



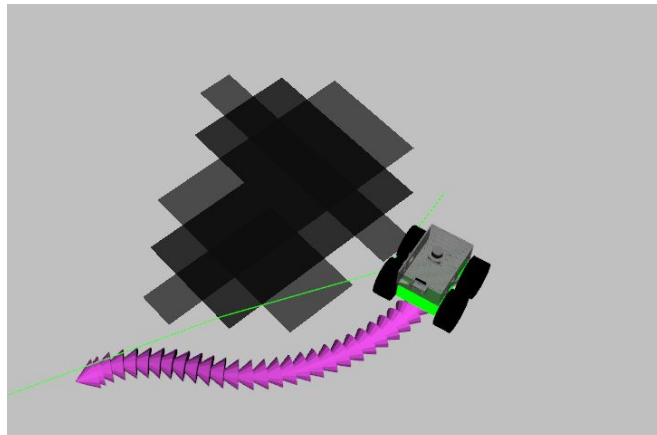
Tutorial on Path Planning

ETH Robotics Summer School

Nick Lawrance and [Luca Bartolomei](#)
5th July, 2022

Tutorial Objectives

- (Brief) introduction to Path Planning
- Description of path-planning pipeline:
 - Global Planning
 - Local Planning
- How to install and run the packages



(Brief) introduction to Path Planning

Path planning:

- Autonomous goal-oriented navigation
- Obstacle avoidance

ETH zürich

A Fully-Integrated Sensing and Control System for
High-Accuracy Mobile Robotic Building Construction

A. Gawel, H. Blum, J. Pankert, K. Krämer,
L. Bartolomei, S. Ercan, F. Farshidian, M. Chli,
F. Gramazio, R. Siegwart, M. Hutter and T. Sandy

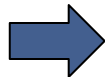
IROS 2019



(Brief) introduction to Path Planning

- Hierarchical architecture

Global Planner

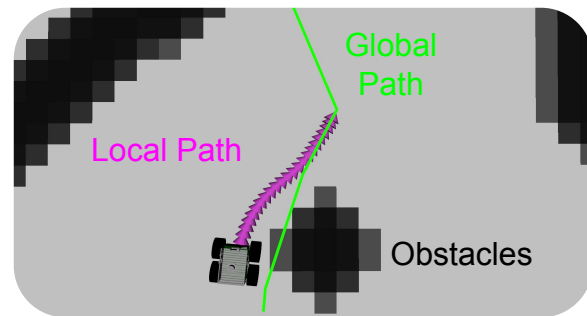


- Coarse path to goal configuration
- Optimistic (unknown space is free)
- Low re-planning rate

Local Planner

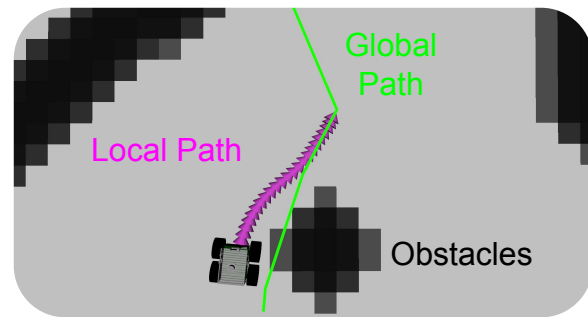
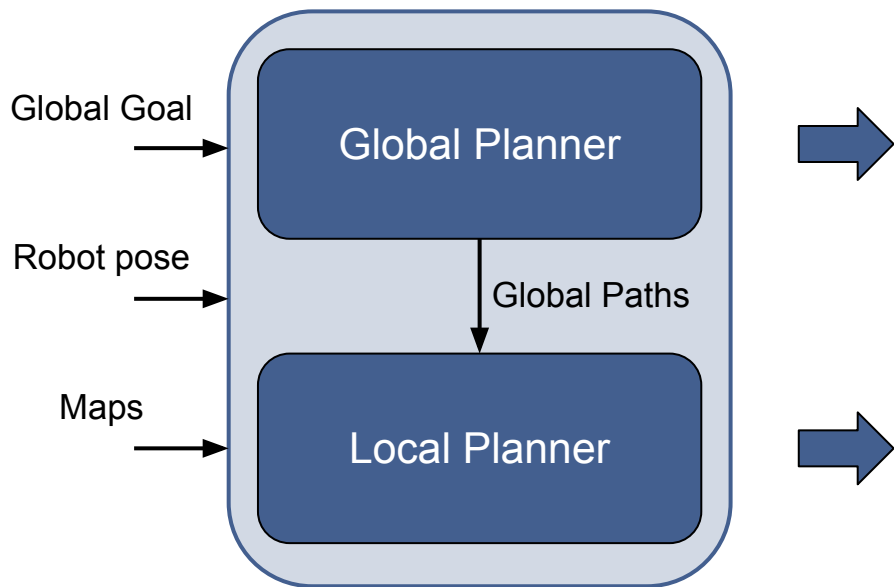


- Local obstacle avoidance
- Feasible trajectories for controllers
- High re-planning rate



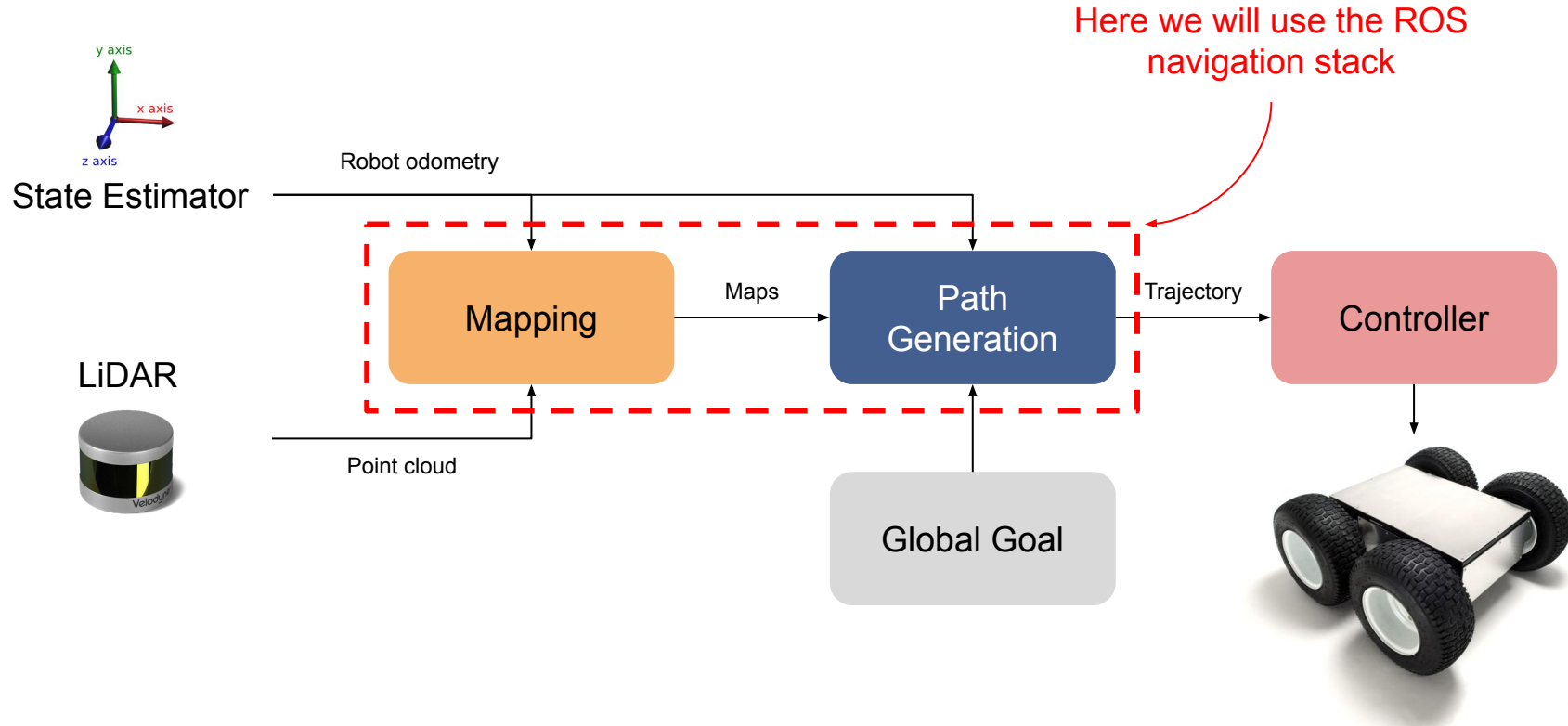
(Brief) introduction to Path Planning

- Hierarchical architecture



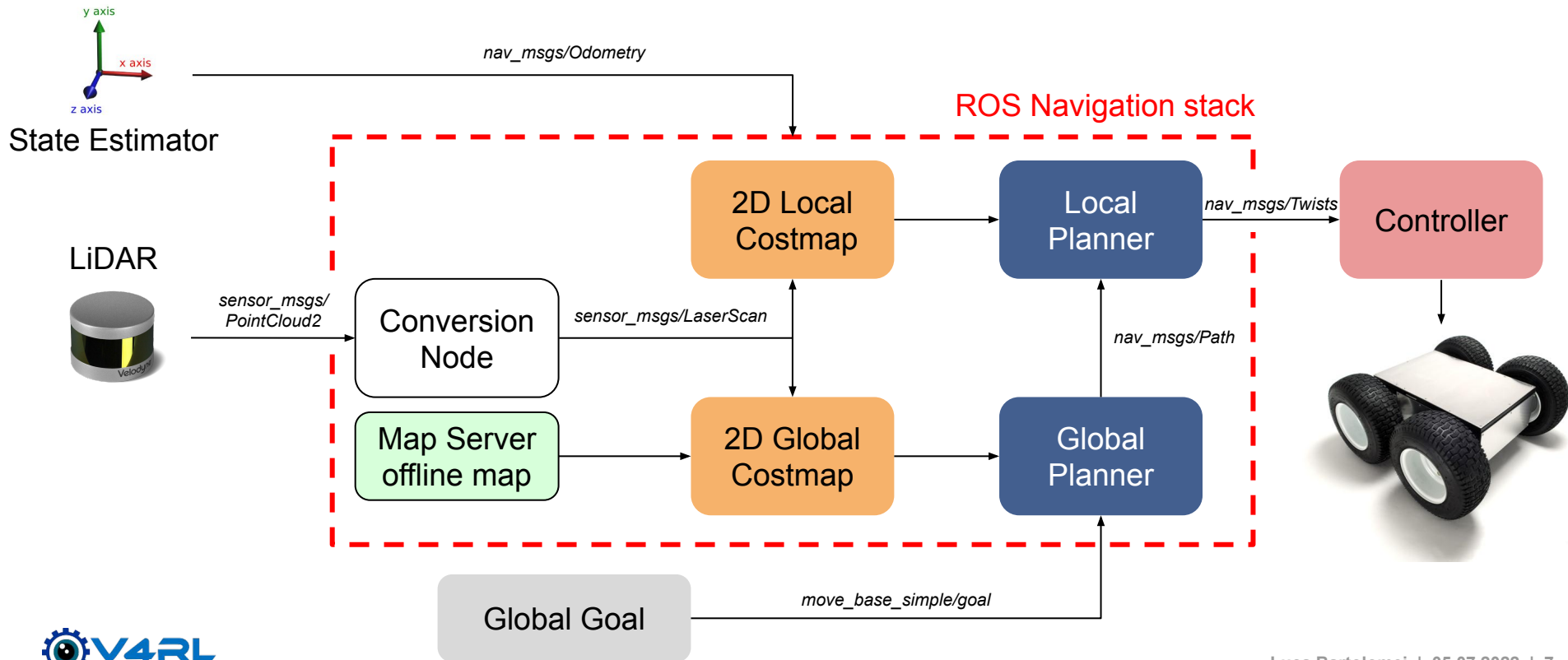
- Coarse path to goal configuration
 - Optimistic (unknown space is free)
 - Low re-planning rate
-
- Local obstacle avoidance
 - Feasible trajectories for controllers
 - High re-planning rate

Pipeline Overview



Detailed Pipeline

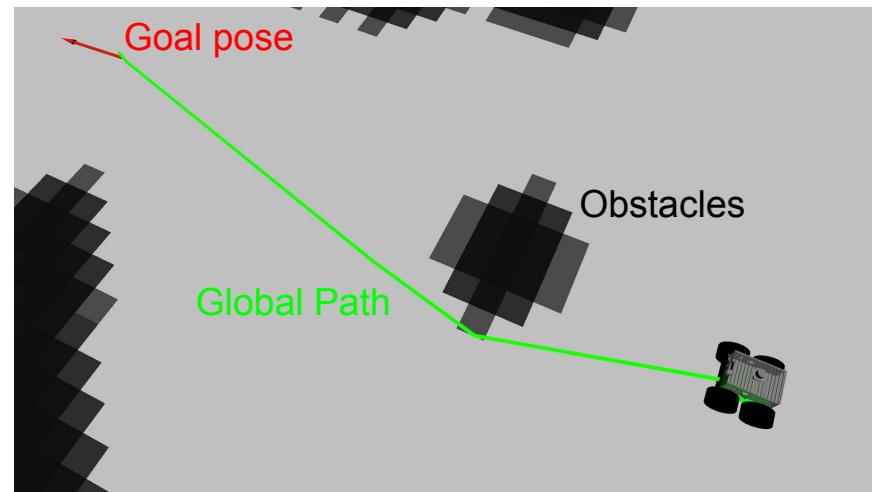
Docs: [move_base - ROS Wiki](#)



Path Planner: Global Planner

Docs: [move_base - ROS Wiki](#)

- Path from current state to global goal
- Optimistic \rightarrow unknown space = free



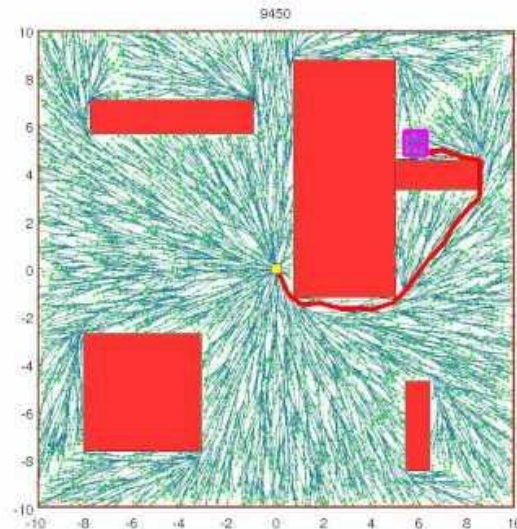
➡ **OMPL** (Open Motion Planning Library) - <https://ompl.kavrakilab.org>

Path Planner: Global Planner

Docs: [move_base - ROS Wiki](#)

- Custom implementation based on OMPL
 - Code:
`smb_path_planner/smb_ompl_planner`
- Basic version: RRT*
 - Probabilistically complete and optimal algorithm

RRT*



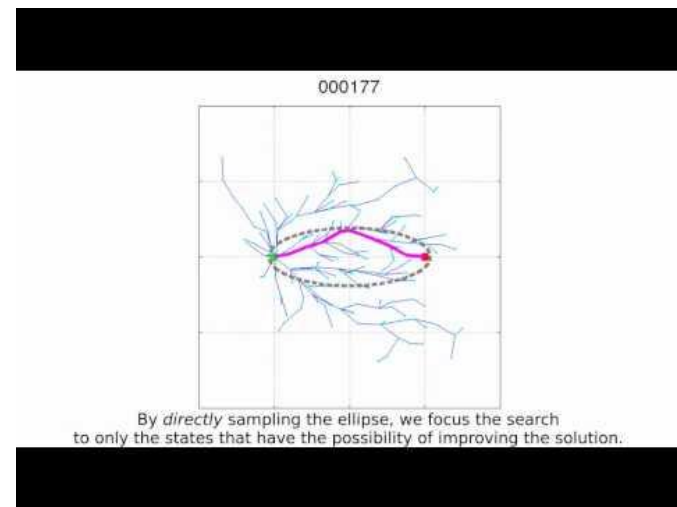
*"Sampling-based Algorithms for Optimal Motion Planning",
S. Karaman and E. Frazzoli, IJRR 2011*

Path Planner: Global Planner

Docs: [move_base - ROS Wiki](#)

- Custom implementation based on OMPL
 - Code:
`smb_path_planner/smb_ompl_planner`
- Basic version: RRT*
 - Probabilistically complete and optimal algorithm
- Possibility to swap solver from config file:
 - Informed RRT*
 - RRT#
 - PRM

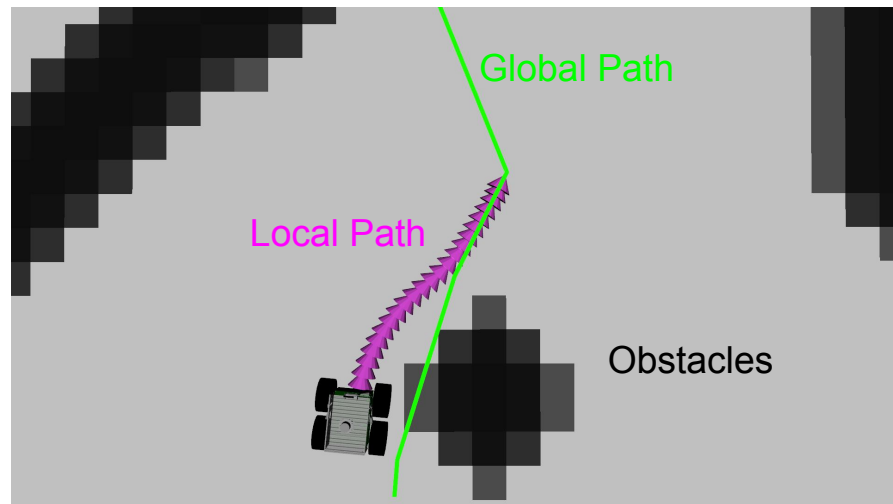
Informed RRT*



"Informed RRT: Optimal Incremental Path Planning Focused through an Admissible Ellipsoidal Heuristic", J.D. Gammell, IROS 2014*

Path Planner: TEB Local Planner

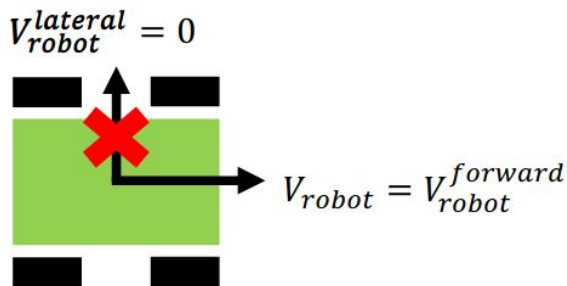
- Compute locally optimal paths
 - Vehicle dynamics
 - Obstacle avoidance
- Pessimistic
→ unknown space = occupied
- Use Time-Elastic-Band (TEB) Local Planner
 - C. Rösmann et al., IROS 2017
 - Docs: [teb_local_planner - ROS Wiki](https://wiki.ros.org/teb_local_planner)



Path Planner: TEB Local Planner

Docs: [teb_local_planner - ROS Wiki](#)
[Link to TEB paper](#)

- Online trajectory optimization
- Objective: reach goal in minimum time
 - Kinodynamic constraints (velocity, acceleration, ...)
 - Non-holonomic kinematics



[Link to video](#)

Path Planner: TEB Local Planner

Docs: [teb_local_planner - ROS Wiki](#)

[Link to TEB paper](#)

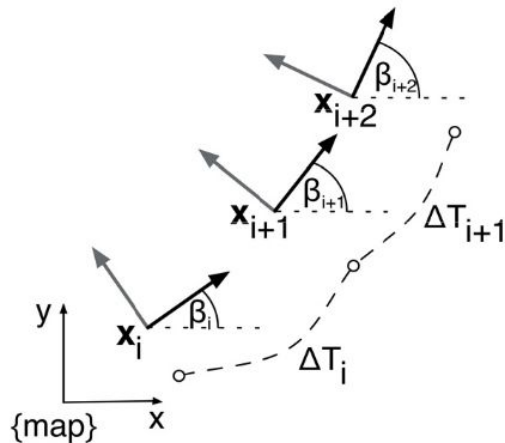
- Trajectory parametrized as a rubber band

- Waypoints:

$$\mathbf{x}_i = (x_i, y_i, \beta_i)^T \in \mathbb{R}^2 \times S^1$$

- Timing between waypoints:

$$\tau = \{\Delta T_i\}_{i=0 \dots n-1}$$



- Optimize cost function:

$$f(B) = \sum_k \gamma_k f_k(B)$$

Function to be optimized

Waypoints and timing sequences

Weighted sum of

- Objectives (e.g. shortest path), and
- Soft constraints (e.g. kinematics, dynamics)

Installation & Simulation Set-up

- Main webpage: https://github.com/ETHZ-RobotX/smb_path_planner

The screenshot shows the GitHub repository page for `ETHZ-RobotX/smb_path_planner`. The repository is public and has 19 unwatched items, 18 forks, and 59 stars. The main branch is `master`, with 6 branches and 2 tags. The repository description is "Repository for the path planner for SMB for the ETH Robotics Summer School". The repository includes a README.md, a LICENSE, and a .github/workflows directory. The repository is licensed under BSD-3-Clause. The repository has 19 watchers, 18 forks, and 59 stars. The repository is categorized under robotics, path-planning, and summer-school. The repository has 1 release, SummerSchool2021, published on Jul 14, 2021. The repository has no packages published. The repository has 7 contributors. The repository has a language distribution of 84.2% C++ and 10.2% Python.

Repository: ETHZ-RobotX/smb_path_planner (Public)

Navigation: Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, Settings

Branches: master (6 branches, 2 tags)

Commits: mantelt Updating link in README.md (291fec2, 3 hours ago, 220 commits)

Files:

- .github/workflows: Add github action (2 months ago)
- docs: Add pdf of path planning tutorial slides (11 months ago)
- smb_navigation: Fix missing dependencies (3 days ago)
- smb_navigation_rviz: Remove catkin simple dependency from rviz plugin (12 months ago)
- smb_navigation_scripts: Expose the arguments for follow-waypoint use case to the top launch f... (12 months ago)
- smb_ompl_planner: Fix parameter setting for global planner occupancy map size: modify p... (12 months ago)
- smb_path_planner: Add package with utilities to test with mabi (2 years ago)
- traversability_layer: add missing roslint dependency (3 months ago)
- LICENSE: Update LICENSE (7 months ago)
- README.md: Updating link in README.md (3 hours ago)

About: Repository for the path planner for SMB for the ETH Robotics Summer School

Readme: BSD-3-Clause license, 59 stars, 19 watching, 18 forks

Releases: 1 release: SummerSchool2021 (Latest) on Jul 14, 2021

Packages: No packages published. Publish your first package.

Contributors: 7 contributors

Languages: C++ 84.2%, Python 10.2%

README.md:

Super Mega Bot Path Planner

Package for path planning of Super Mega Bots for the ETH Robotics Summer School. The package has been tested under ROS Noetic and Ubuntu 20.04.

Author: Luca Bartolomei
Affiliation: Vision For Robotics Lab, ETH Zurich

Installation & Simulation Set-up

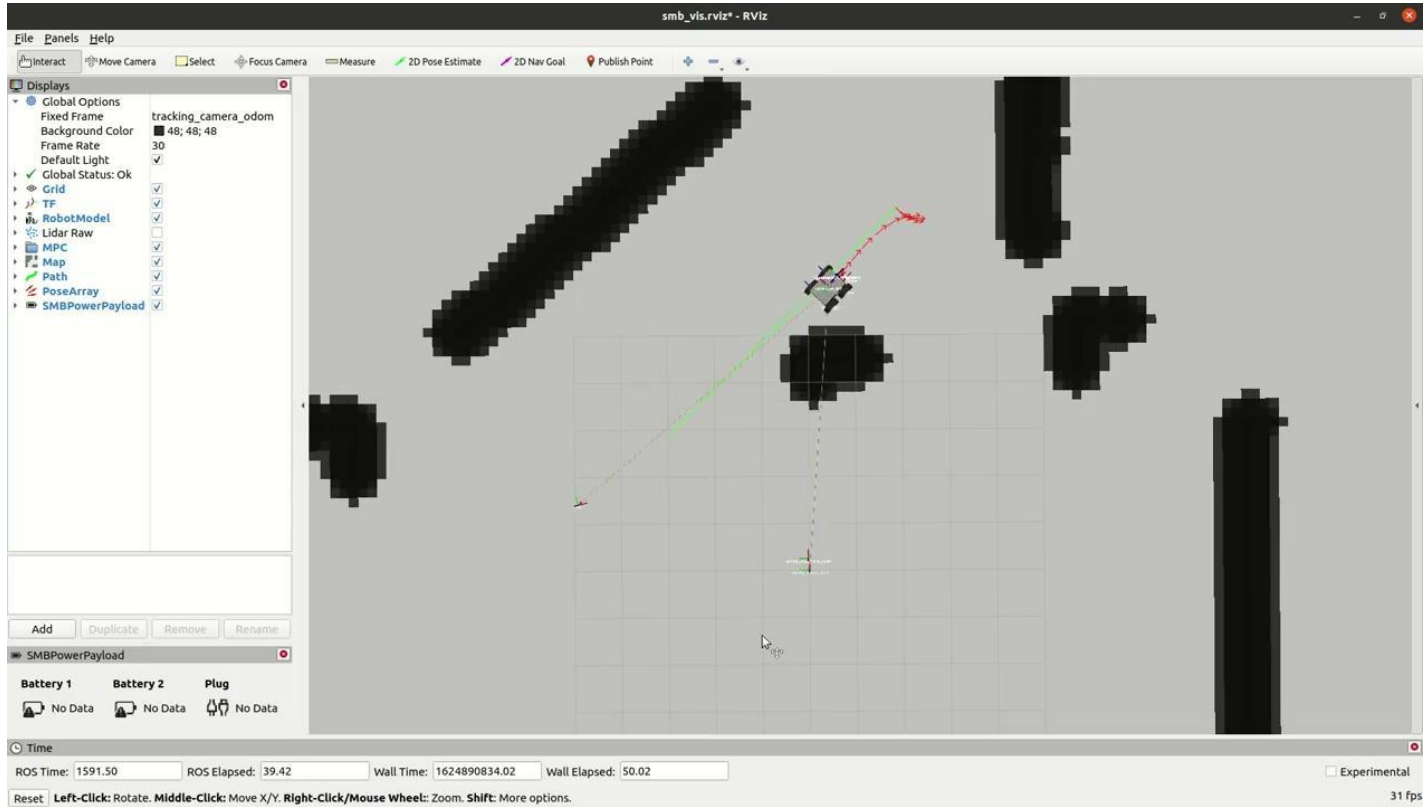
- Main webpage: https://github.com/ETHZ-RobotX/smb_path_planner
- Simply follow the instructions to install the planner

Main Packages	Description
<code>smb_navigation</code>	Main package (utilities, launch and configuration files)
<code>smb_navigation_scripts</code>	Utility scripts (follow waypoints, point cloud processing)
<code>smb_ompl_planner</code>	Global planner based on OMPL library
<code>smb_navigation_rviz</code>	RViz plugin to select the navigation goal

Running in Simulation - basic usage

- Refer to the documentation
- Quick start:
 1. Start the simulation: `$ roslaunch smb_gazebo sim.launch`
 - a. You should be able to see the SMB in RViz
 2. Start the planner:
`$ roslaunch smb_navigation navigate2d_ompl.launch sim:=true`
`global_frame:=tracking_camera_odom`
 - a. You should see the occupancy map in RViz
 3. Send goal position using "2D Nav Goal" button in RViz

Running in Simulation - basic usage



Running in Simulation - basic usage

- Refer to the documentation

- Run `rqt_reconfigure` to tune the planner online (remember to save the parameters!):

```
$ rosrun rqt_reconfigure rqt_reconfigure
```

- Play with the different global planners

- Change OMPL planner by editing config file:

```
smb_navigation/config/ompl_global_planner.yaml
```

- Run `move_base` global planner (using A*):

```
$ roslaunch smb_navigation navigate2d.launch sim:=true  
global_frame:=tracking_camera_odom
```

4. Enjoy!

Running in Simulation - advanced features

- Refer to the [documentation](#) for advanced features

- It is possible to specify...

- ... a different odometry topic:

```
$ roslaunch smb_navigation navigate2d_ompl.launch odom_topic:=/odom_new
```

- ... different reference frames:

```
$ roslaunch smb_navigation navigate2d_ompl.launch global_frame:=map_new  
robot_base_frame:=base_new
```

Running in Simulation - advanced features

- Refer to the documentation for advanced features

- It is possible to specify...

- ... a different odometry topic:

```
$ roslaunch smb_navigation navigate2d_ompl.launch odom_topic:=/odom_new
```

- ... different reference frames:

```
$ roslaunch smb_navigation navigate2d_ompl.launch global_frame:=map_new  
robot_base_frame:=base_new
```

- Use existing global maps for planning (e.g. from previous missions)
- Follow a set of waypoints

Running in Simulation - advanced features

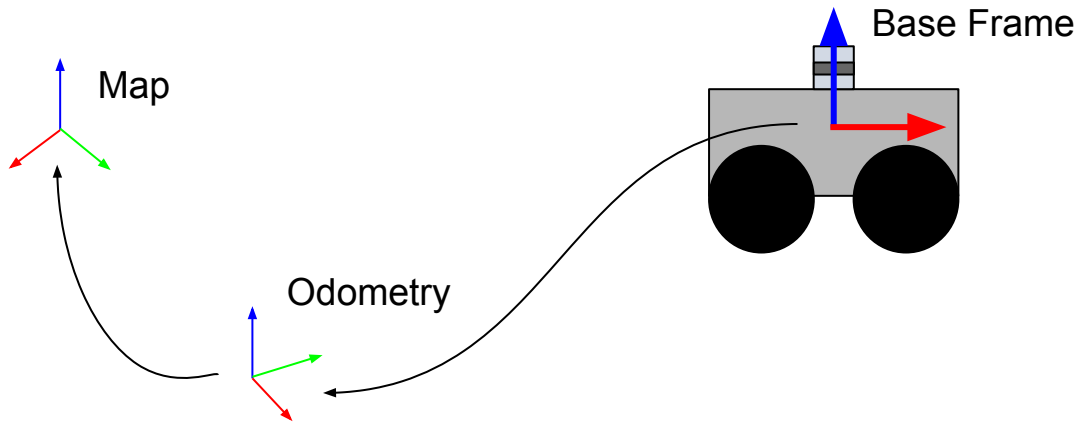
- Refer to the [documentation](#) for advanced features
- It is possible to specify...
 - ... a different odometry topic:

```
roslaunch smb_navigation navigate2d_ompl.launch odom_topic:=/odom_new
```
 - ... different reference frames:

```
roslaunch smb_navigation navigate2d_ompl.launch global_frame:=map_new  
robot_base_frame:=base_new
```
- Use existing global maps for planning (e.g. from previous missions)
- Follow a set of waypoints

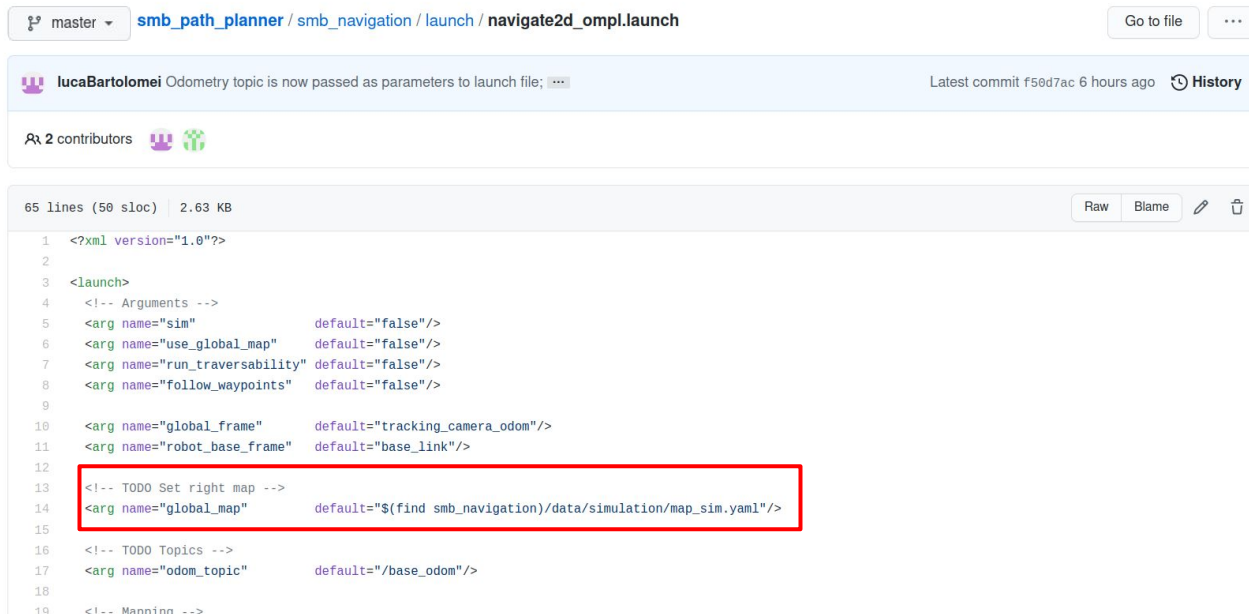
Running in Simulation - use existing global maps (doc)

- Global planning → static and globally consistent map
 - Finds the shortest path in the complete map (*Map* frame)
 - Uses TF connections to retrieve transformations (from *Base Frame* to *Map*)
- Local planning & control → *Odometry* frame
 - Use odometry information directly (drifting, but continuous estimates)



Running in Simulation - use existing global maps (doc)

- For this tutorial, we provide a map of the simulation environment
- Follow these steps:
 - Set the path to the global map in the launch file (already done for simulation)



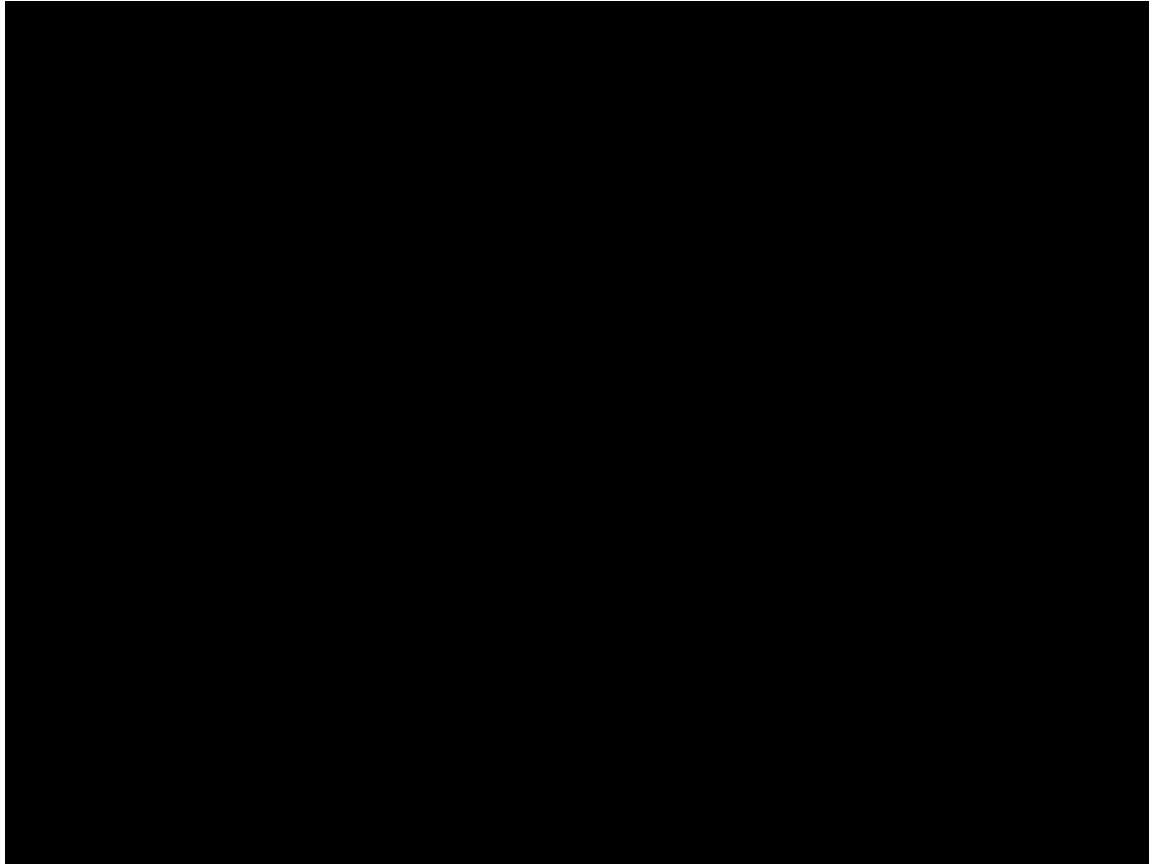
```
1 <?xml version="1.0"?>
2
3 <launch>
4   <!-- Arguments -->
5   <arg name="sim" default="false"/>
6   <arg name="use_global_map" default="false"/>
7   <arg name="run_traversability" default="false"/>
8   <arg name="follow_waypoints" default="false"/>
9
10  <arg name="global_frame" default="tracking_camera_odom"/>
11  <arg name="robot_base_frame" default="base_link"/>
12
13  <!-- TODO Set right map -->
14  <arg name="global_map" default="$(find smb_navigation)/data/simulation/map_sim.yaml"/>
15
16  <!-- TODO Topics -->
17  <arg name="odom_topic" default="/base_odom"/>
18
19  <!-- Mappings -->
```

Running in Simulation - use existing global maps (doc)

- For this tutorial, we provide a map of the simulation environment
- Follow these steps:
 - Set the path to the global map in the launch file (already done for simulation)
 - Run the planner:

```
$ roslaunch smb_navigation navigate2d_ompl.launch sim:=true  
global_frame:=tracking_camera_odom use_global_map:=true
```
- The global planner now uses a **static global map**
 - Use `rqt_reconfigure` to turn on the obstacle and inflation layers

Running in Simulation - use existing global maps (doc)

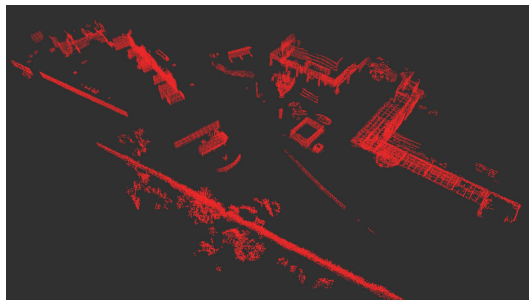


Running in Simulation - create global maps (doc)

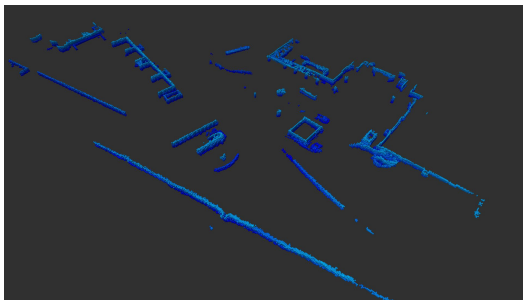
- But how do I create a global map (e.g. from SLAM)?

Running in Simulation - create global maps (doc)

- But how do I create a global map (e.g. from SLAM)?
- These instructions show how to create an occupancy map from * .pcd files



Original * .pcd file
(3D map)



OctoMap
(3D map)



Occupancy Grid
(2D map)

Running in Simulation - create global maps (doc)

Can take several minutes of computation!

- If you have a `compslam_map.pcd` file created by `smb_slam`, run default script:
 - Start `roscore` on your machine
 - `$ cd smb_path_planner/smb_navigation/script`
 - `$ python3 pcd_to_grid_map_default_paths.py`
- Using *default parameters* (`resolution`, `z_min`, `z_max`) defined in `pcd_to_gridmap.sh`!
 - Inspect the intermediate results in RViz
- This will create a `*.yaml` and a `*.pgm` files in `smb_navigation/data/test`
 - Make sure the origin in the `*.yaml` file does not contain NaNs
→ Otherwise, replace them with zeros
 - Check that the path to `*.pgm` file in `*.yaml` file is correct (relative paths are ok)

Manual work!

Running in Simulation - create global maps (doc)

Can take several minutes of computation!

- If you want more control, use the script directly:

```
$ cd smb_path_planner/smb_navigation/script
$ chmod +x pcd_to_gridmap.sh
$ ./pcd_to_gridmap.sh <input_file> <output_folder> <run_rviz>
```

True or False



- Using *default parameters* (resolution, z_min, z_max) defined in pcd_to_gridmap.sh!
 - Inspect the intermediate results in RViz
- This will create a *.yaml and a *.pgm files in the specified output folder
 - Make sure the origin in the *.yaml file does not contain NaNs
→ Otherwise, replace them with zeros
 - Check that the path to *.pgm file in *.yaml file is correct (relative paths are ok)

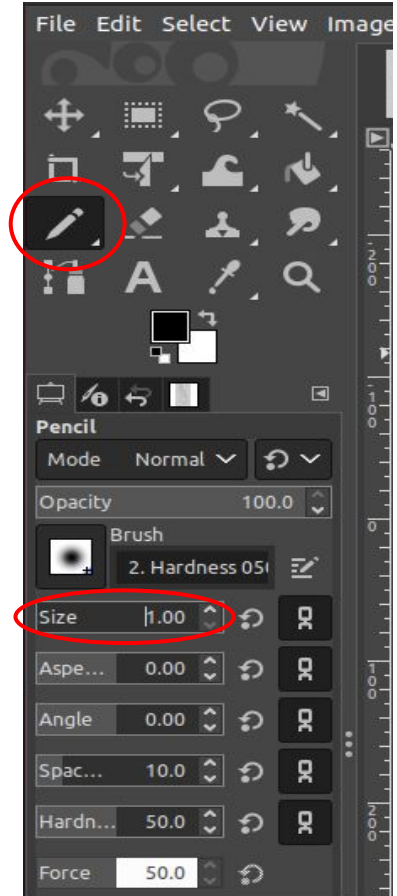
Manual work!

Running in Simulation - create global maps (doc)

- `pcd_to_gridmap.sh` uses default parameters
 - `resolution` → resolution of the grid map
 - `z_min`
 - `z_max` } Height range of the point cloud
- You will need to adjust these parameters to get a good result!
- **Method 1:** trial-and-error
 - Adjust the parameters and inspect results in RViz
 - Repeat until you get a reasonable result
- **Method 2:** use image editing tools
 - Correct the image to remove artifacts

Running in Simulation - create global maps (doc)

- **Method 2:** use image editing tools
 - Correct the image to remove artifacts
- Example with GIMP (install with `sudo apt install gimp`)
 1. Start GIMP and open the file `map.pmg`
 2. Select the pencil tool and choose proper size (e.g. 2)
 3. Select the right color and then you can:
 - a. Remove artifacts
 - b. Draw additional “fake” obstacles (e.g. borders)
 4. Save the new file:
 - a. *File* → *Export As* → `.pgm` extension



Running in Simulation - create global maps (doc)



Running in Simulation - advanced features

- Refer to the [documentation](#) for advanced features
- It is possible to specify...
 - ... a different odometry topic:

```
roslaunch smb_navigation navigate2d_ompl.launch odom_topic:=/odom_new
```
 - ... different reference frames:

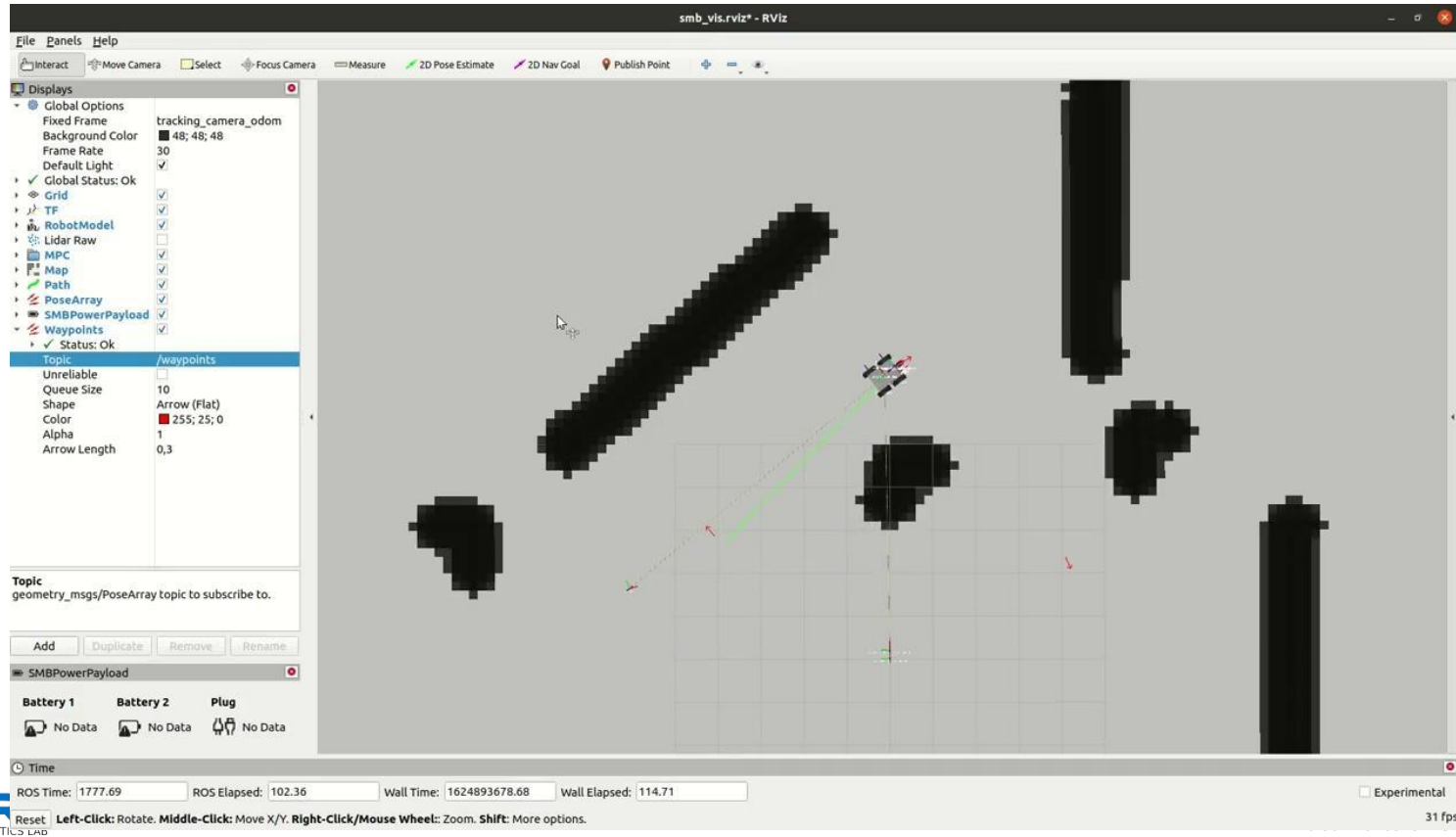
```
roslaunch smb_navigation navigate2d_ompl.launch global_frame:=map_new  
robot_base_frame:=base_new
```
- Use existing global maps for planning (e.g. from previous missions)
- Follow a set of waypoints

Running in Simulation - follow waypoints (doc)

- Start the simulation and run the command:

```
$ roslaunch smb_navigation navigate2d_ompl.launch sim:=true  
global_frame:=tracking_camera_odom follow_waypoints:=true
```

Running in Simulation - follow waypoints (doc)



Running in Simulation - follow waypoints (doc)

- Start the simulation and run the command:

```
$ roslaunch smb_navigation navigate2d_ompl.launch sim:=true  
global_frame:=tracking_camera_odom follow_waypoints:=true
```

- **Online:**

- Start the simulation and the planner
- Set sequence of waypoints in RViz (button “2D Pose Estimate”)
- Call via terminal: `$ rostopic pub /path_ready std_msgs/Empty -1`
- The waypoints will be stored in a file
 - Location specified in the launch file (parameter: `output_folder`)

- **Offline:**

- Specify the input file in the launch file (folder and file name) - make sure it exists!
- Start the simulation and the planner
- Call via terminal: `$ rostopic pub /start_journey std_msgs/Empty -1`

Running on real SMB

- Refer to the documentation

All the advanced features tested in simulation can be used with the real robot as well!

1. Start the robot and the sensors
2. Launch the state estimation and control pipelines
 - a. Make sure everything works correctly (e.g. move the robot around with joypad)

3. Start the planner:

```
$ roslaunch smb_navigation navigate2d_ompl.launch
```

4. Select a goal and start planning

Tutorial Complete!

- Follow the documentation and you should be fine
- Cheat sheet with all the most important commands



References: main configuration files

- Global Cost Map:
[smb_navigation/config/move_base_costmaps/global_costmap_params.yaml](#)
- Local Cost Map:
[smb_navigation/config/move_base_costmaps/local_costmap_params.yaml](#)
- Global Planner:
[smb_navigation/config/move_base_costmaps/ompl_global_planner.yaml](#)
- Local Planner:
[smb_navigation/config/base_local_planner.yaml](#) ([sim](#) / [real](#))

