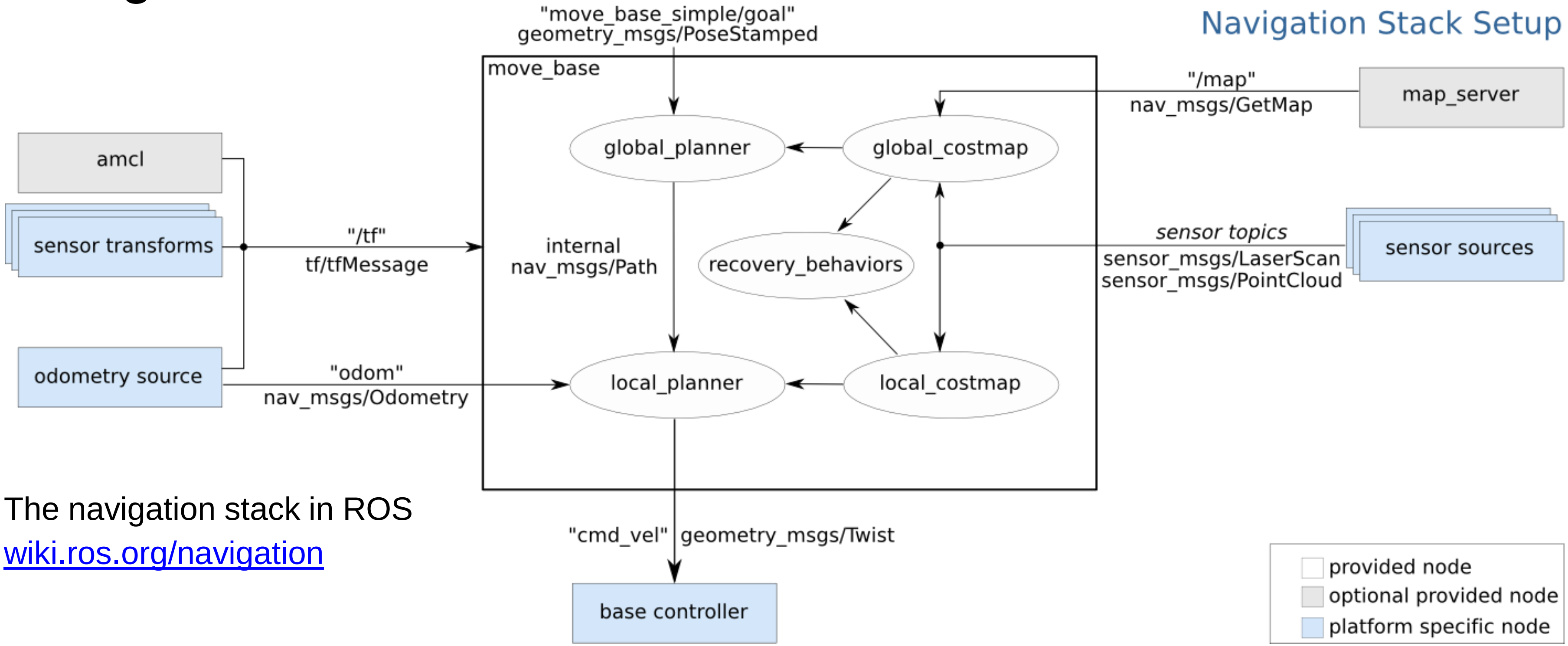


Localization in ROS

“amcl” package implements probabilistic localization. The employed technique is called Monte Carlo Localization.
wiki.ros.org/amcl



Navigation in ROS



The navigation stack in ROS
wiki.ros.org/navigation

Motion Planning in ROS

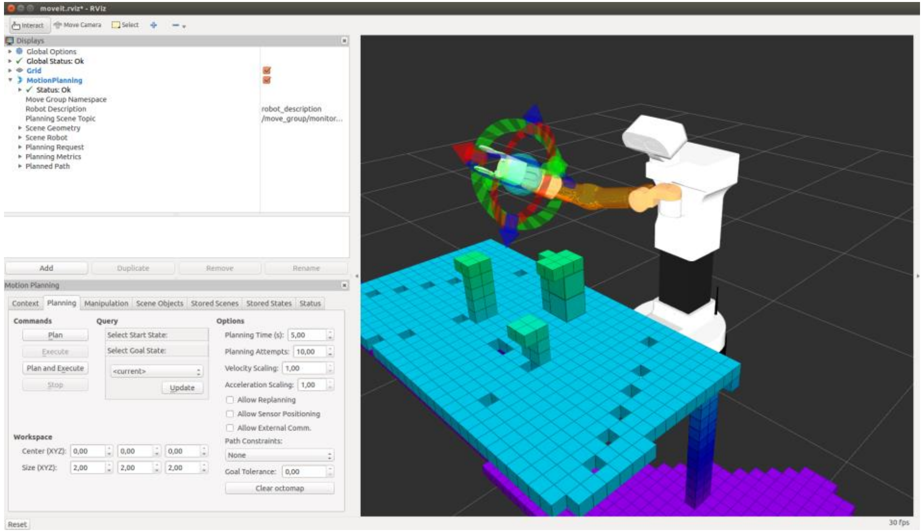
Implements a list of planning algorithms through “Open Motion Planning Library”

Uses 3D gridmaps for obstacle avoidance called “Octomap”

Provides joint- and cartesian space planning for kinematic chains

Planning GUI available in RVIZ

MoveIt!
moving robots into the future
moveit.ros.org



http://blog.pal-robotics.com/wp-content/uploads/2018/01/rviz_octomap_interact-ros-tiago-pal-robotics-moveit.png

Move robot to goal using C++

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

In order to move robot 1 meter forward from his current location use :

```
move_base_msgs::MoveBaseGoal goal;
goal.target_pose.header.stamp =
ros::Time::now();
//reference frame is current location of the robot
goal.target_pose.header.frame_id = "base_link";
//moving distance is 1 meter
goal.target_pose.pose.position.x = 1.0;
//robot is moving forward
goal.target_pose.pose.orientation.w = 1.0;
```

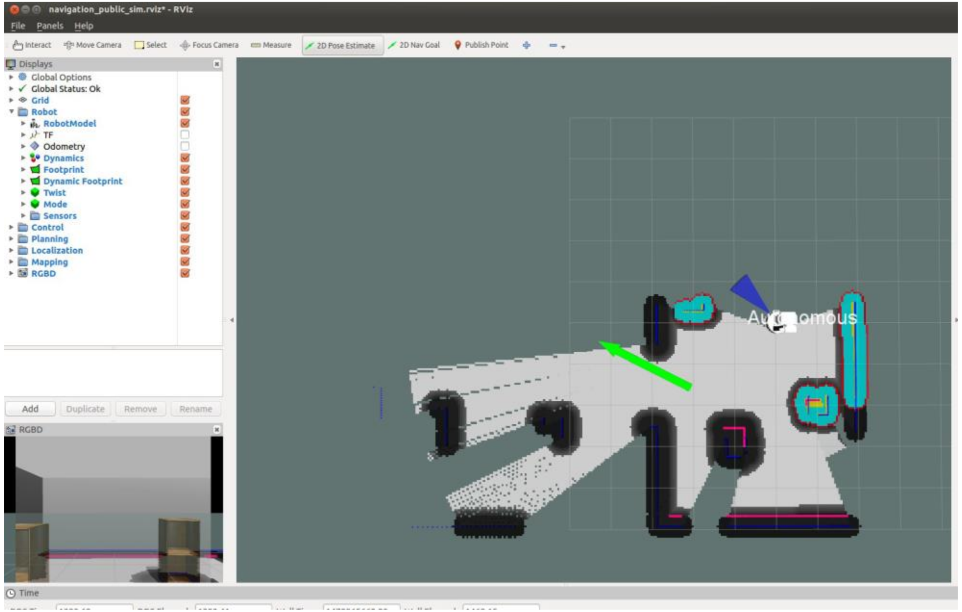
Move robot to goal using C++

To move the robot to a specific goal on the map do:

```
move_base_msgs::MoveBaseGoal goal;
goal.target_pose.header.stamp =
ros::Time::now();
//reference frame is map
goal.target_pose.header.frame_id = "map";
//set goal as 2D coordinate
goal.target_pose.pose.position.x = 1.22;
goal.target_pose.pose.position.y = -0.56;
//set goal orientation
goal.target_pose.pose.orientation.x = 0.0;
goal.target_pose.pose.orientation.y = 0.0;
goal.target_pose.pose.orientation.z = 0.91;
goal.target_pose.pose.orientation.w = 0.43;
```

Find goal coordinates on the map

Use rviz button 2D Pose Estimate to select desired position. Hold your mouse, and select desired orientation by rotating the arrow.



The resulted position will be published to topic:
/initialpose

Make sure you subscribed to it before using 2D Pose Estimate

Exercises

Before doing the exercises, follow along these tutorials:

- wiki.ros.org/Robots/TIAGo/Tutorials/Navigation/Mapping
- wiki.ros.org/Robots/TIAGo/Tutorials/Navigation/Localization
- wiki.ros.org/Robots/TIAGo/Tutorials/MoveIt/Planning_joint_space
- https://wiki.ros.org/Robots/TIAGo/Tutorials/MoveIt/Planning_cartesian_space

There are 2 distinct tasks:

1. Generate a map using gmapping
2. Navigate Tiago in the map by creating a package and using amcl
3. Flip a table using MoveIt!

Create one tar file with subfolders “task_1”, “task_2” and “task_3” for the three tasks and the following naming scheme:

- “Name_lastName_roboCupHome_tutorial3.zip”

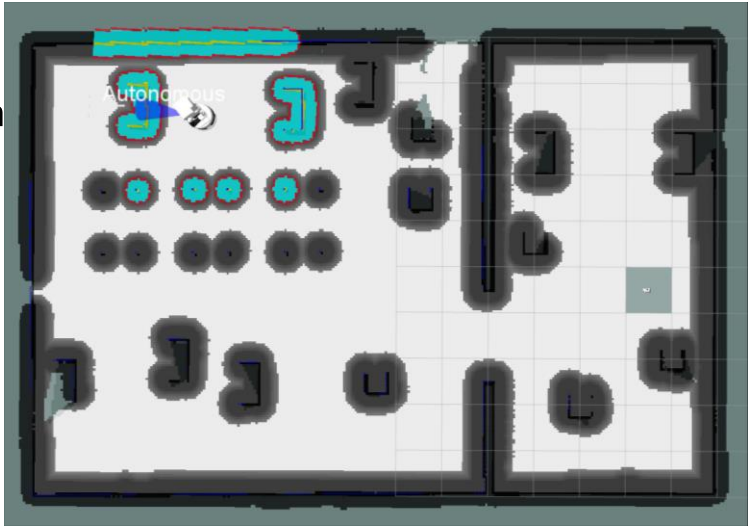
Send the files to: robocup.atHome.ics@gmail.com

Exercises

Task 1: Generate a Map

Generate two maps using Tiago:

- *"small_office"* which is the default world for Tiago navigation
- *"tutorial_office"* which is contained in the tiago_gazebo package



Task 1 Deliverables:

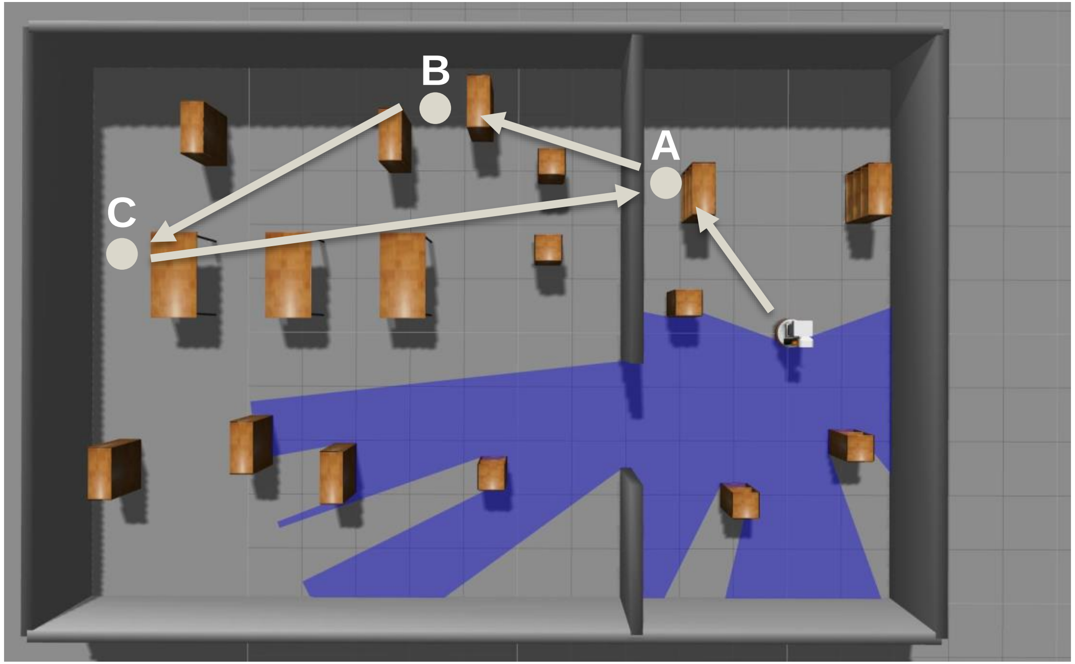
Create one subfolder *"task_1"* containing the following files:

- *map.pgm*
- *map.yaml*
- *mmap.yaml*
- *submap_0.pgm*
- *transformation.xml*

Exercises

Task 2: Navigate the Map
Create a package to navigate Tiago in the *“small_office”* world. The robot should continuously and **autonomously** move between points A,B,C and loop as shown in the picture. Try to avoid knocking over stuff. Use the TIAGo and navigation_stack tutorials for reference.

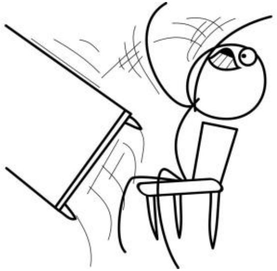
Optional task:
Try to make the localization procedure as robust as you can.

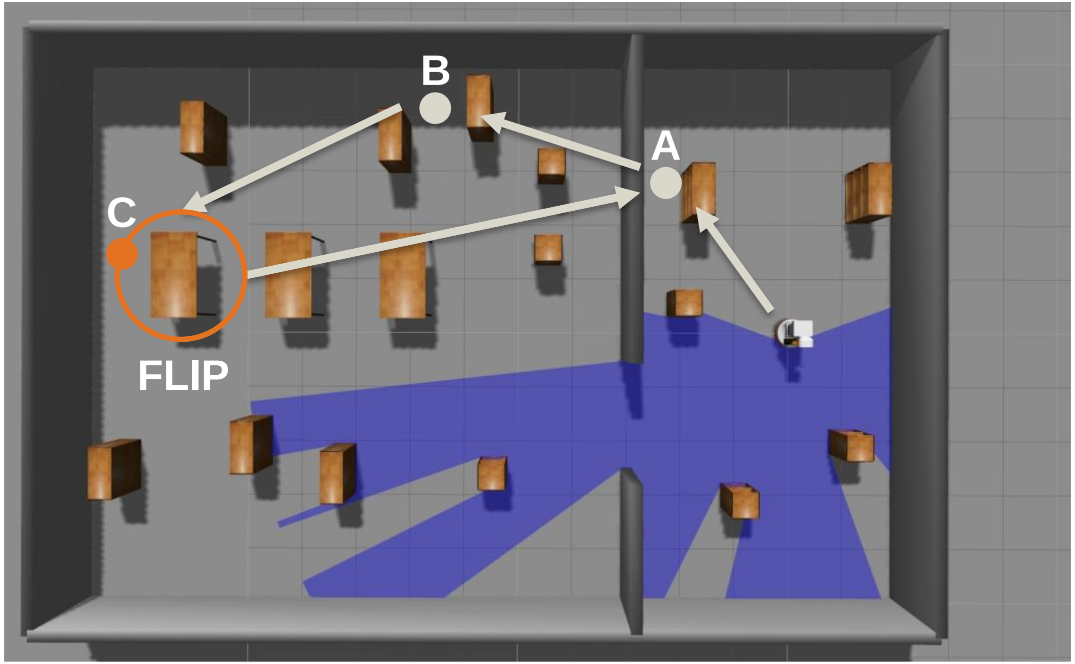


Exercises

Task 3: Flip the Table
Use task 2 as a starting point. The robot should flip over the table at position C using the MoveIt! planning capabilities, before starting the patrol. Try to avoid knocking over the other stuff. Use the TIAGo, navigation_stack and MoveIt! tutorials for reference.

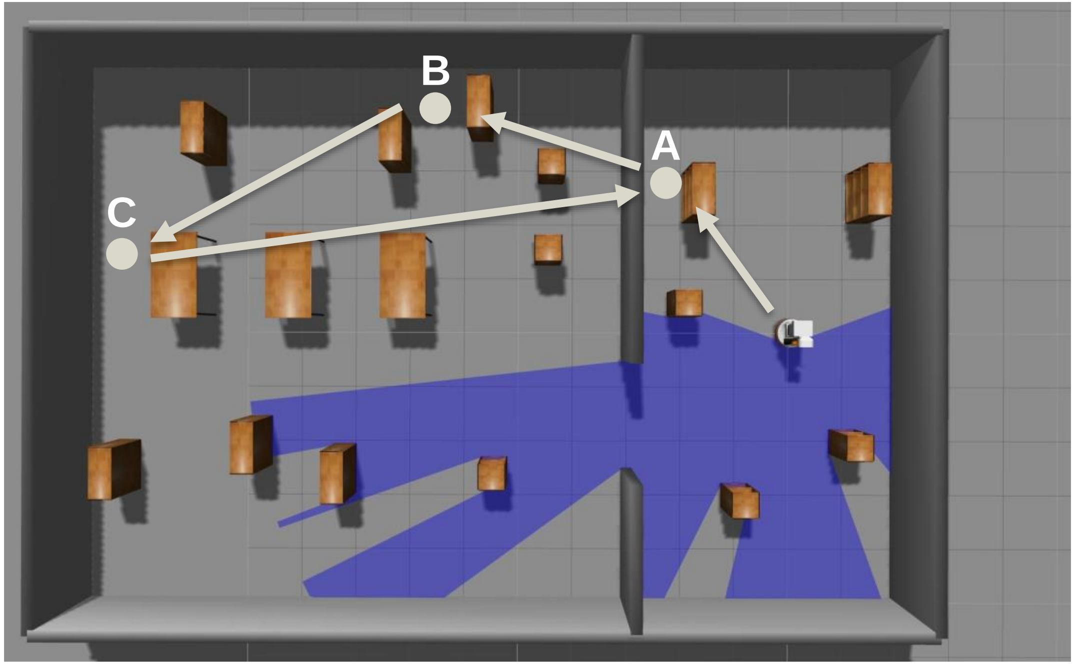
Optional task:
Use a different planner and notice the difference in planning behavior





Exercises

Task 2 and 3 Deliverables:
The package **source code** in *“task_2” and “task_3”. No binaries!*
The packages should contain a .launch file each *“navigation.launch”* which automatically starts the navigation after launching. The c++ code should be written in one .cpp file (plus .h file if required).
Make sure your code compiles and works!
Make sure the .launch file starts everything!
Optional tasks are not graded but may help you with the project.



Further Reading

Mapping:

- Grisetti, Giorgio, Cyrill Stachniss, and Wolfram Burgard. "Improved techniques for grid mapping with rao-blackwellized particle filters." *IEEE transactions on Robotics* 23.1 (2007): 34-46.

Localization:

- Fox, Dieter, et al. "Monte carlo localization: Efficient position estimation for mobile robots." *AAAI/IAAI* 1999.343-349 (1999): 2-2.

SLAM:

- Thrun, Sebastian, and John J. Leonard. "Simultaneous localization and mapping." *Springer handbook of robotics*. Springer Berlin Heidelberg, 2008. 871-889.

Motion Planning:

- LaValle, Steven M. *Planning algorithms*. Cambridge university press, 2006.