ETH zürich

Tutorial on Path Planning

ETH Robotics Summer School

Luca Bartolomei

30 June, 2019

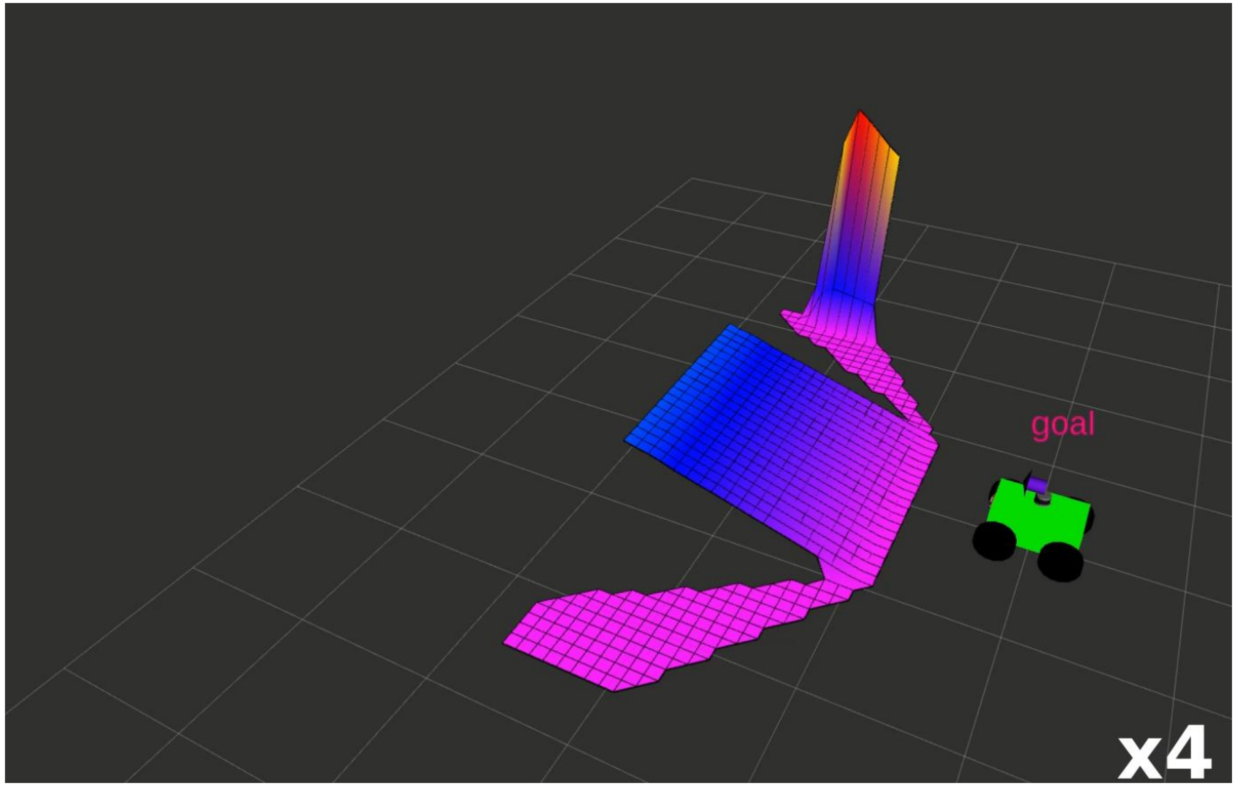
VISION FOR ROBOTICS LAB

Tutorial Objectives

1. Introduction to Path Planning

2. Description of Path Planning pipeline
   - Voxblox & Traversability Mapping
   - Global & Local Planning

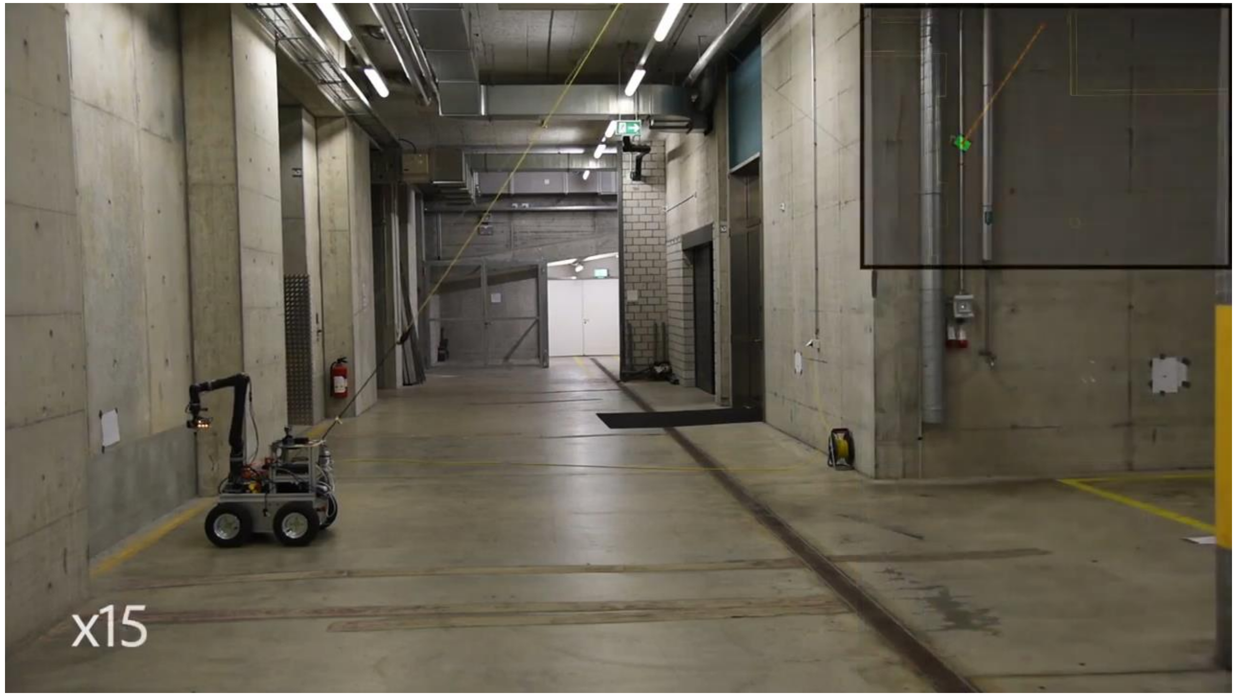
3. How to install & run the packages



Introduction to Path Planning

- Perform autonomous navigation to fulfil high-level goals
  - Reach task location
  - Exploration and data collection
  - …

- Main components for path planning:
  1. Mapping
  2. Path generation

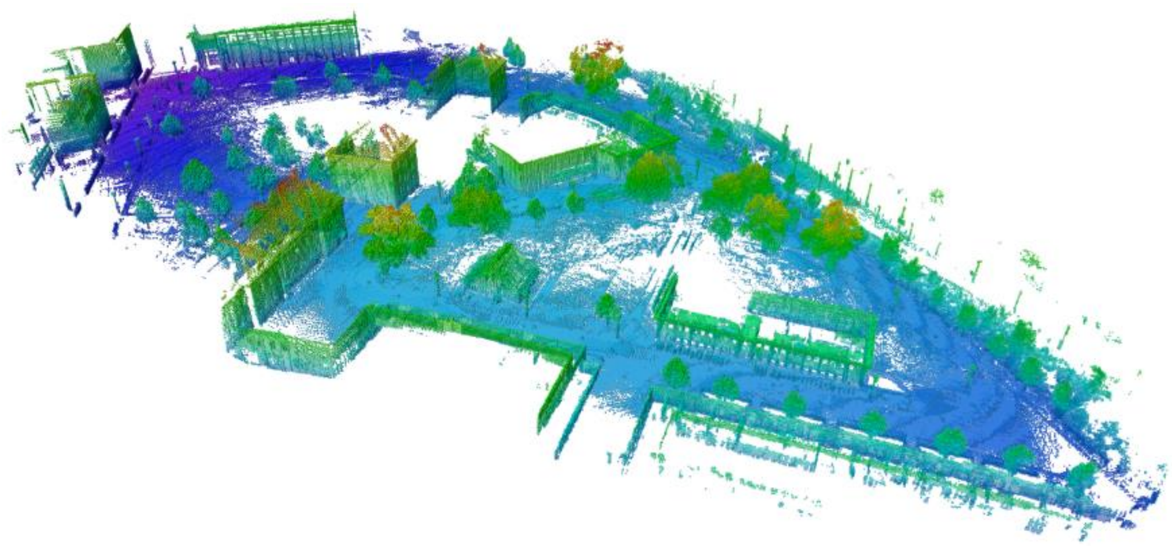


"A Fully-Integrated Sensing and Control System for High-Accuracy Mobile Robotic Building Construction", A. Gawel et al., IROS 2019

Introduction to Path Planning: Mapping

- Different map representations:

OctoMap → 3D occupancy grid



“OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees”, A. Hornung et al., Autonomous Robots 2013

Voxblox → meshes and distance fields

“Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-Board MAV Planning”, H. Oleynikova et al., IROS 2017

Introduction to Path Planning: Path Generation

- Hierarchical architecture

Global Planner →

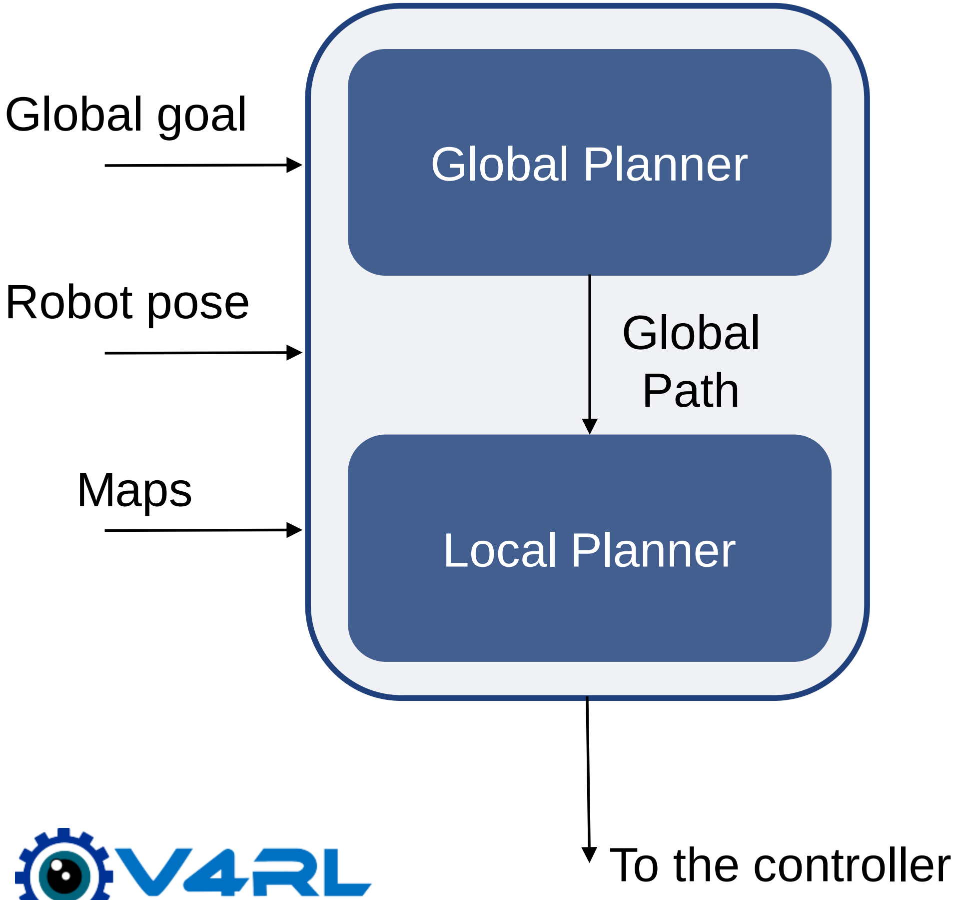
- Find coarse path towards goal configuration
- Optimistic (unknown space is free)
- Lower re-planning rate

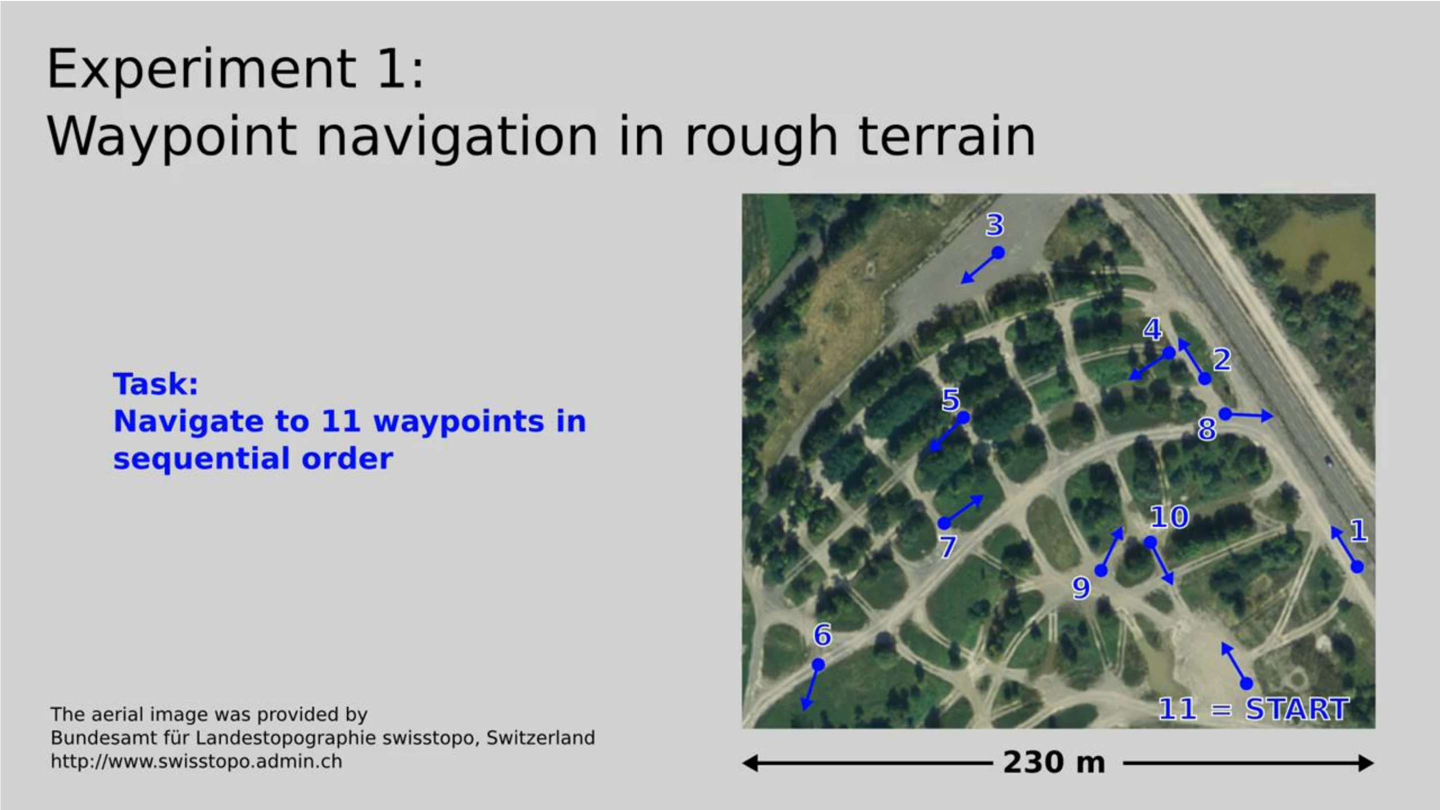
Local Planner →

- Local obstacle avoidance
- Consider vehicle dynamics
- Higher re-planning rate

Introduction to Path Planning: Path Generation

- Hierarchical architecture

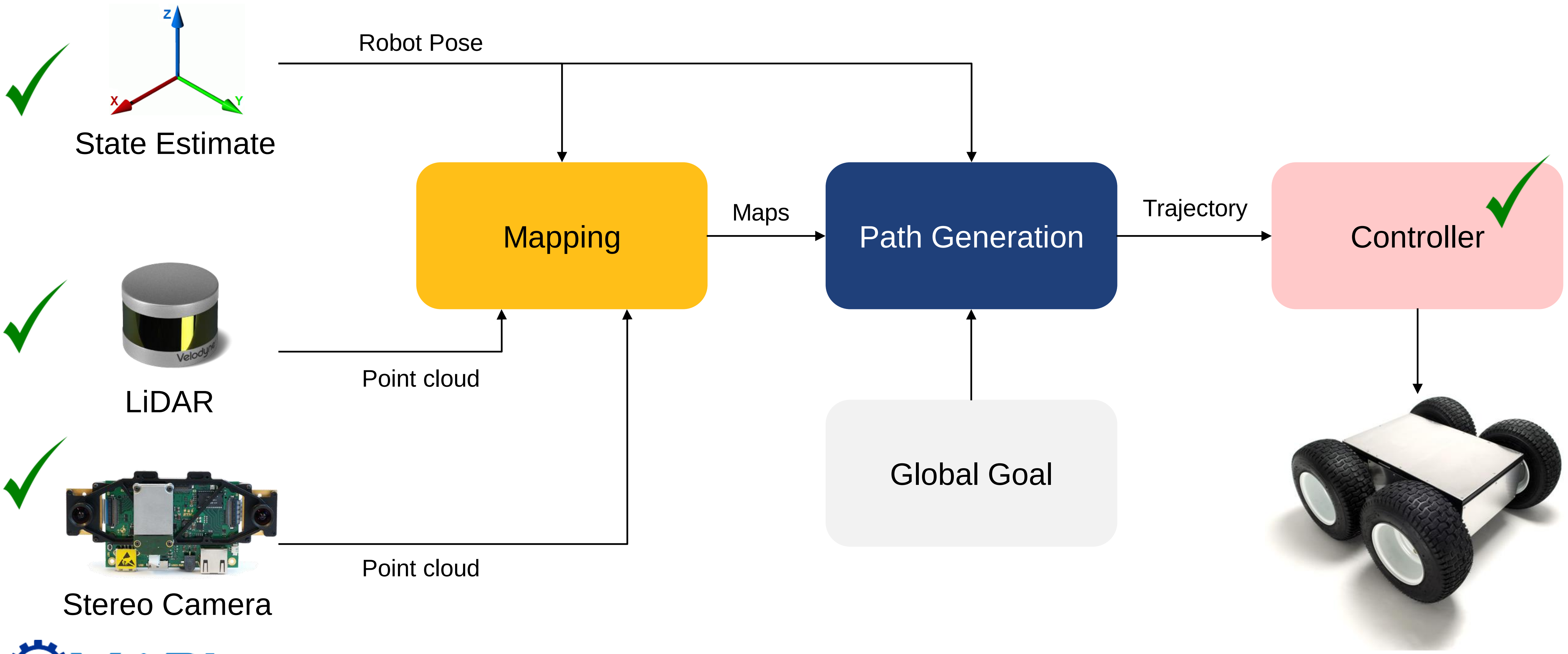




"Driving on Point Clouds: Motion Planning, Trajectory Optimization, and Terrain Assessment in Generic Nonplanar Environments", P. Krusi et al., Journal of Field Robotics 2017

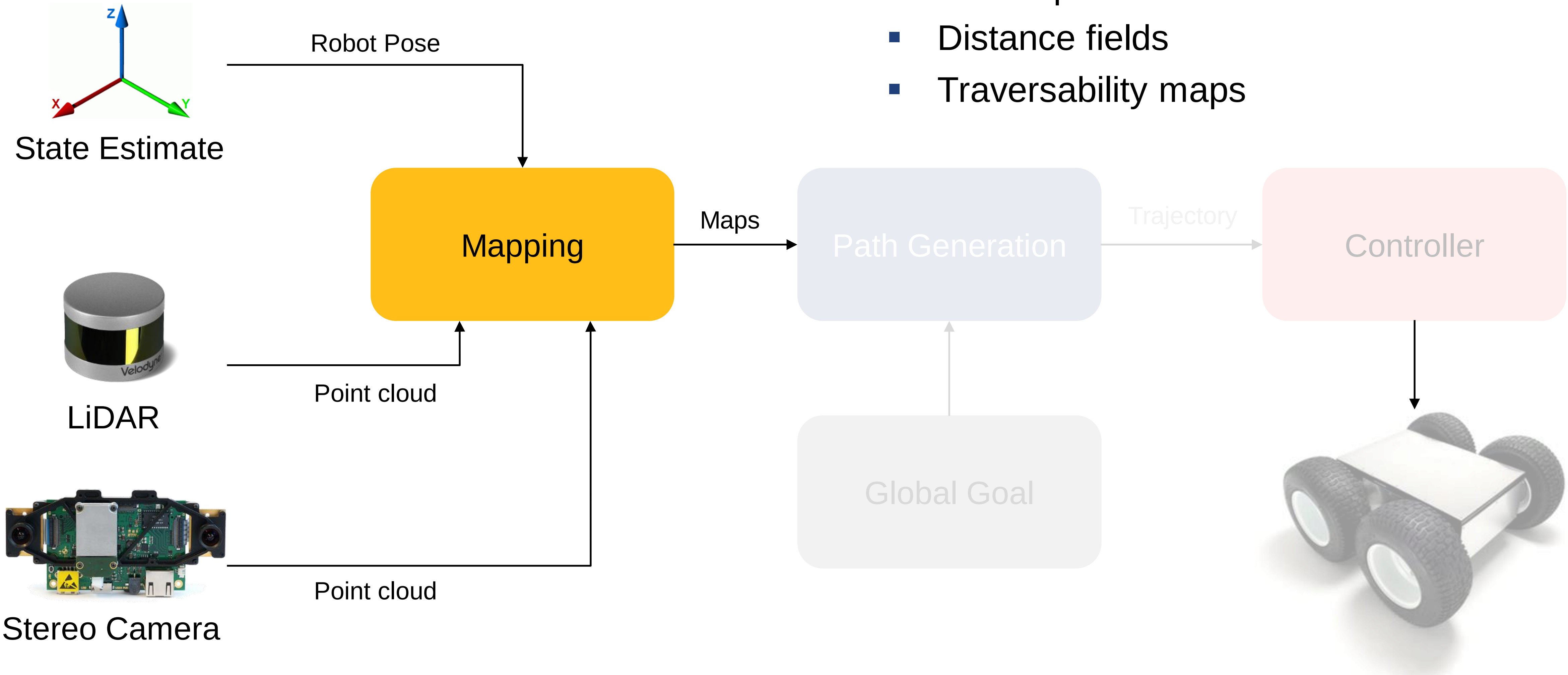
VISION FOR ROBOTICS LAB

Pipeline Overview



Pipeline Overview: Mapping

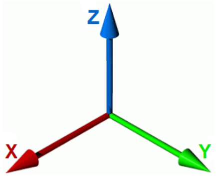
Create maps of the environment
- Distance fields
- Traversability maps



Pipeline Overview: Mapping

Create maps of the environment

- Distance fields
- Traversability maps



State Estimate

Robot Pose

LiDAR

Point cloud

Stereo Camera

Point cloud

Mapping

Mapping

Voxblox

voxblox

LiDAR Point Cloud

Traversability Estimation

Stereo Camera Point Cloud

Mapping: Voxblox

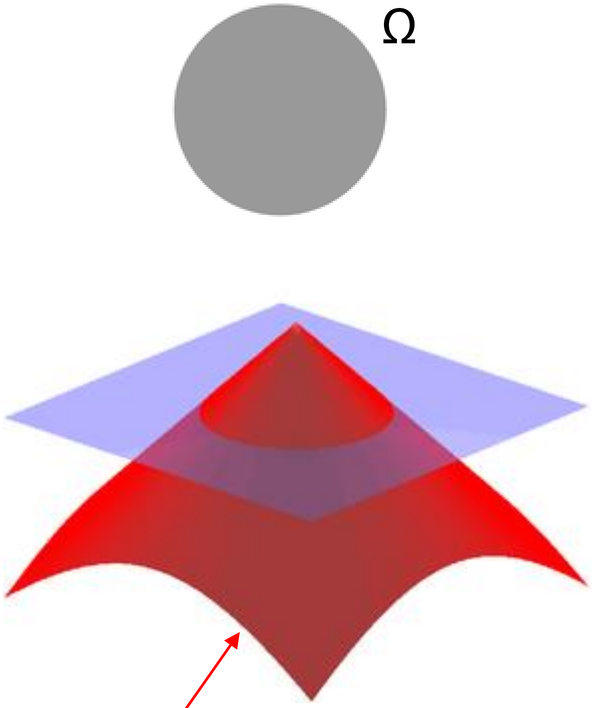
voxblox

Inputs: LiDAR point cloud, Robot pose

- Mesh representations
- Construct signed distance fields

Signed distance function of set Ω :

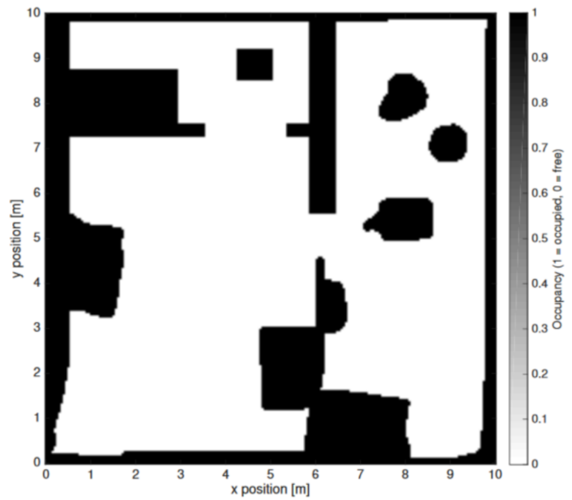
distance of point x from the boundary of Ω, with sign determined by whether $x \in \Omega$



Signed distance function

Mapping: Voxblox

voxblox



Occupancy Grid

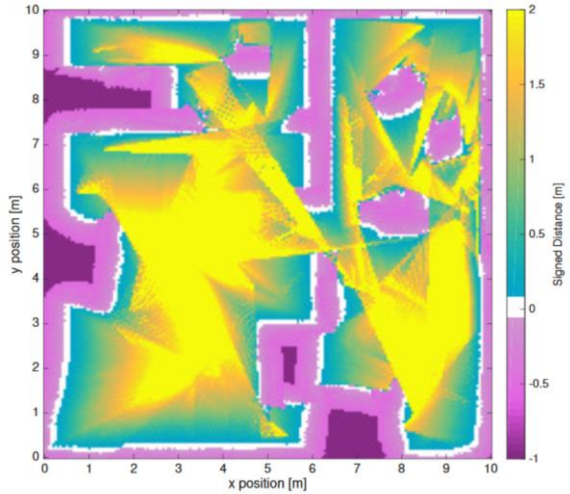
Inputs: LiDAR point cloud, Robot pose

- Mesh representations
- Construct signed distance fields
  - TSDF → Truncated Signed Distance Field

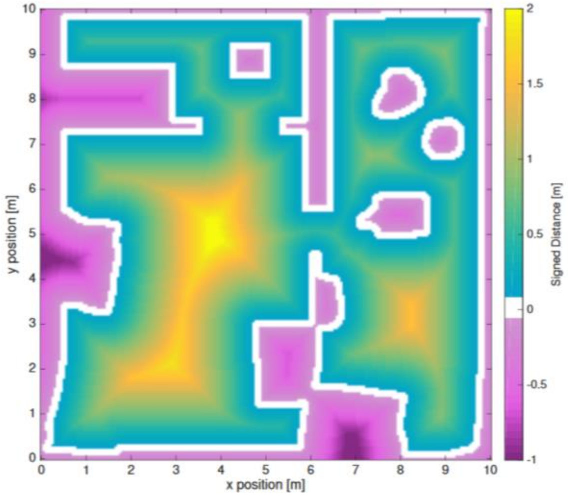
    Distance to the surface along the ray direction from the center of the sensor

  - ESDF → Euclidean Signed Distance Field

    Distance to the nearest obstacle for every position



TSDF



ESDF

Mapping: Voxblox

voxblox

Inputs: LiDAR point cloud, Robot pose
- Mesh representations
- Construct signed distance fields
  - TSDF → Truncated Signed Distance Field
  - ESDF → Euclidean Signed Distance Field

→ Distances to obstacles
→ Gradients of distance fields
→ "Global" map for collision checks

"Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-Board MAV Planning", H. Oleynikova et al., IROS 2017

Mapping: Voxblox

voxblox

- Documentation → https://voxblox.readthedocs.io

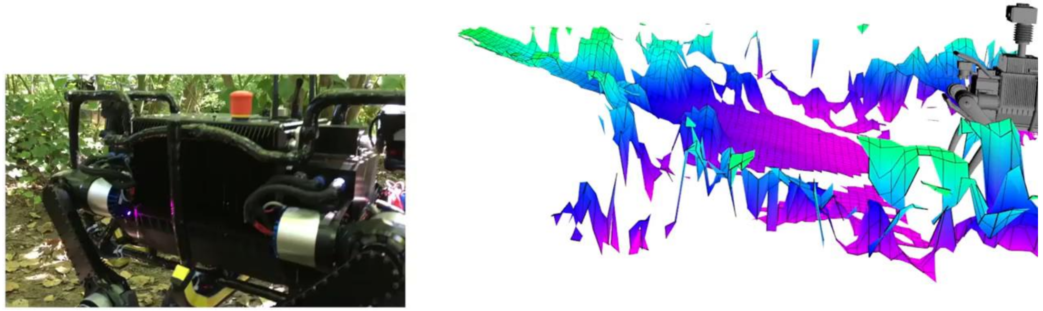
Subscribed Topics	ROS message
/velodyne_points	[sensor_msgs/PointCloud2]

Published Topics	ROS message
mesh	[voxblox_msgs/Mesh]
esdf_pointcloud	[sensor_msgs/PointCloud2]
tsdf_pointcloud	[sensor_msgs/PointCloud2]
traversable	[sensor_msgs/PointCloud2]
occupied_nodes	[visualization_msgs/MarkerArray]

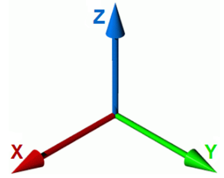
Need to specify robot dimension

Mapping: Traversability Estimation

Inputs: Stereo point cloud, Robot pose



"Probabilistic Terrain Mapping for Mobile Robots with Uncertain Localization", P. Fankhauser et al., R-AL 2018



State Estimate

Robot Pose

Stereo Camera

Point cloud

Elevation Mapping

Elevation Map

Traversability Estimation

Local Traversability Map

Each position in traversability map has traversability score in range $[0, 1]$

Mapping: Traversability Estimation

- Documentation:

→ https://github.com/ANYbotics/elevation_mapping

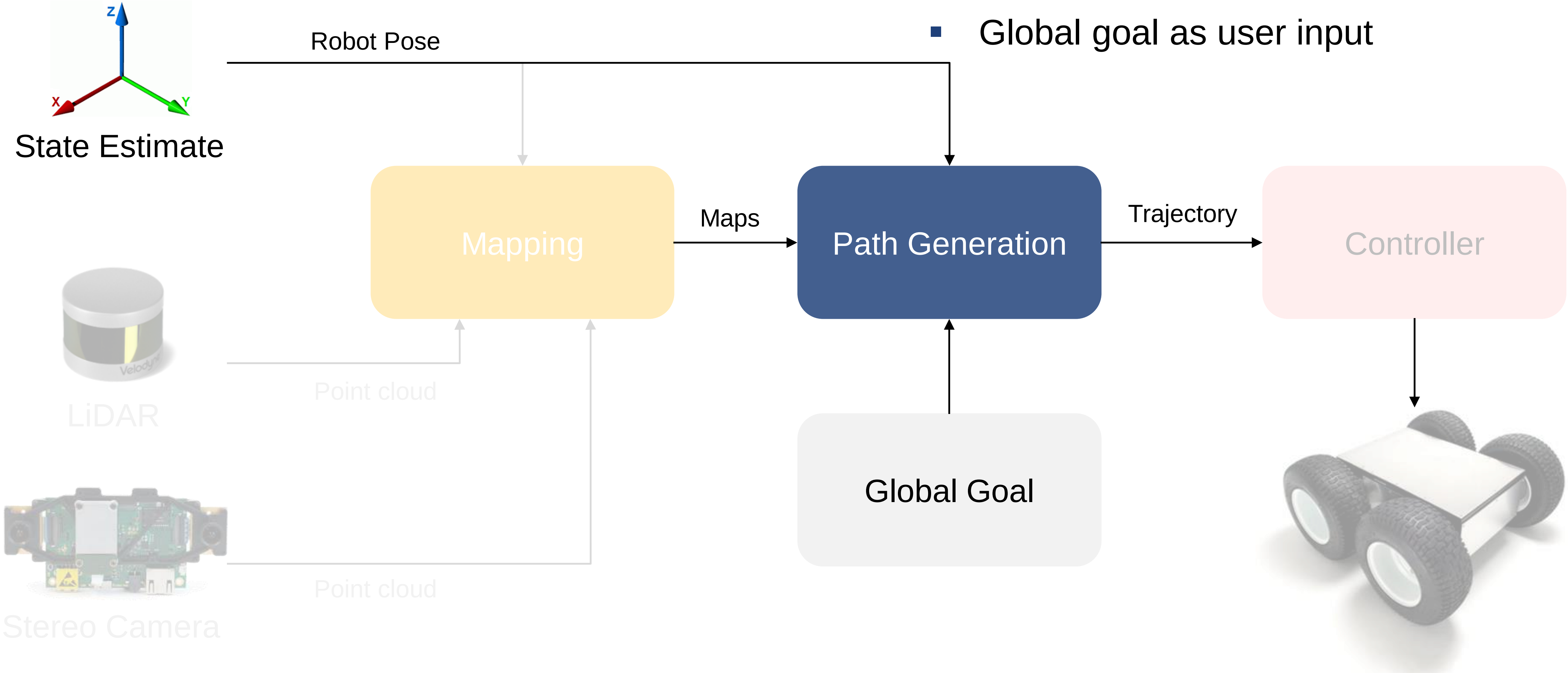
→ https://github.com/leggedrobotics/traversability_estimation

Subscribed Topics	ROS message
/vi_sensor/pointcloud	[sensor_msgs/PointCloud2]
/stamped_pose_covariance	[geometry_msgs/PoseWithCovarianceStamped]

Published Topics	ROS message
/elevation_mapping/elevation_map	[grid_map_msgs/GridMap]
/traversability_map	[grid_map_msgs/GridMap]

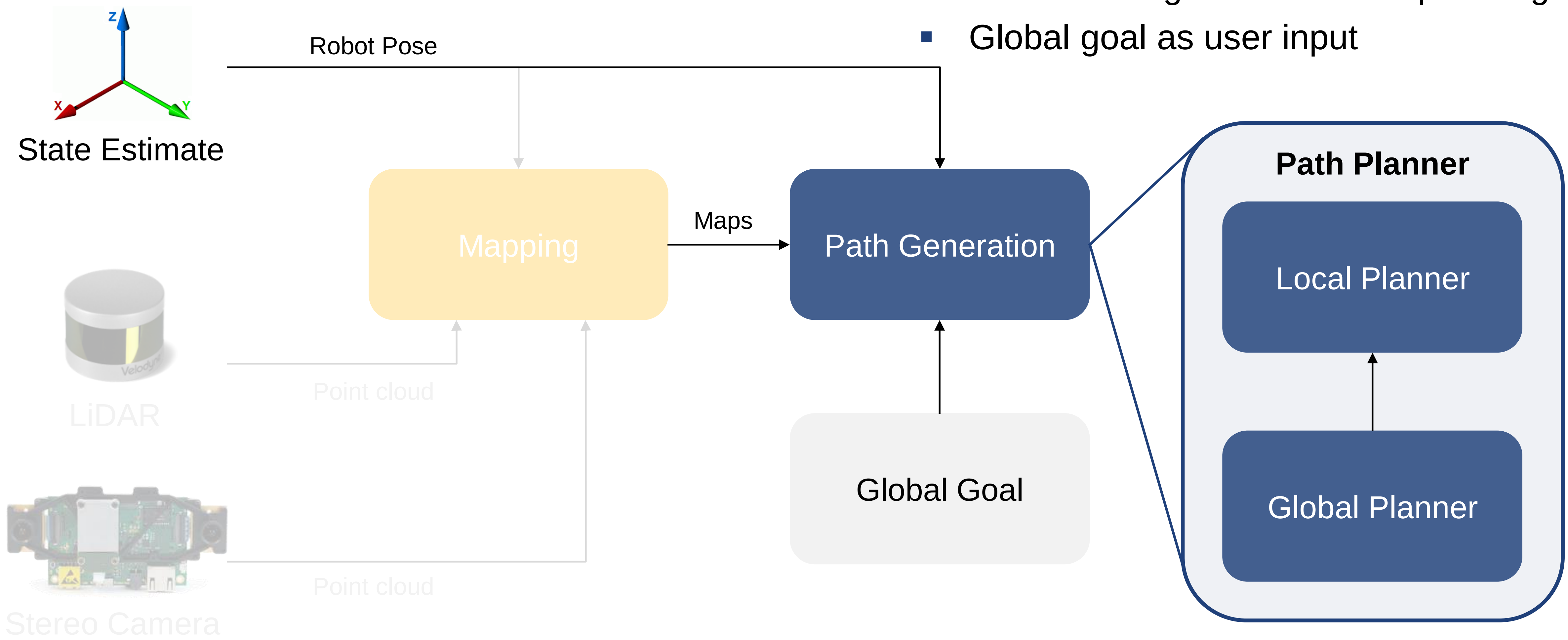
Pipeline Overview: Path Planner

- Use maps to generate paths
- Divided into global and local planning
- Global goal as user input



Pipeline Overview: Path Planner

- Use maps to generate paths
- Divided into global and local planning
- Global goal as user input



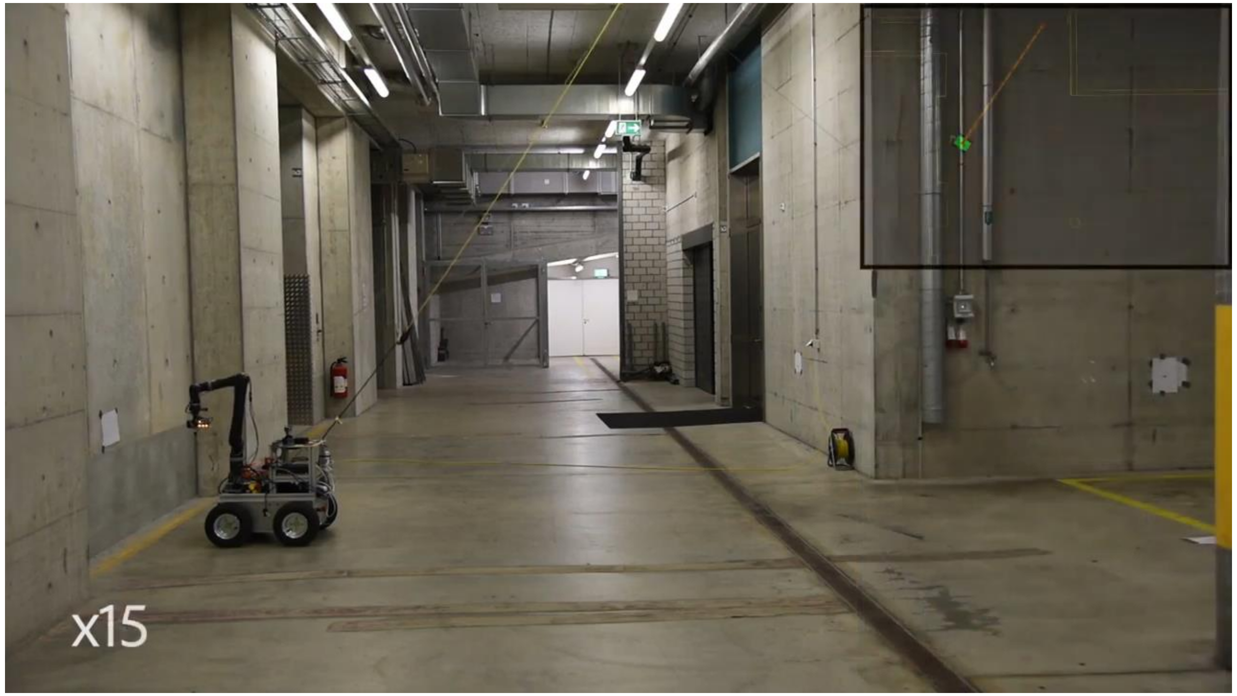
Path Planner: Global Planner

- Path from current state directly to global goal configuration
- Optimistic → unknown space = free
- Uses TSDF and traversability map

Global Planning Algorithm

Graph-based → A*, PRM, …

Tree-based → RRT*, RRT#, …



“A Fully-Integrated Sensing and Control System for High-Accuracy Mobile Robotic Building Construction”, A. Gawel et al., IROS 2019

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

Path Planner: Global Planner - OMPL

- Path Planning Library by Ioan Suçan, Mark Moll and Lydia E. Kavraki [1]
- Library to solve planning problem in different state spaces: $\mathbb{R}^3, SE(3), \mathbb{R}^2, SE(2)$
- Implementation of many sampling-based planners (RRT, PRM, …)

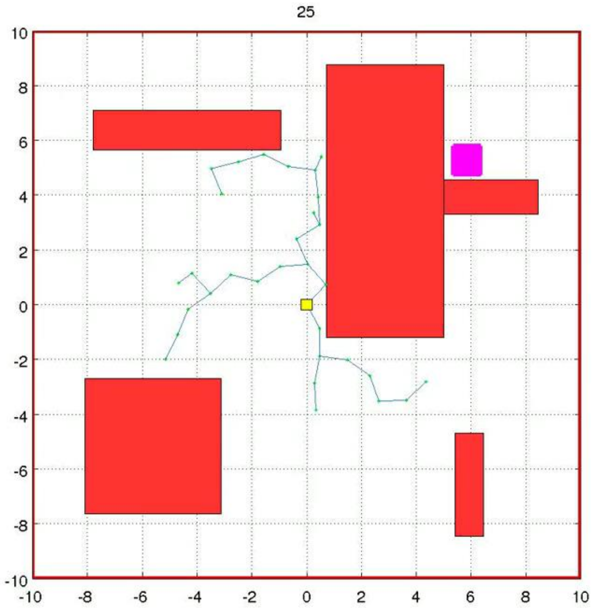
Picture from https://ompl.kavrakilab.org/

[1] "The Open Motion Planning Library, Ioan Suçan, Mark Moll and Lydia E. Kavraki, IEEE Robotics & Automation Magazine, 2012

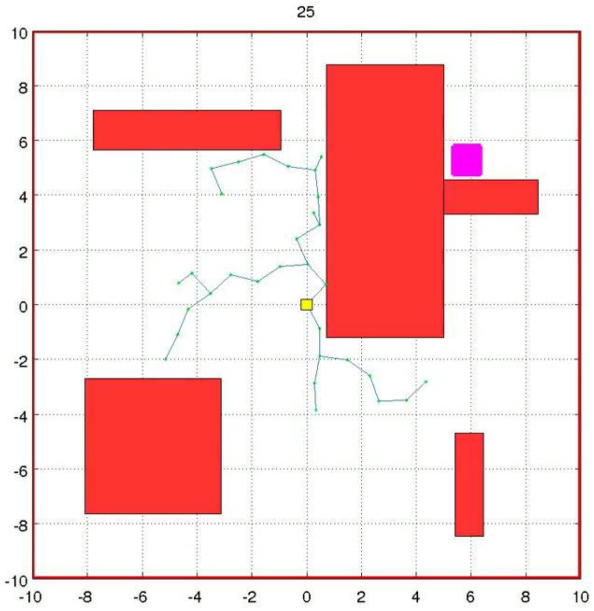
Path Planner: Global Planner – RRT*

- RRT*(Optimal Rapidly-exploring Random Tree)
  - Probabilistically complete and optimal algorithm
  - Introduces *nearest neighbor operations* and *rewiring*

RRT



RRT*



VISION FOR ROBOTICS LAB

“Sampling-based algorithms for optimal motion planning”, S. Karaman and E. Frazzoli, IJJR 2011

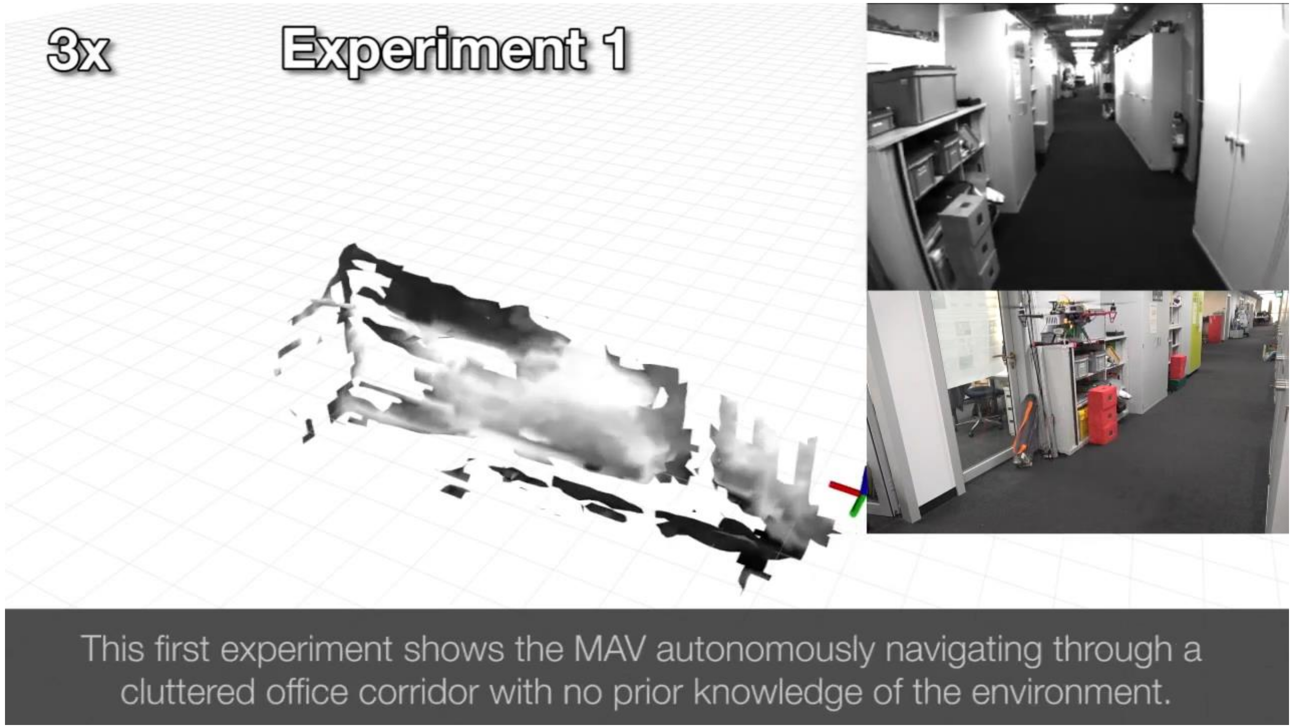
Path Planner: Global Planner – Informed RRT*

- Improve convergence time
  → "smart sampling"

- Main steps:
  1. Find an initial solution with RRT*
  2. Focus sampling in the ellipsoid around this initial solution

Path Planner: Local Planner

- Compute locally optimal path:
  - Vehicle dynamics
  - Obstacle avoidance
- Pessimistic:
  → unknown space = occupied
- Uses ESDF and traversability map



"An Open-Source System for Vision-Based Micro-Aerial Vehicle Mapping, Plannig, and Flight in Cluttered Environments", H. Oleynikova et al., IJJR 2019

Path Planner: Local Planner

- Local planner solver: CHOMP
  (**C**ovariant **H**amiltonian **O**ptimization for **M**otion **P**lanning)

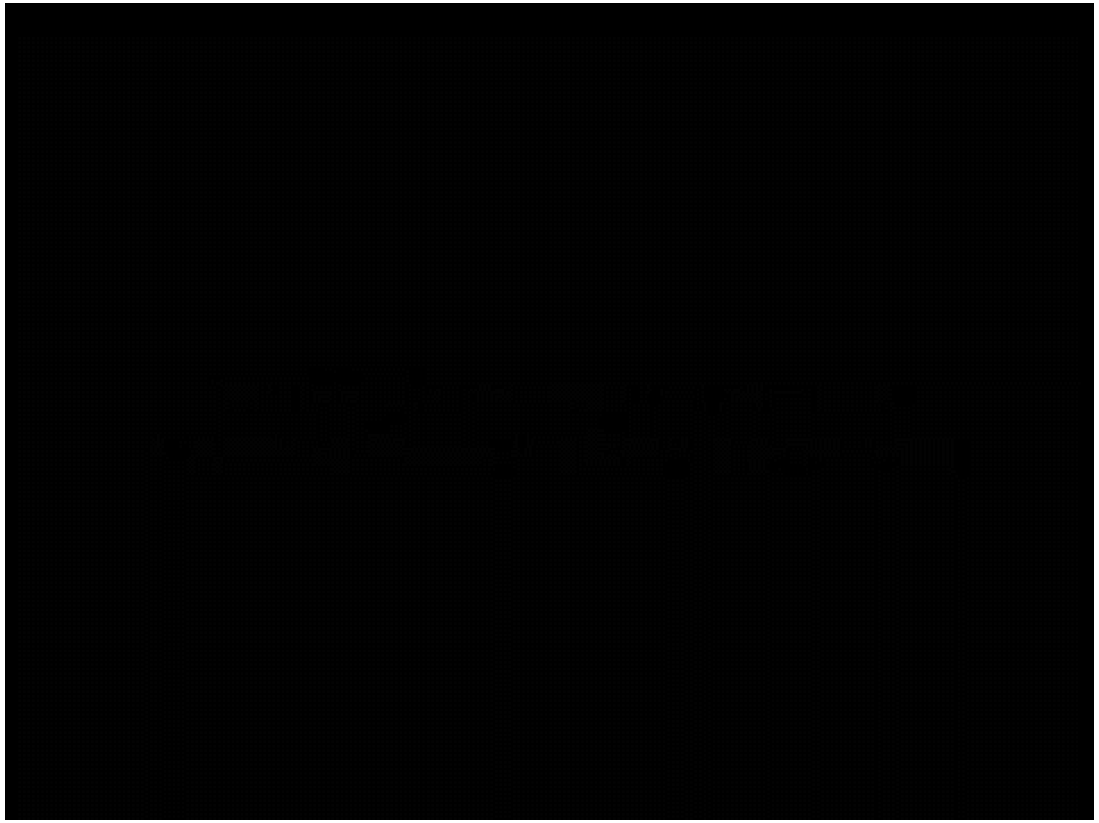
Optimal trajectory

$$\xi^* = \underset{\xi}{\operatorname{argmin}}\, \mathcal{F}_{obst}[\xi] + \lambda\, \mathcal{F}_{smooth}[\xi]$$

Obstacle avoidance objective

Dynamic feasibility objective

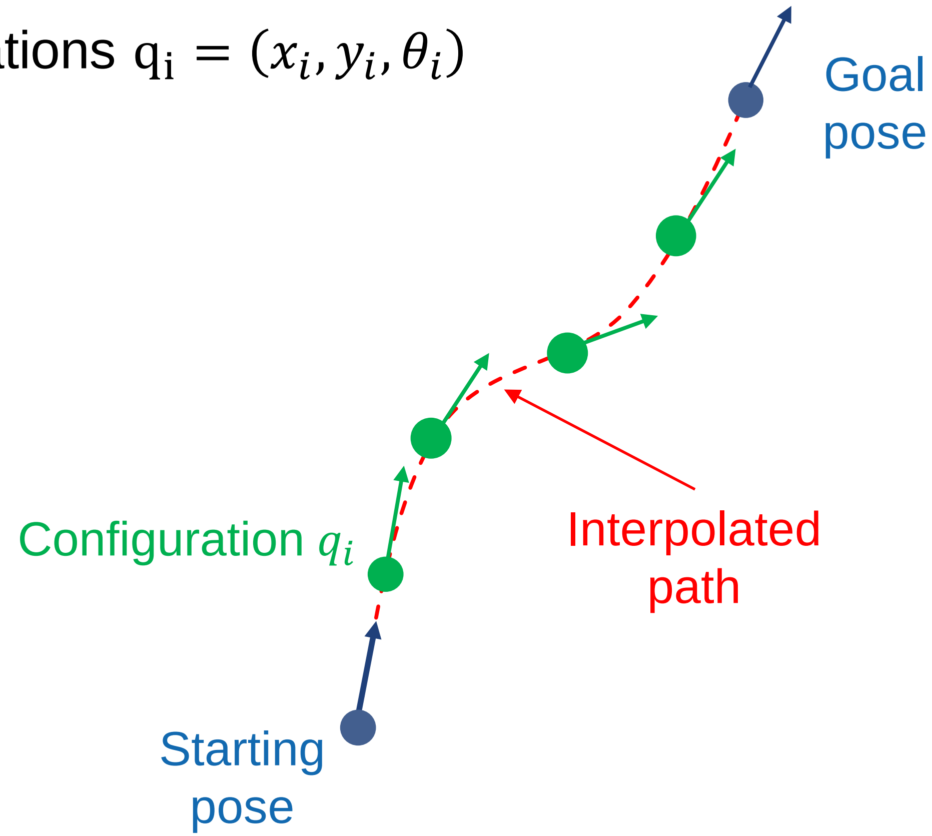
➡ Add nonholonomic constraints to optimization problem



"CHOMP: Covariant Hamiltonian optimization for motion planning", M. Zucker et al., IJJR 2013

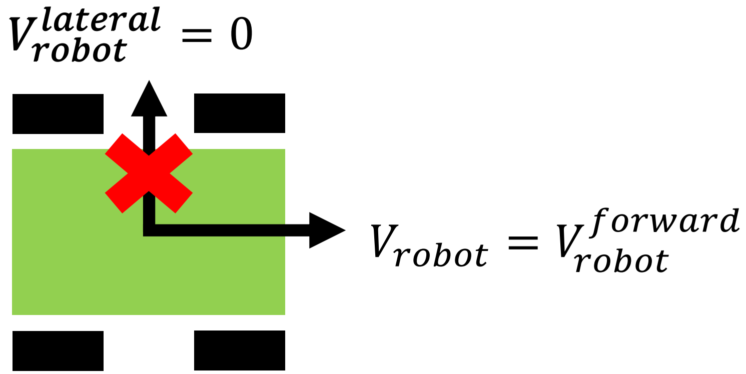
Path Planner: Local Planner

- Restrict planning in SE(2)
  → Trajectory ξ : discrete sequence of $n+1$ robot configurations $\mathrm{q_i} = (x_i, y_i, \theta_i)$



- Nonholonomic constraints:

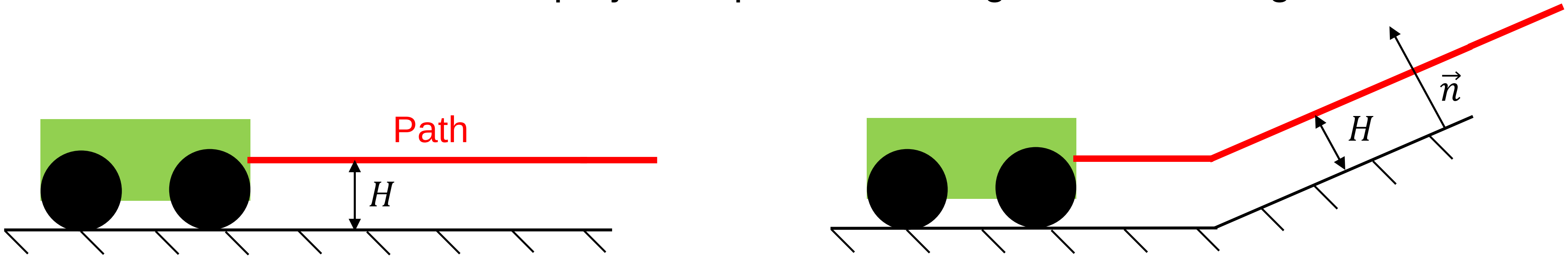
1. Rolling constraints: $v_x \sin(\theta) - v_y \cos(\theta) = 0$
2. Enforce forward motions: $||V_{robot}|| - V_{robot} = 0$



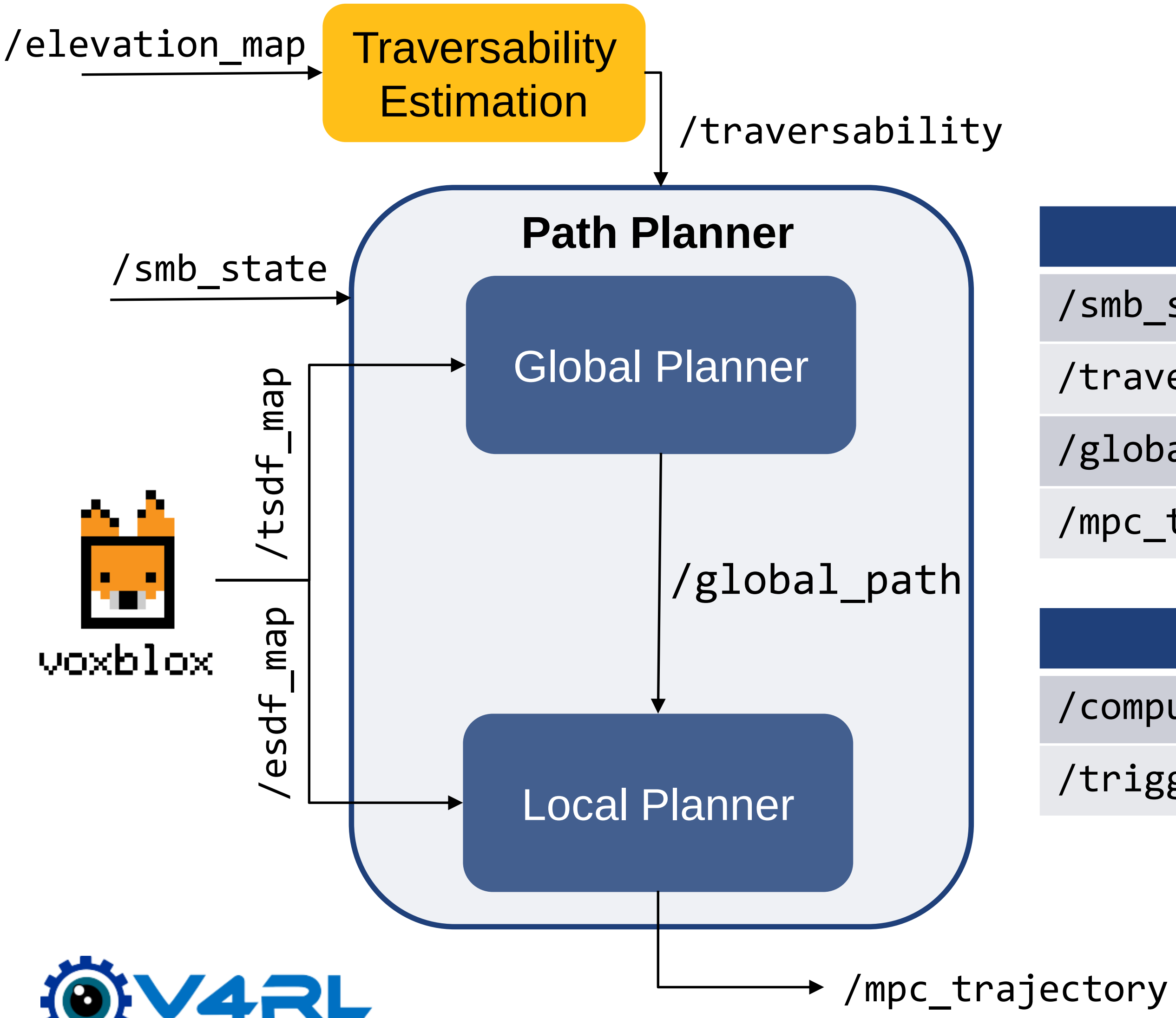
Path Planner: Maps

- Voxblox
  → Query each candidate position for distance to closest obstacles

- Traversability map
  → Fix a *planning height* H from the ground (planning in 2D)
  1. Query positions for traversability score
  2. Perform collision checks on projected positions along normal to the ground



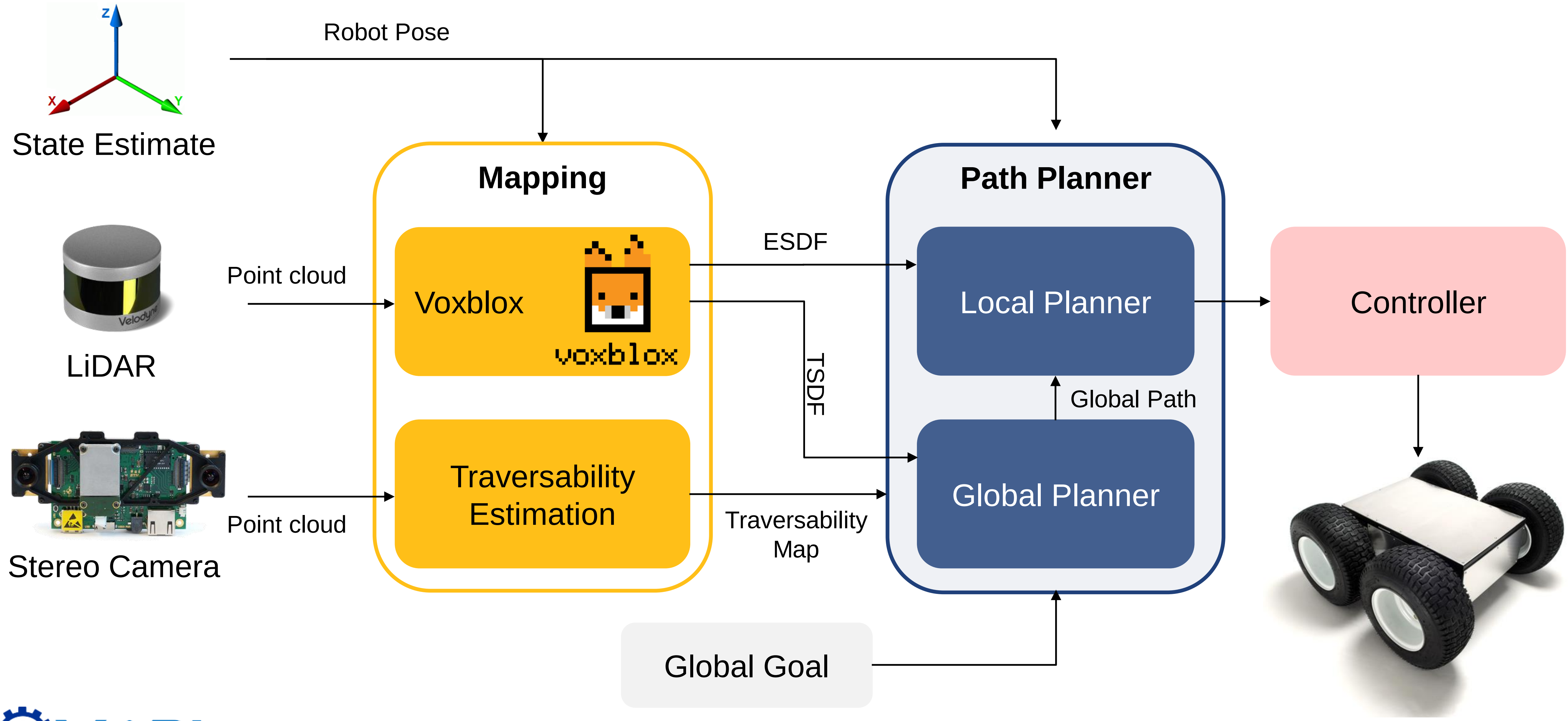
Path Planner: ROS Communication



Topics	ROS message
/smb_state	[smb_msgs/SmbState]
/traversability	[grid_map_msgs/GridMap]
/global_path	[nav_msgs/Path]
/mpc_trajectory	[nav_msgs/Path]

Services	ROS service
/compute_global_path	[smb_planner_msgs/PlannerService]
/trigger_local_planner	[std_srvs/Empty]

Full Pipeline



Installation of the packages (1/3)

- Follow the instructions for the summer school mono-repo:

  https://github.com/ethz-asl/eth_robotics_summer_school_2019

- Create a catkin workspace:

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws
catkin init
catkin config --extend /opt/ros/melodic
catkin config --merge-devel
catkin config -DCMAKE_BUILD_TYPE=Release
```

Installation of the packages (2/3)

- Clone the repository and update the workspace:

```
cd ~/catkin_ws/src/
git clone https://github.com/ethz-asl/eth_robotics_summer_school_2019.git
wstool init
wstool merge eth_robotics_summer_school_2019/dependencies.rosinstall
wstool up
```

- Build the workspace:

```
cd ~/catkin_ws/
catkin build
```

Installation of the packages (3/3)

- Be careful **to be up-to-date with all the packages**!
- If you have cloned the repository a few days ago, pull again the new changes:

```
cd ~/catkin_ws/src/
wstool up
```

- Then, build the workspace again

Structure of the Path Planner Repository

- Meta-package: `smb_path_planner`

`smb_planner_common`
→ Common parameters files/classes for local and global planners
→ It stores all parameters in `smb_planner_common/cfg`

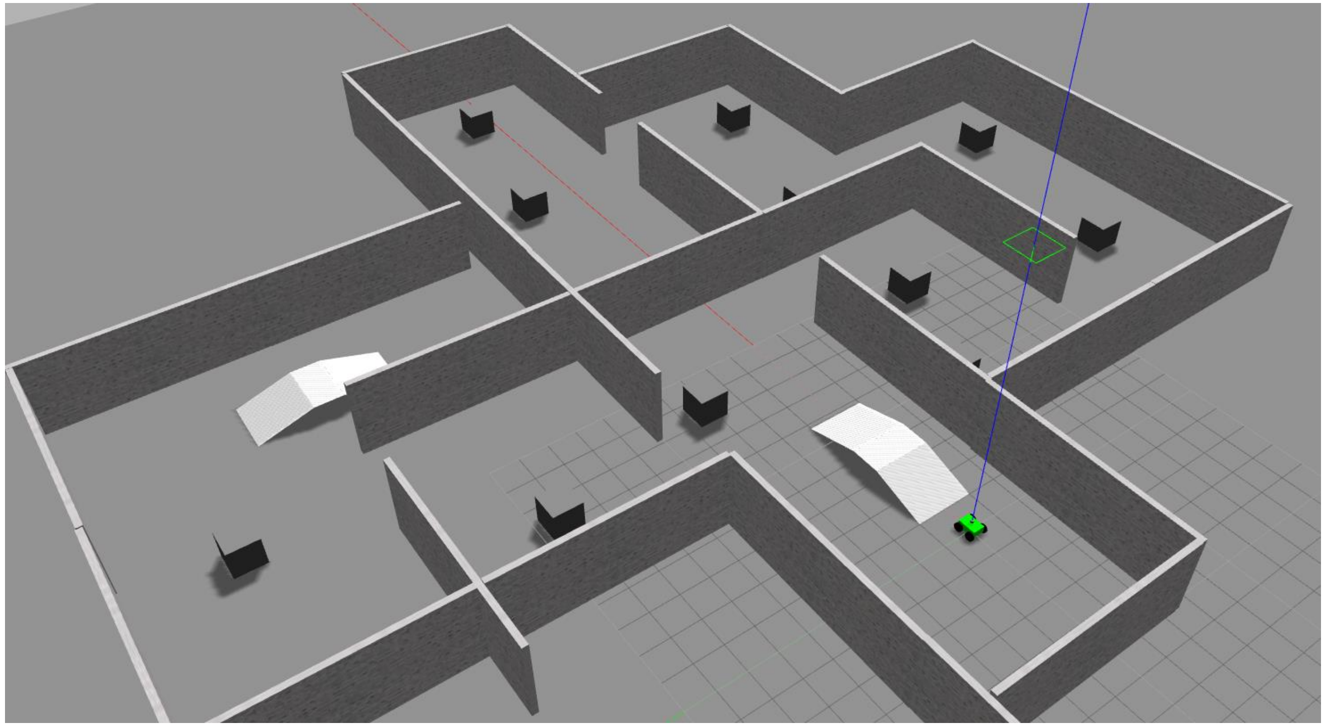
`smb_planner_rviz`
→ Implementation of the RViz planning panel

`smb_local_planner`
`smb_global_planner`
Implementation of the actual planners

Running the Simulation (1/6)

- To start the simulation in Gazebo, run in one terminal:

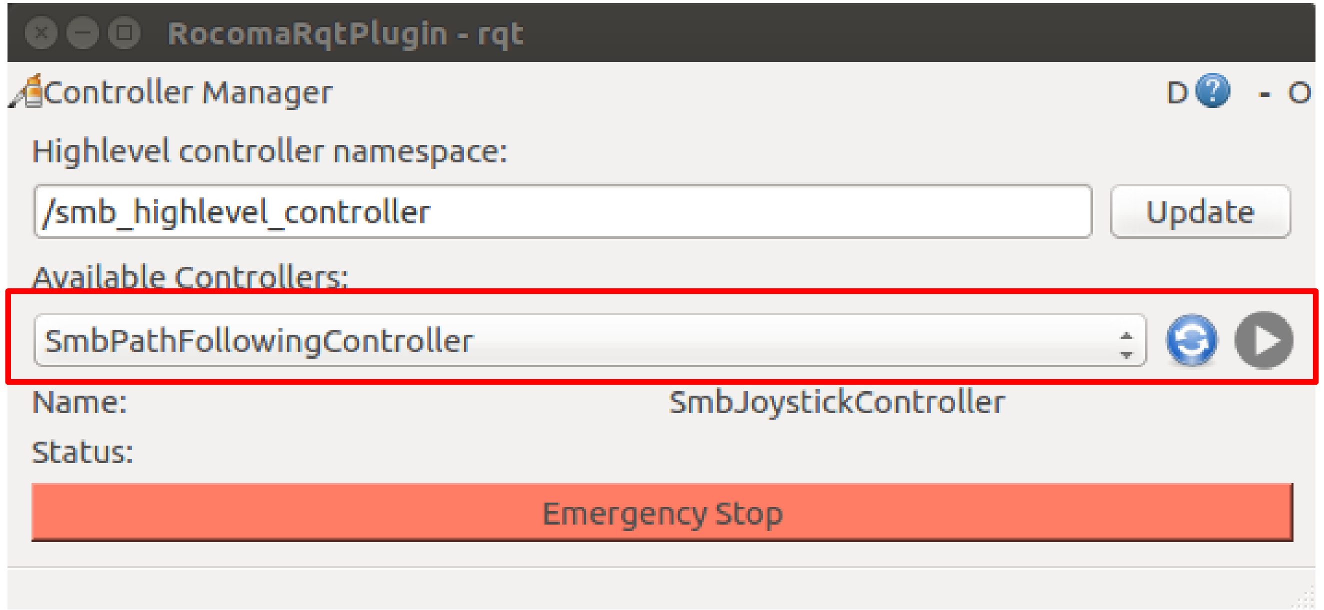
```
roslaunch smb_sim smb_path_planner.launch run_gazebo_gui:=true
```



This launch files opens:
1. Gazebo
2. RViz
3. Control panel

Running the Simulation (2/6)

- Select the right controller from the control panel:



1. Select controller
2. If the controller does not show up, press `refresh`
3. Press `play`

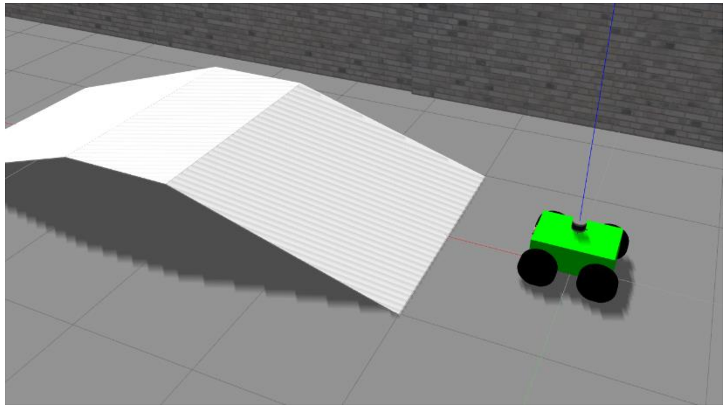
Running the Simulation (3/6)

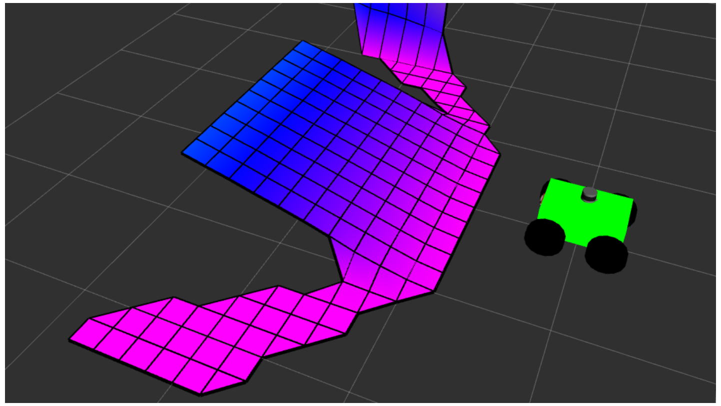
- Start the elevation mapping package:

```
roslaunch smb_local_planner smb_elevation_mapping_simulation.launch
```

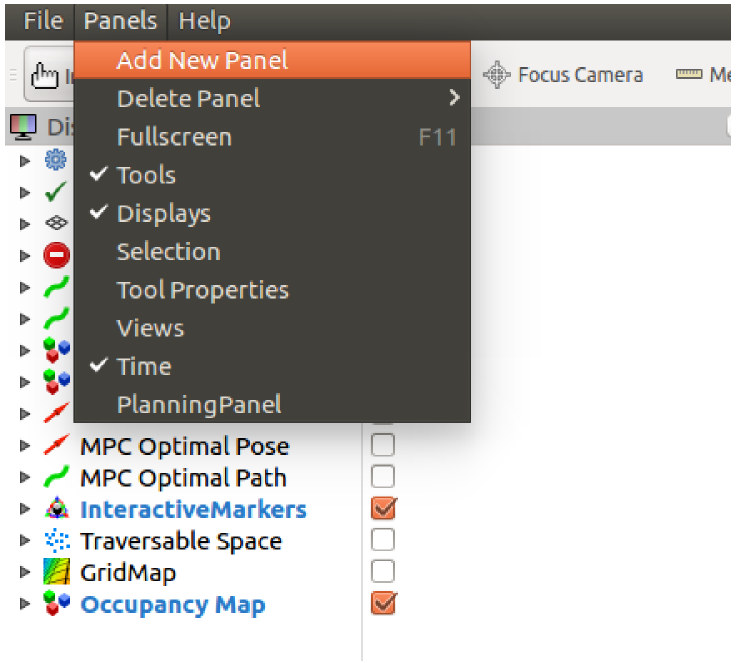
- Start the local and the global planners:

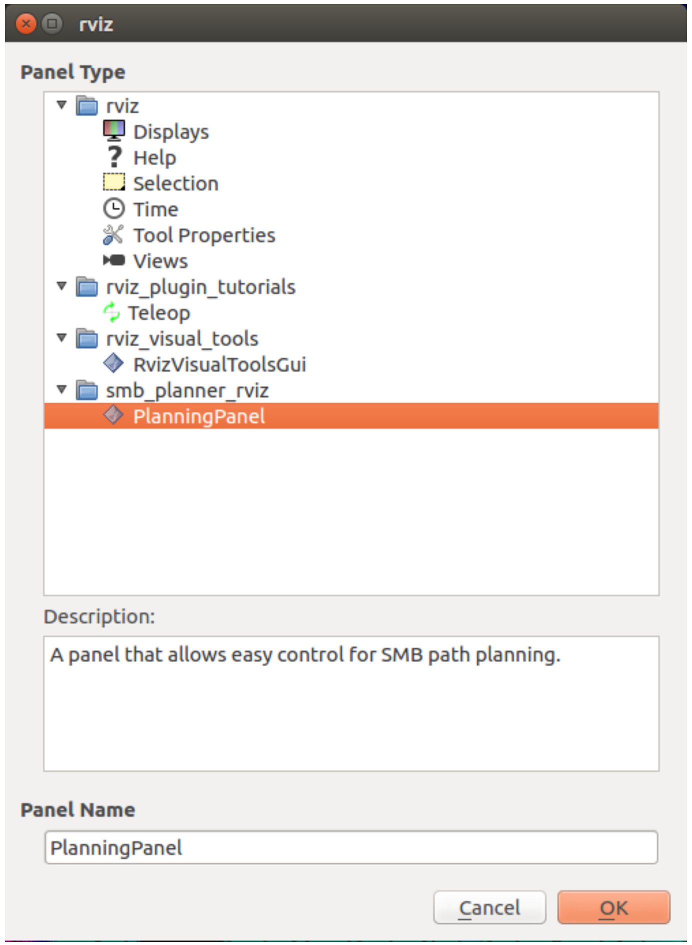
```
roslaunch smb_local_planner smb_planner_simulation.launch
```



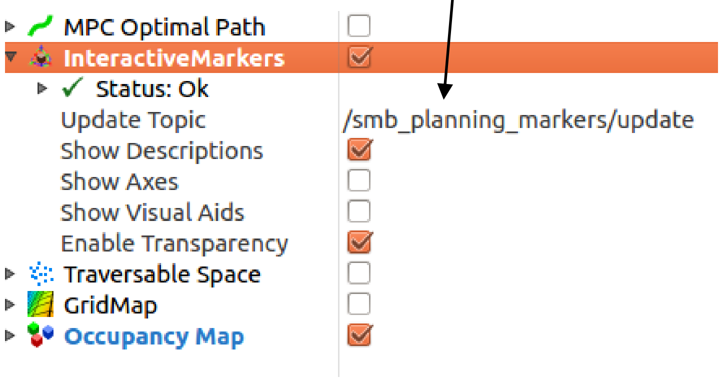


Running the Simulation (4/6)





Select the right topic!



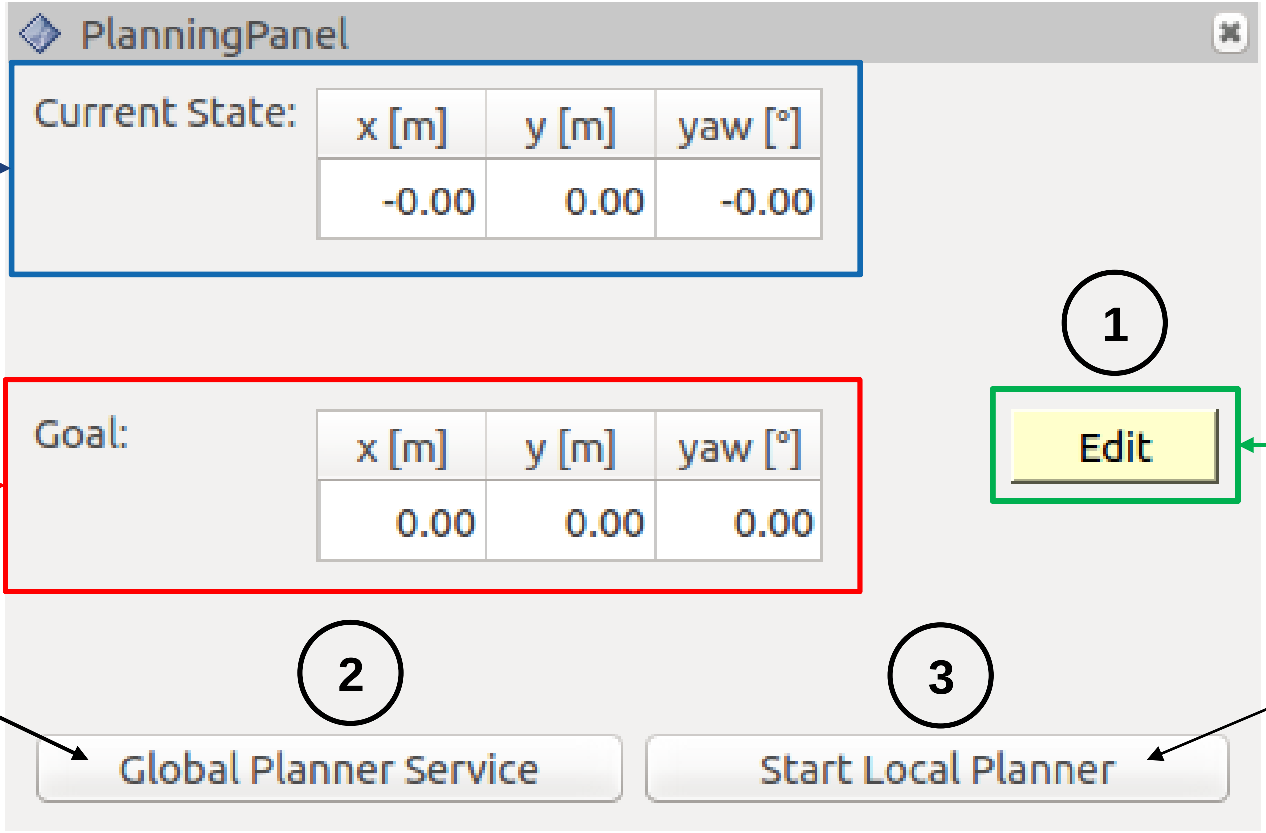
Running the Simulation (5/6)

- To send global goals and trigger the planners, use the Planning Panel in RViz

This will change with current pose information

Global goal information

Always trigger the global planner first!



Press this button to use RViz interactive markers to move global goal

Once you have a global path, trigger local planner

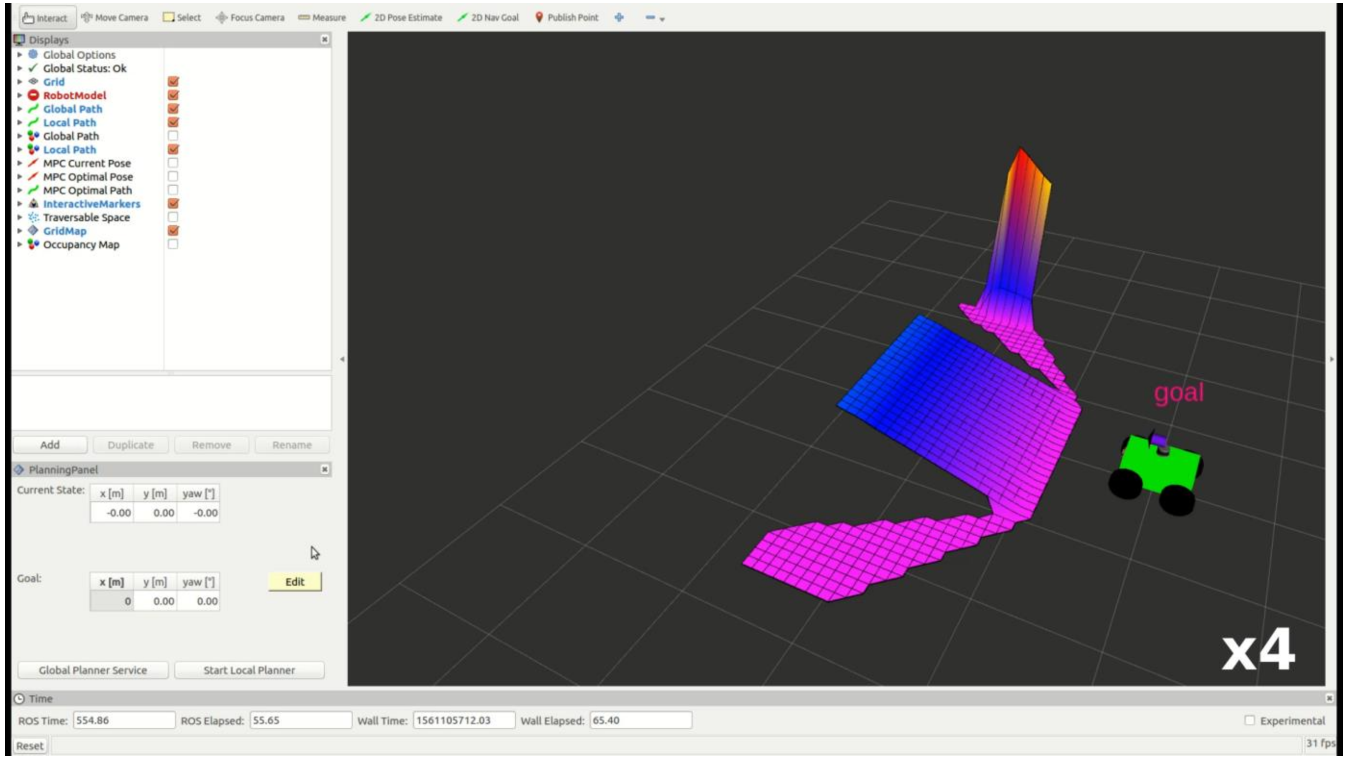
Running the Simulation (6/6)

To start the mission:

1. Set a global goal (planning panel)

2. Start global planning:
   `Global Planner Service`

3. Start local planning:
   `Start Local Planner`



Running the simulation: Parameters

➡ `smb_path_planner/smb_planner_common/cfg/smb_planner_parameters_simulation.yaml`

Parameter	Default value	Effects
`robot_radius`	0.1 m	Influences traversable space in Voxblox
`planning_height`	0.5 m	Nominal planning height H
`check_traversability`	True	Whether or not to use traversability map when planning
`global_timer_dt`	0.1 s	Spinning time for collision checks in global planner
`num_seconds_to_plan`	5.0 s	Time given to global planner to find a path
`local_replan_dt`	0.25 s	Re-planning rate for the local planner
`command_dt`	0.25 s	Rate for local planner to send commands to controller
`local_goal_distance`	2.0 m	Distance of local goal along global path from current position
CHOMP parameters	-	Parameters for CHOMP solver for local planner
……		

Running on the Real Robot (1/3)

- On the PC of the SMB, start in separate terminals:
  - Main `roscore`

```
$ roscore                                                          # terminal 1
```

  - LPC (state estimation, controllers)

```
$ roslaunch smb_lpc lpc.launch                                     # terminal 2
```

  - ICP Mapper (LiDAR mapping)

```
$ roslaunch ethzasl_icp_mapper supermegabot_robosense_dynamic_mapper.launch   # terminal 3
```

- On your PC, start the Operator PC (OPC):

```
$ roslaunch smb_opc opc.launch                                     # User PC
```

Running on the Real Robot (2/3)

Required only if traversability checks are enabled

- On the PC of the SMB, start the planners:

`$ roslaunch smb_local_planner smb_elevation_mapping_real.launch`	`# terminal 4`
`$ roslaunch smb_local_planner smb_planner_real.launch`	`# terminal 5`

- **Task**:
  - From what you have learned in simulation, tune the parameters for the real robot
  - The parameters can be found in:
    `smb_path_planner/smb_planner_common/cfg/smb_planner_parameters.yaml`
  - In particular, tune the CHOMP parameters
  - Note: Remember to be up-to-date with the repositories

Running on the Real Robot (3/3)

- In case you don't manage to run the full planner:
  → Run global planner + SMB controller

- Change the output topic name of the global planner in:
  `smb_planner_common/cfg/topics.yaml`

  Modify the `globalPlanner/outputTrajectoryMsgName`:
  `/global_path` ➡ `/mpc_trajectory`