



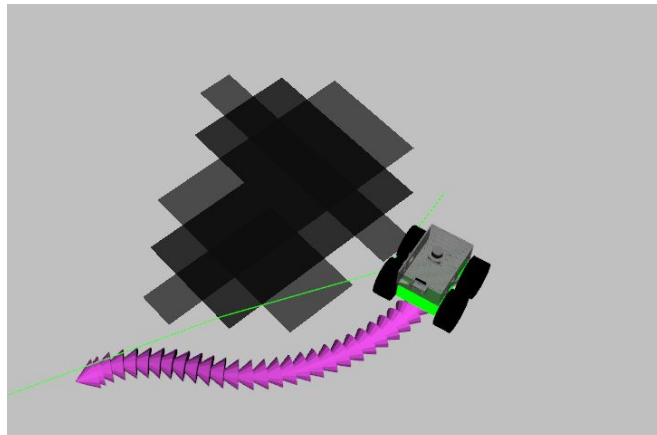
Tutorial on Path Planning

ETH Robotics Summer School

Nick Lawrance and [Luca Bartolomei](#)
6th July, 2021

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

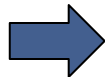
IROS 2019



(Brief) introduction to Path Planning

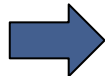
- Hierarchical architecture

Global Planner

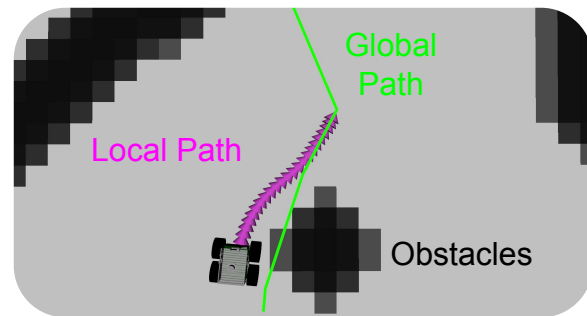


- Coarse path to goal configuration
- Optimistic (unknown space is free)
- Low re-planning rate

Local Planner

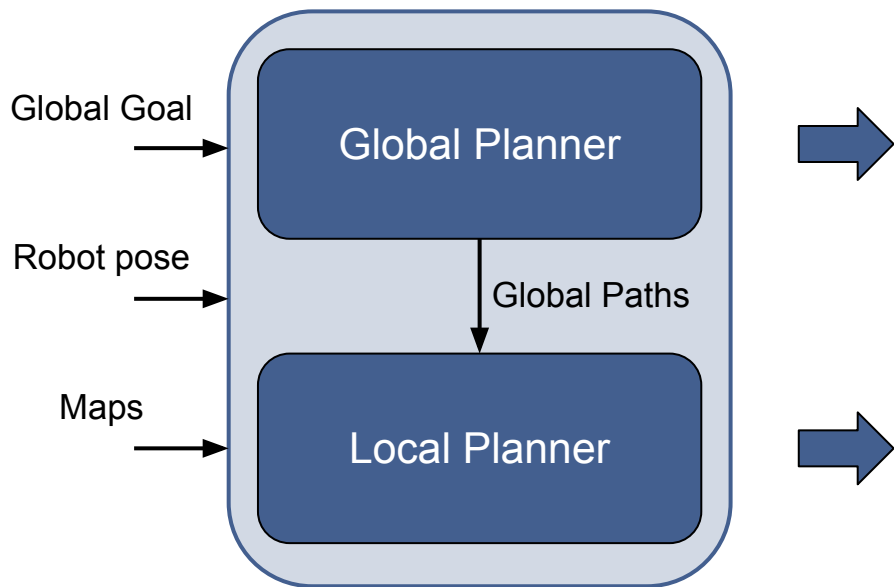


- Local obstacle avoidance
- Feasible trajectories for controllers
- High re-planning rate

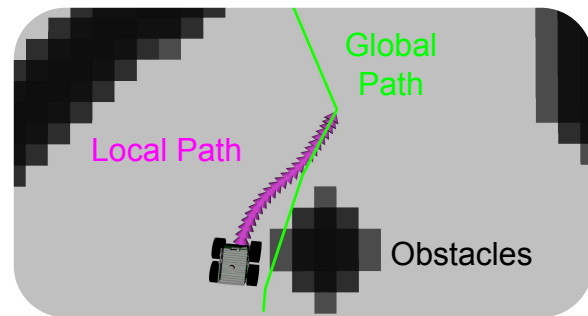


(Brief) introduction to Path Planning

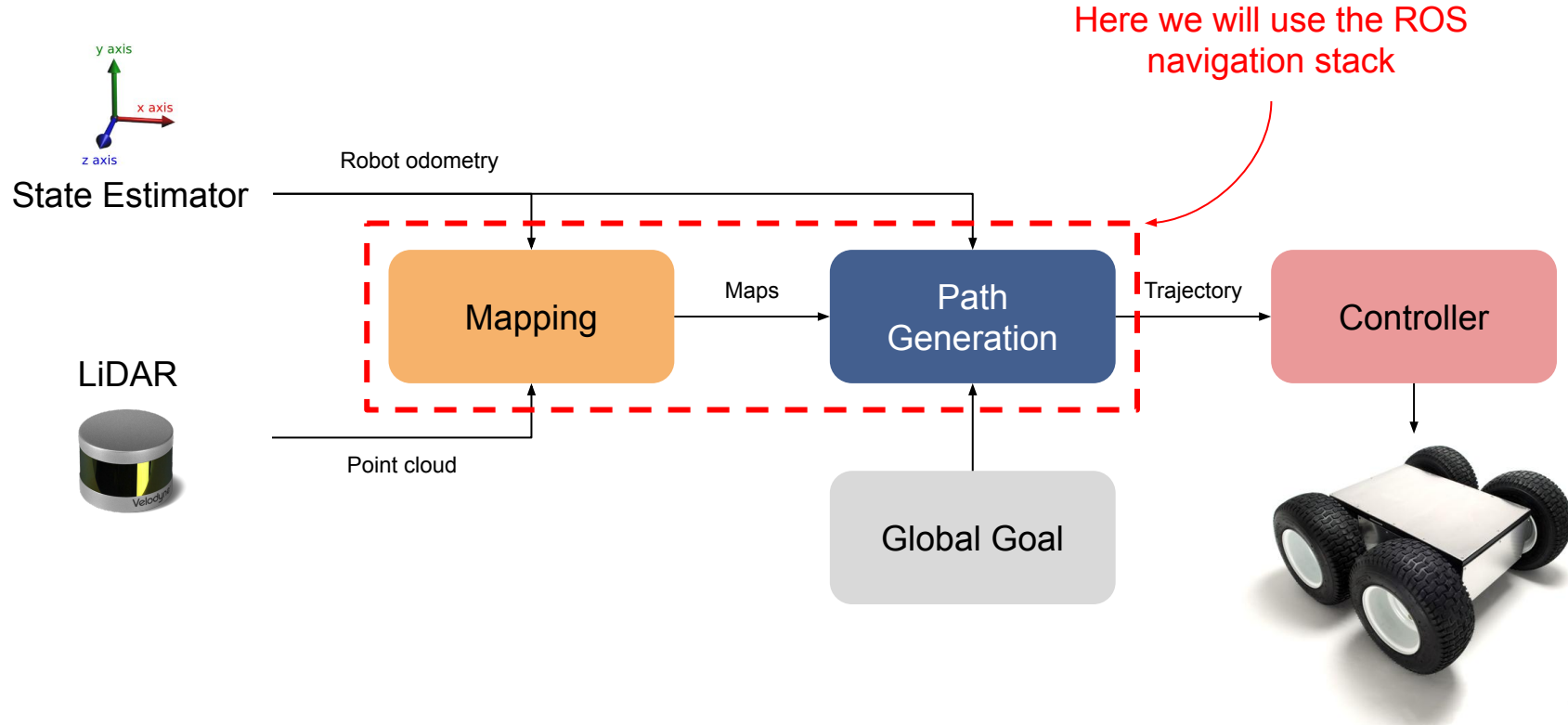
- Hierarchical architecture



- 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

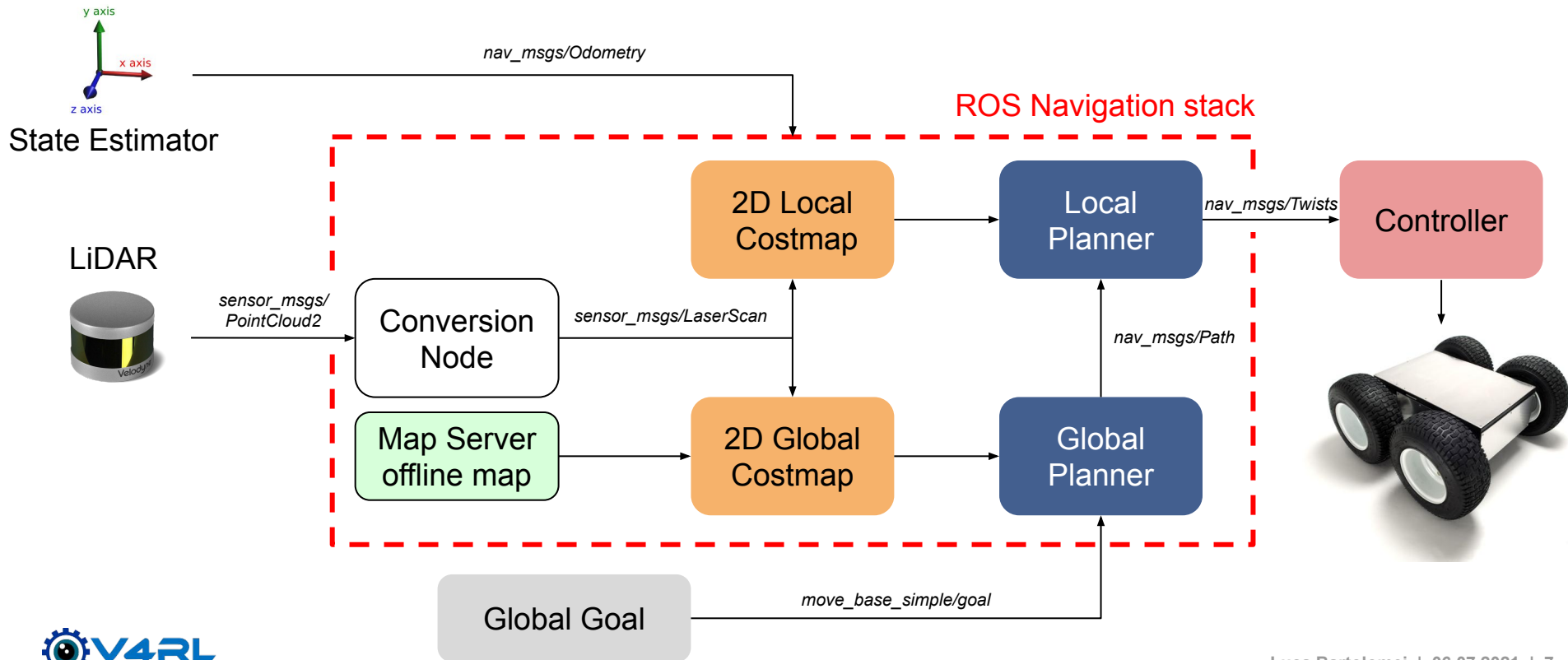


Pipeline Overview



Detailed Pipeline

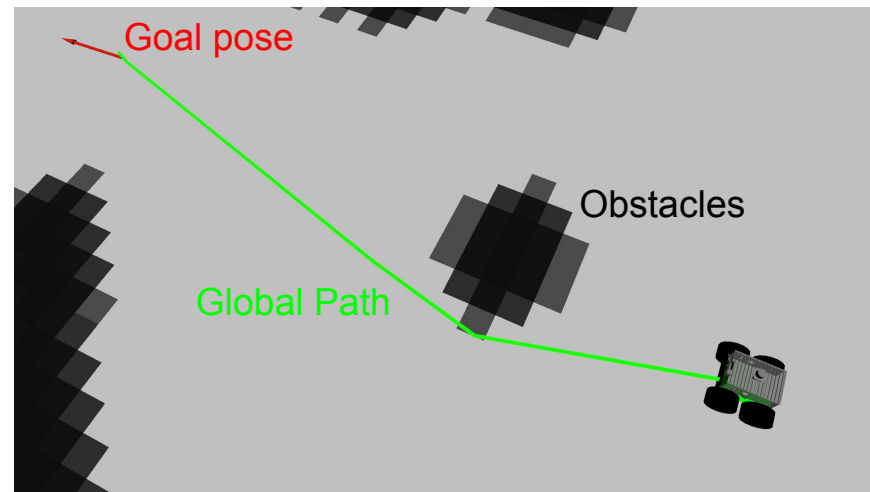
Docs: [move_base - ROS Wiki](#)



Path Planner: Global Planner

Docs: [move_base - ROS Wiki](#)

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



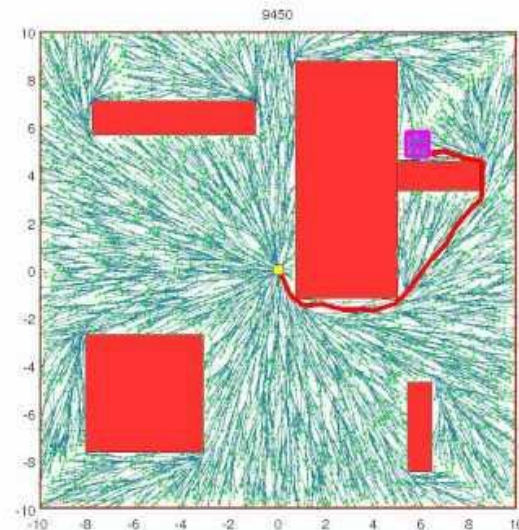
➡ **OMPL** (Open Motion Planning Library) - <https://ompl.kavrakilab.org>

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*



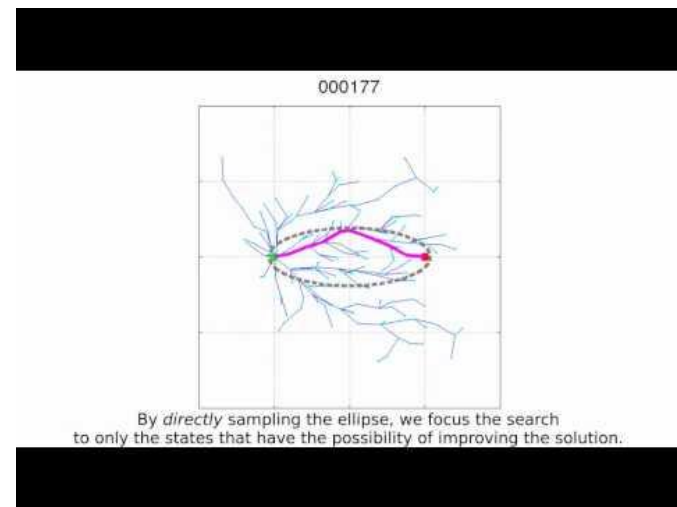
"Sampling-based Algorithms for Optimal Motion Planning",
S. Karaman and E. Frazzoli, IJRR 2011

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

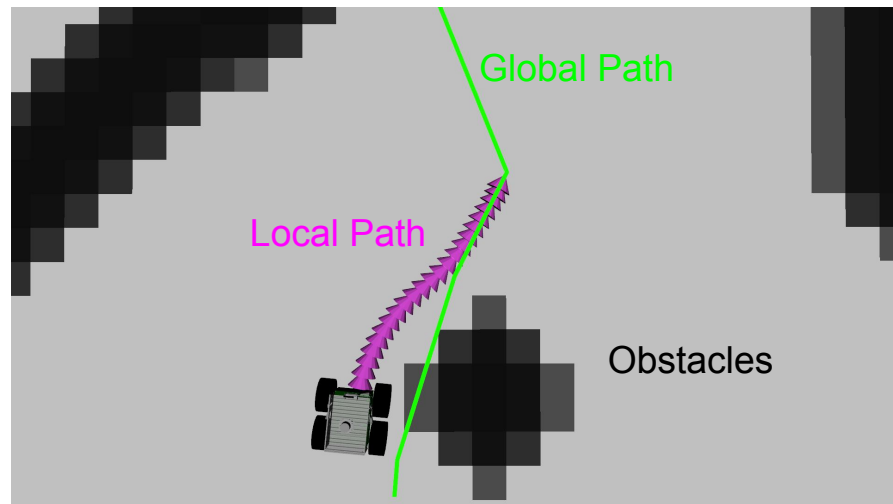
Informed RRT*



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

Path Planner: TEB Local Planner

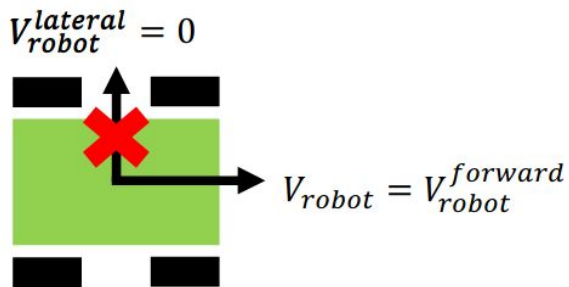
- Compute locally optimal paths
 - Vehicle dynamics
 - Obstacle avoidance
- Pessimistic
→ unknown space = occupied
- Use Time-Elastic-Band (TEB) Local Planner
 - C. Rösmann et al., IROS 2017
 - Docs: [teb_local_planner - ROS Wiki](#)



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](#)

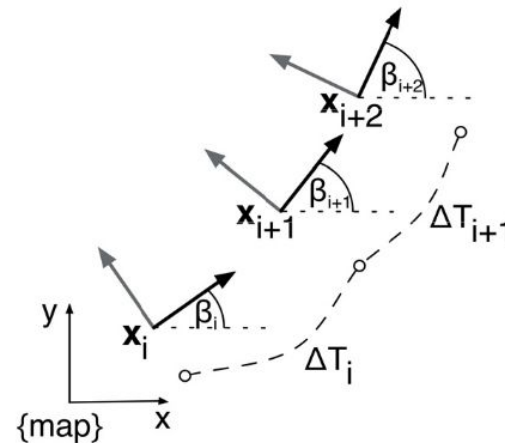
- Trajectory parametrized as a rubber band

- 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/VIS4ROB-lab/smb_path_planner

The screenshot shows the GitHub repository page for `VIS4ROB-lab/smb_path_planner`. The repository has 13 stars, 34 forks, and 9 watchers. It is a public repository with 6 branches and 1 tag. The repository is managed by `lucaBartolomei`, who merged pull request #15 from `VIS4ROB-lab/noetic` 28 days ago. The repository contains several files and folders, including `smb_navigation`, `smb_navigation_rviz`, `smb_navigation_scripts`, `smb_ompl_planner`, `smb_path_planner`, `traversability_layer`, `LICENSE`, and `README.md`. The `README.md` file is expanded, showing the title "Super Mega Bot Path Planner" and a description: "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." The author is Luca Bartolomei, affiliated with Vision For Robotics Lab, ETH Zurich. The contact information is `lucaBartolomei@ethz.ch`. The repository also has a `Readme` tab, a `BSD-3-Clause License`, and a `Releases` section with 1 tag. The `Packages` section shows no published packages. The `Contributors` section lists `lucaBartolomei`, `RobotX-SMB`, and `mantelt`.

Installation & Simulation Set-up

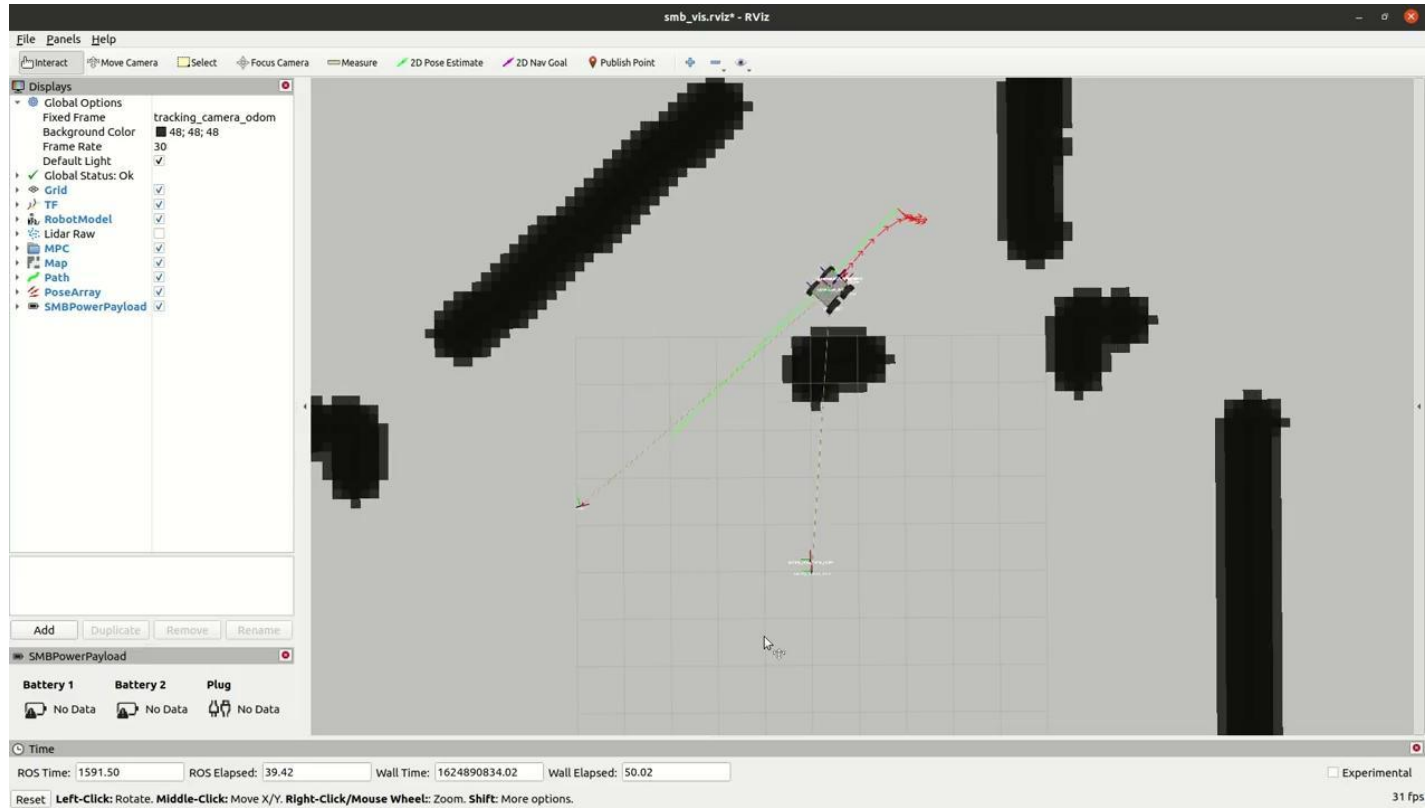
- Main webpage: https://github.com/VIS4ROB-lab/smb_path_planner
- Simply follow the instructions to install the planner

Main Packages	Description
smb_navigation	Main package (utilities, launch and configuration files)
smb_navigation_scripts	Utility scripts (follow waypoints, point cloud processing)
smb_ompl_planner	Global planner based on OMPL library
smb_navigation_rviz	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

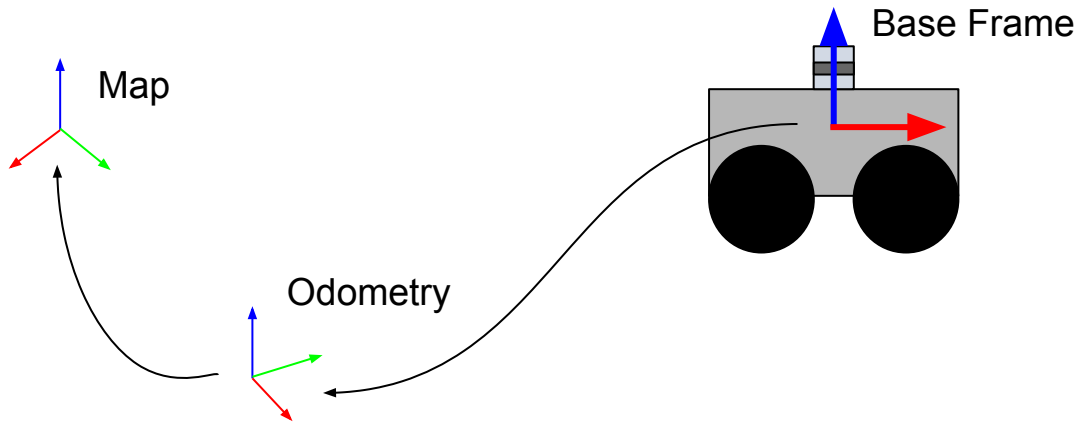
- Refer to the [documentation](#) for advanced features
- It is possible to specify...
 - ... a different odometry topic:

```
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

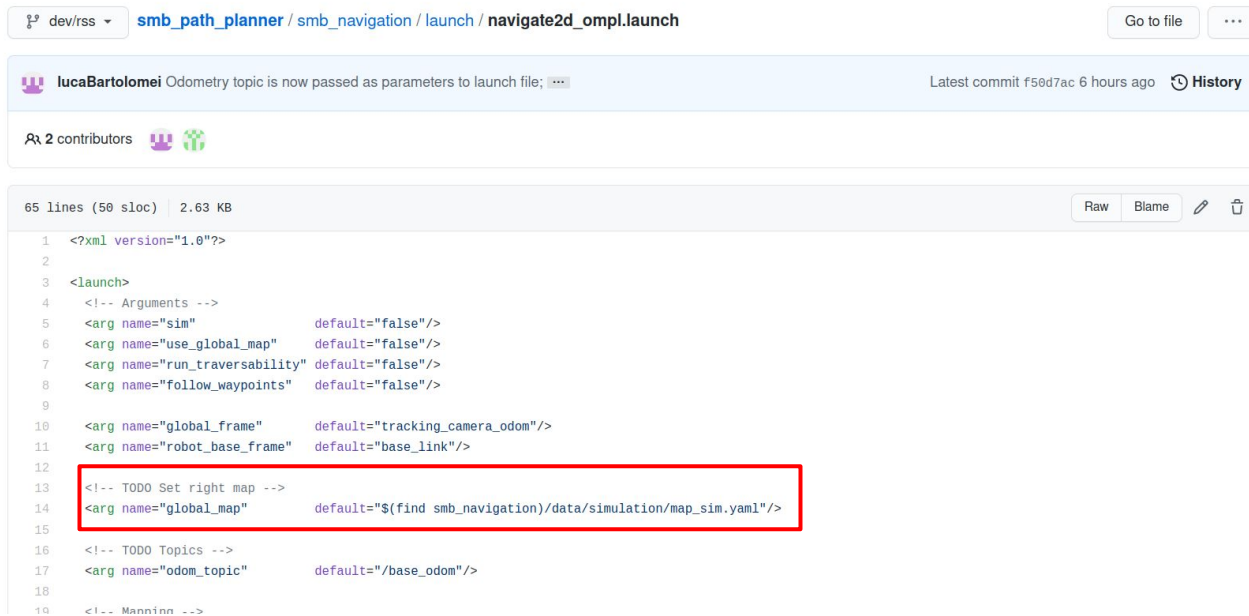
Running in Simulation - use existing global maps (doc)

- 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)



The screenshot shows a code editor interface for the file `smb_path_planner / smb_navigation / launch / navigate2d_ompl.launch`. The editor displays XML code for a ROS launch file. A red rectangle highlights the line `<arg name="global_map" default="$(find smb_navigation)/data/simulation/map_sim.yaml"/>`, which is part of a commented-out section. The code includes arguments for simulation mode, global map usage, traversability, waypoints, global frame, robot base frame, and odometry topic.

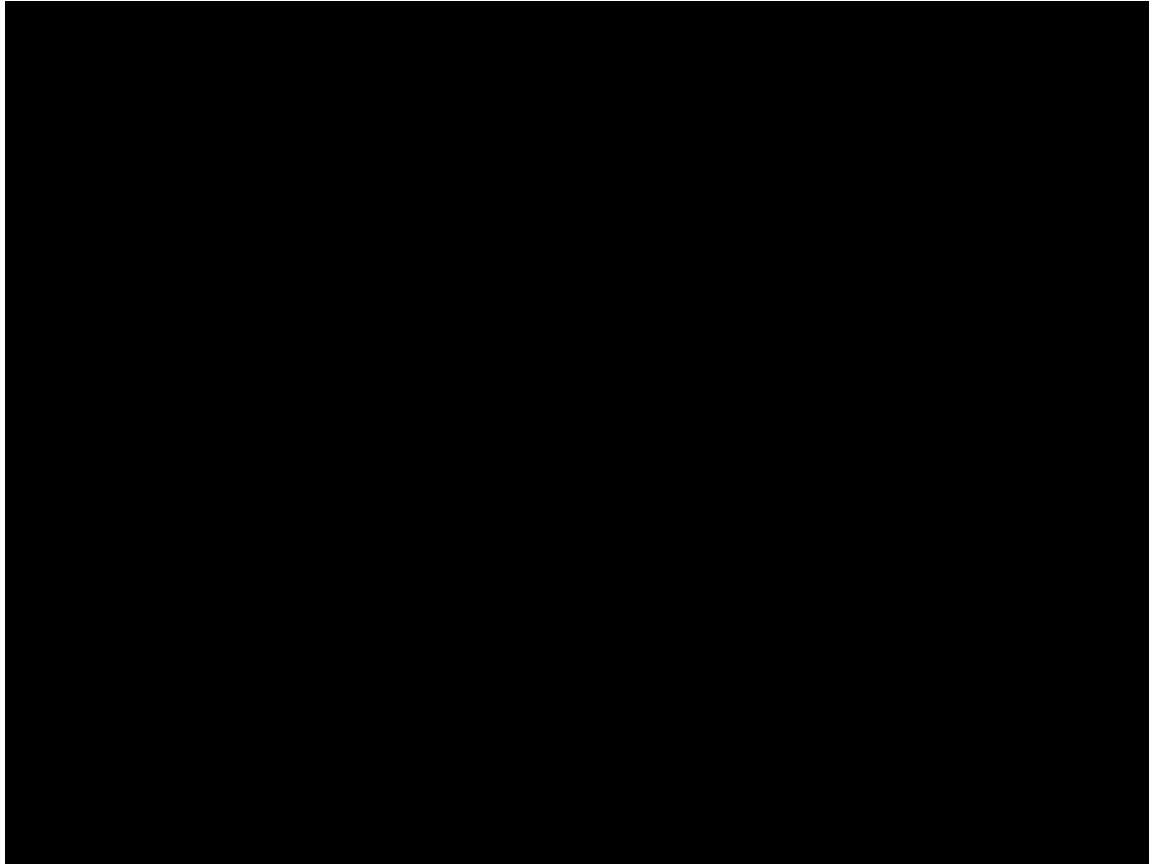
```
1 <?xml version="1.0"?>
2
3 <launch>
4   <!-- Arguments -->
5   <arg name="sim" default="false"/>
6   <arg name="use_global_map" default="false"/>
7   <arg name="run_traversability" default="false"/>
8   <arg name="follow_waypoints" default="false"/>
9
10  <arg name="global_frame" default="tracking_camera_odom"/>
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  <!-- Mapping -->
```

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)

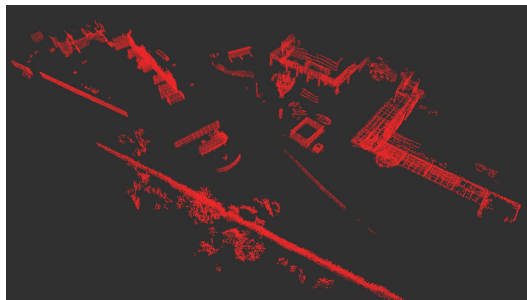


Running in Simulation - create global maps (doc)

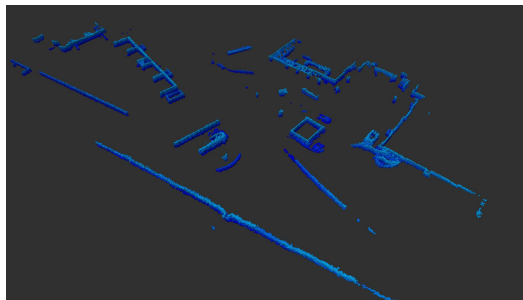
- But how do I create a global map (e.g. from SLAM)?

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>`
- 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 - 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:

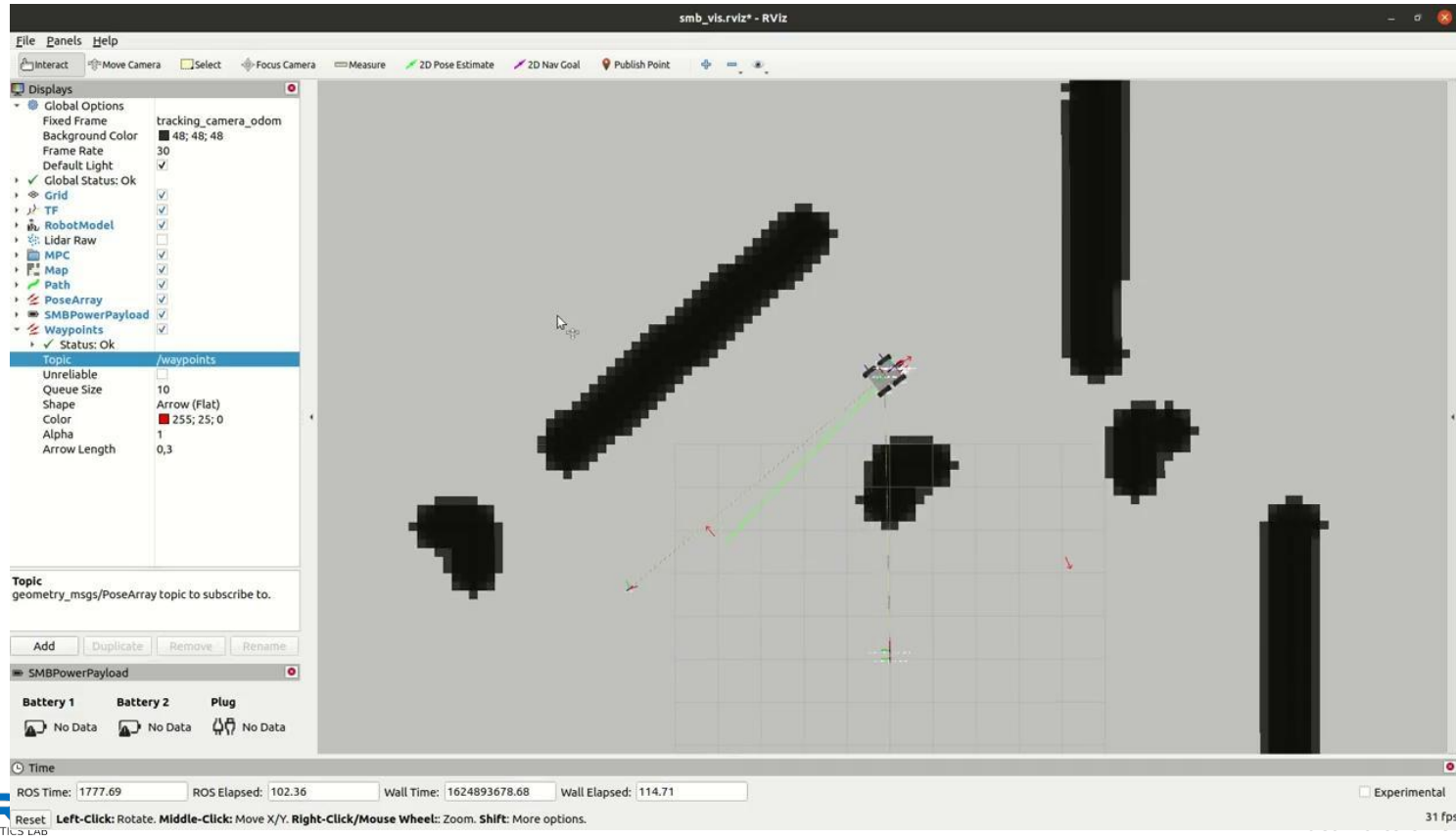
```
roslaunch smb_navigation navigate2d_ompl.launch global_frame:=map_new  
robot_base_frame:=base_new
```
- Use existing global maps for planning (e.g. from previous missions)
- Follow a set of waypoints

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](#))

