Robotics, Artificial Intelligence and Embedded Systems

Final Presentation: Intelligent Mobile Robots with ROS

NavPy - A Python Navigation Stack

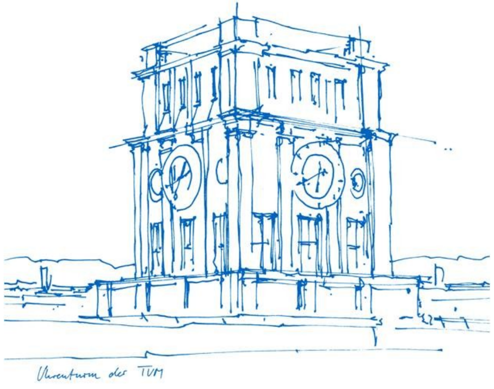
Technical University of Munich

Department of Computer Science

I6 - Chair of Robotics, Artificial Intelligence and Embedded Systems

Fabian Kolb, Qianhao Li, Bastian Wittmann

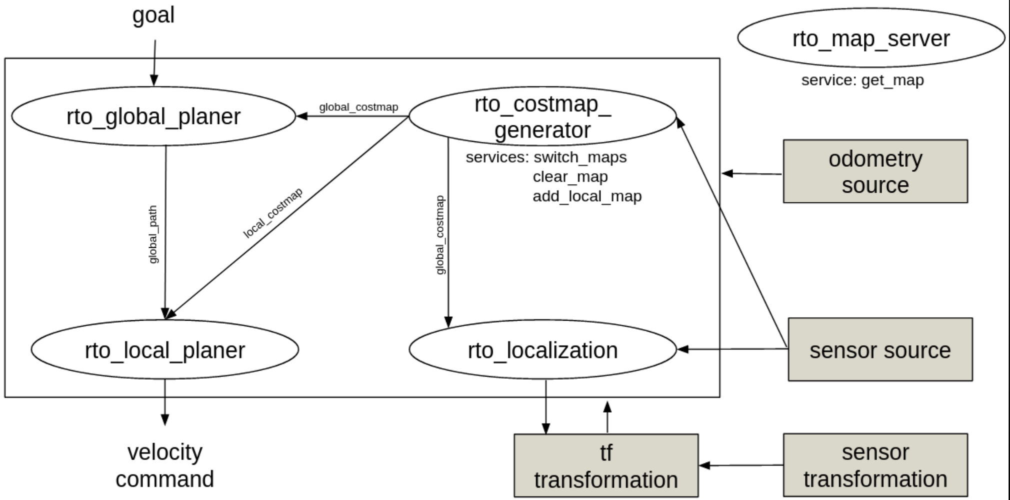
Munich, 03/16/2021



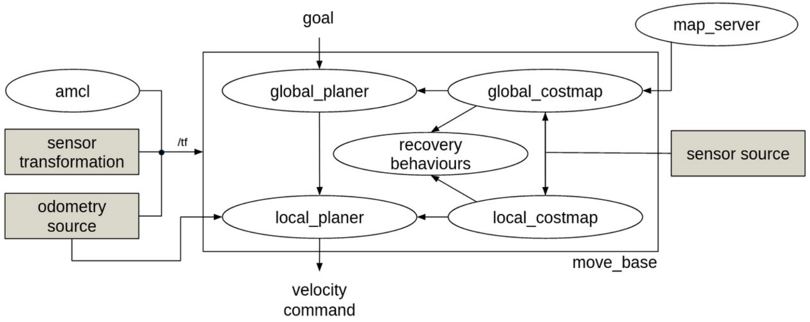
Roadmap

- **Introduction and Overview**
- **Main Elements**
  - Costmap Generator
  - Localization
  - Global Planning
  - Local Planning
- **Challenges and Outlook**
- **Live Demonstration**

NavPy vs. ROS Navigation Stack

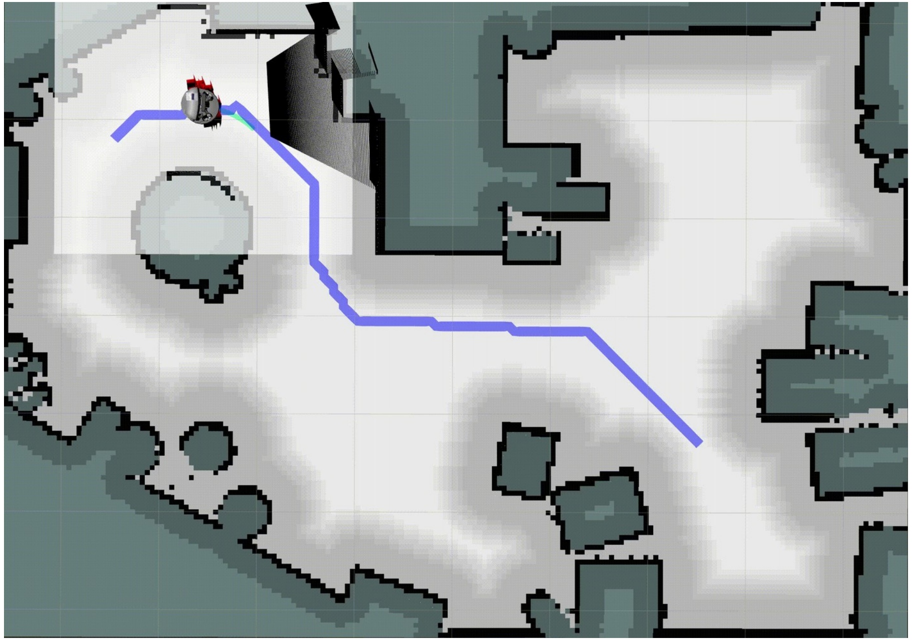


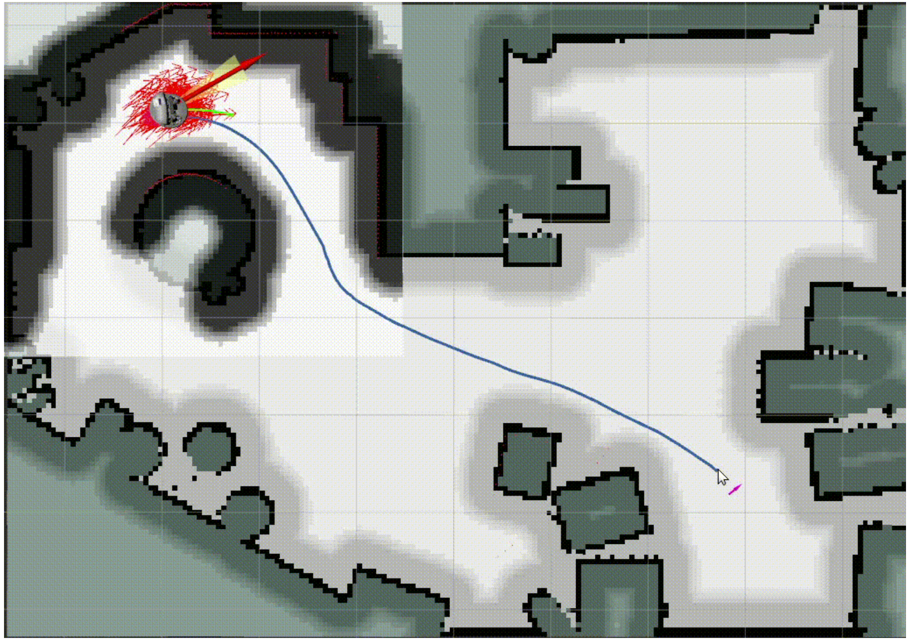
NavPy



ROS Navigation Stack

NavPy vs. ROS Navigation Stack

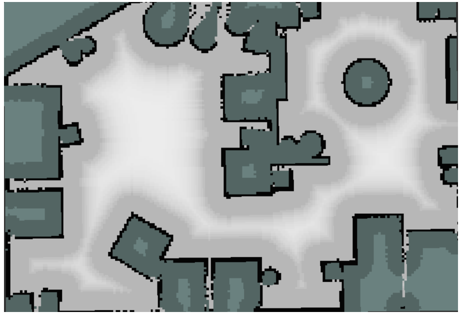

NavPy


ROS Navigation Stack

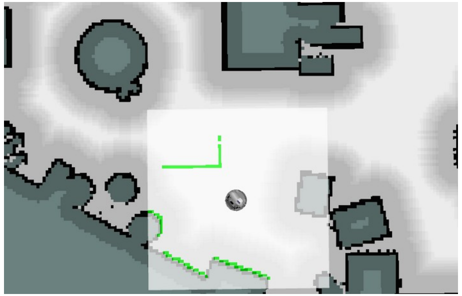
Costmap Generator - An Overview

- costmap generator is responsible for creating the local and global costmap
- global costmap can be cleared, extended and changed via services

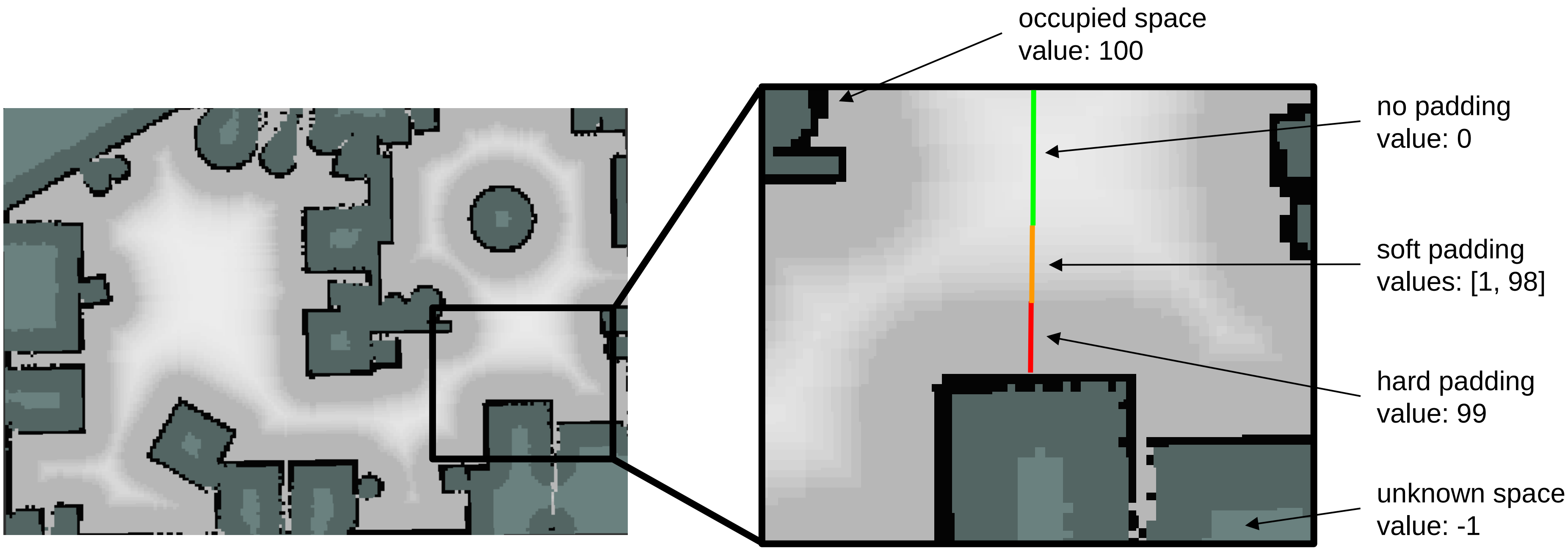
- global costmap:
  - pads the static map stored in the map server
  - allows to penalize paths close to obstacles
  - allows the use of a point representation for path planning



- local costmap:
  - recognizes dynamic changes in the environment by considering the current laser scan range measurements
  - can be incorporated in the global costmap to allow a better localization performance and to cope with environment changes

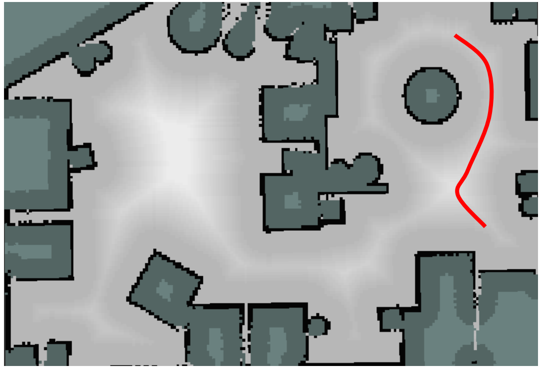


Costmap Generator - Global Costmap

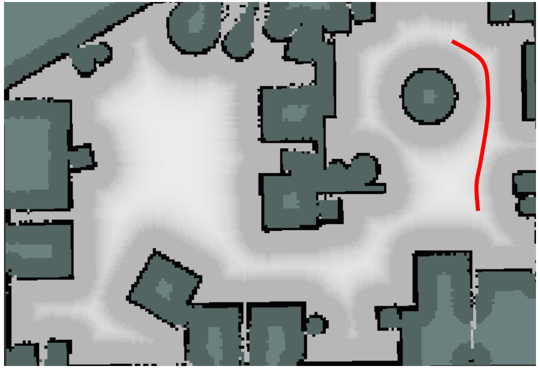


Costmap Generator - Global Costmap

- possible decay types lead to different global planner behaviours
- there exist multiple decay types (reciprocal, linear, exponential)



decay type: linear
decay distance: 1.0m



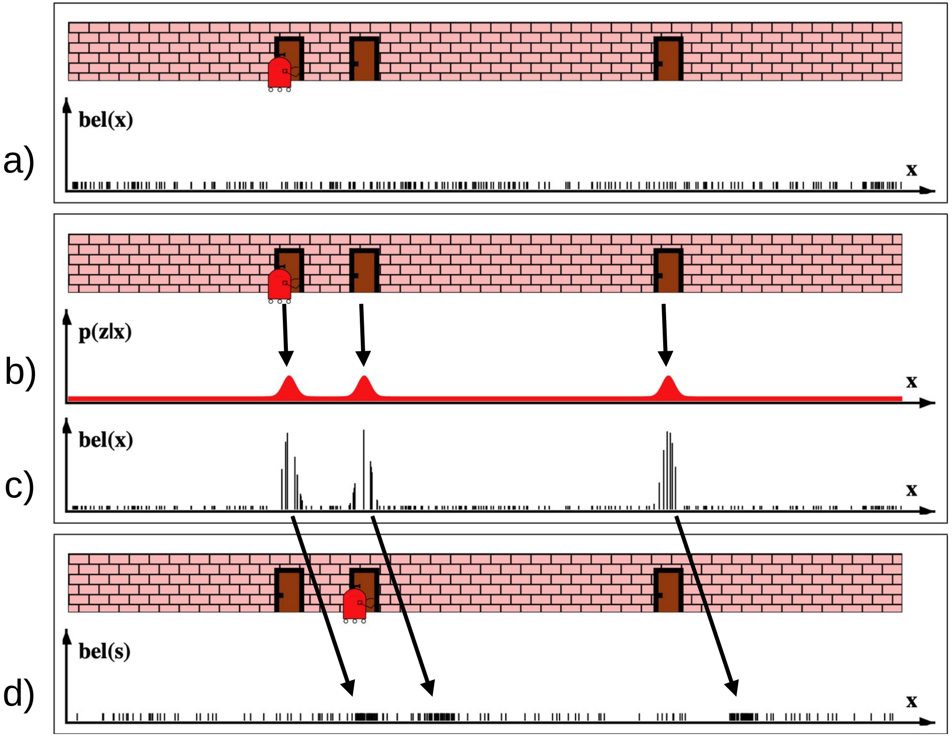
decay type: exponential
decay distance: 1.0m

decay type: exponential
decay distance: 0.2m

Monte Carlo Localization (MCL)

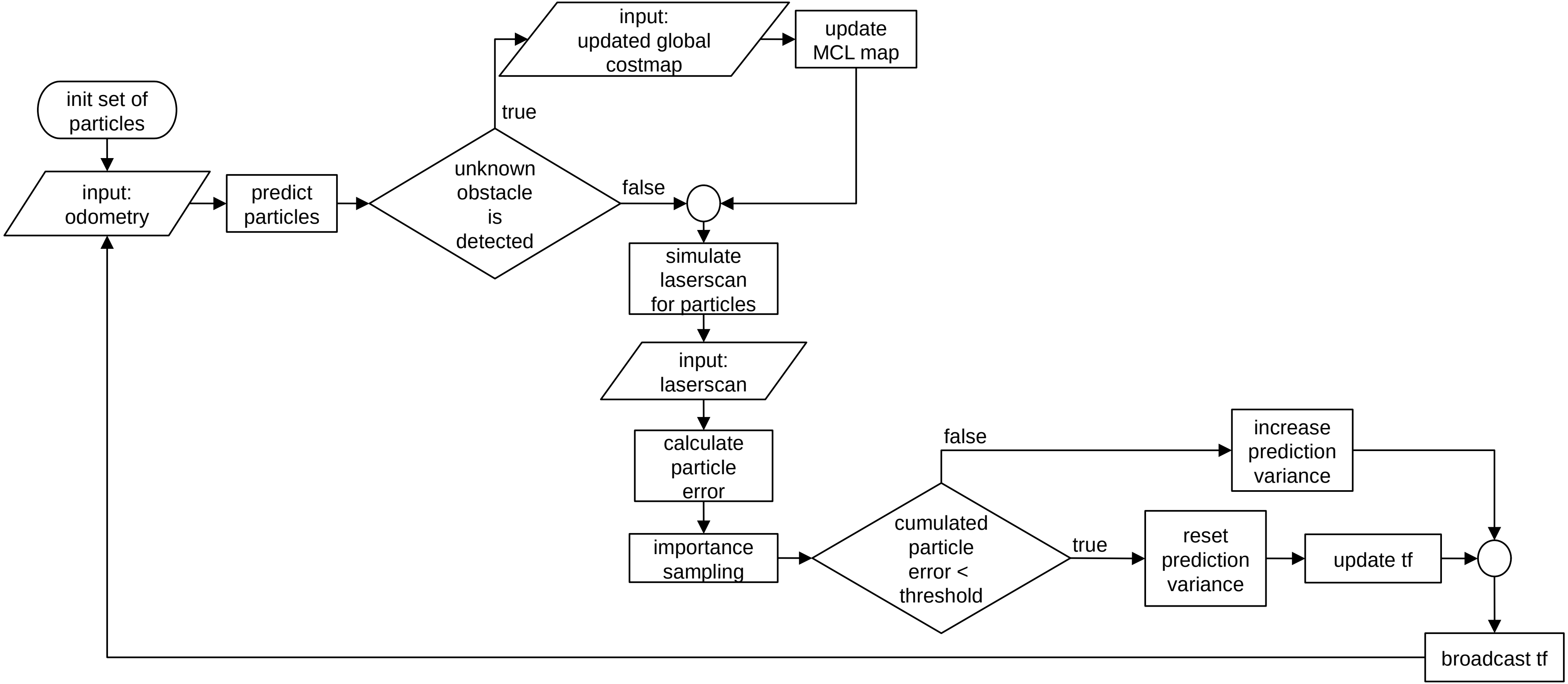
- Robot pose is represented by the distribution of particles

- **prediction**: move particles according to robot motion
- **update**:
  - particles which are likely to give the sensor measurements receive high importance weights
  - a new set of particles is generated by resampling particles with an high importance weight



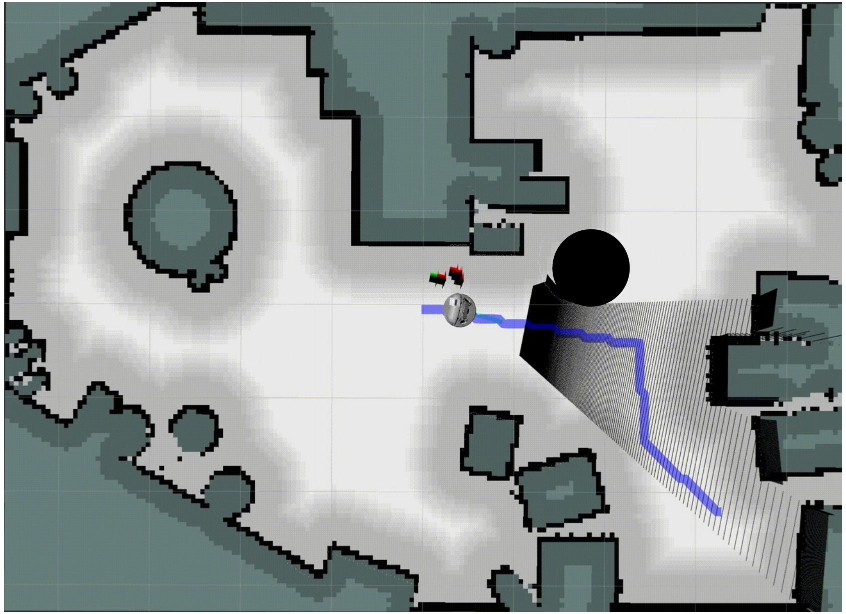
picture adapted from: Thrun, S. and Burgard, W. and Fox, D. and Arkin, R.C. (2005). Probabilistic Robotics. https://books.google.de/books?id=2Zn6AQAAQBAJ

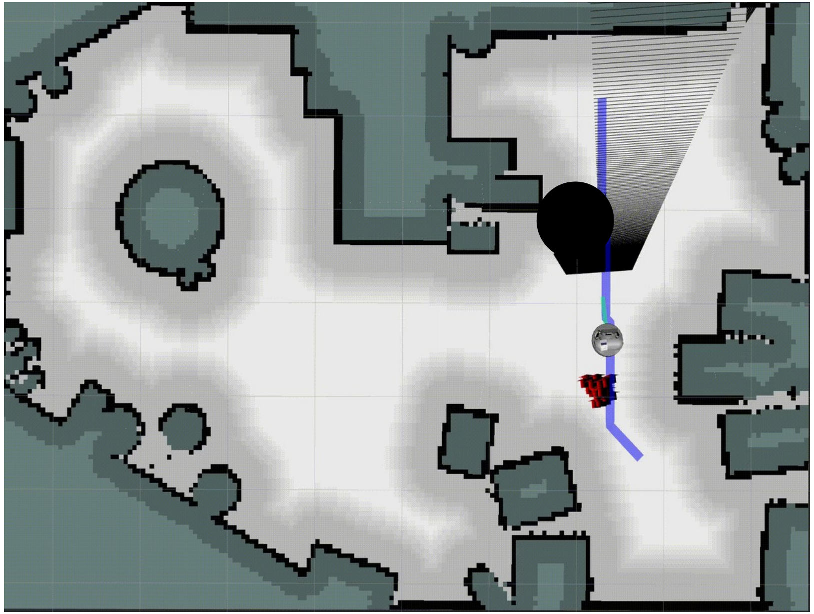
MCL - The Algorithm



MCL - Unknown Obstacles

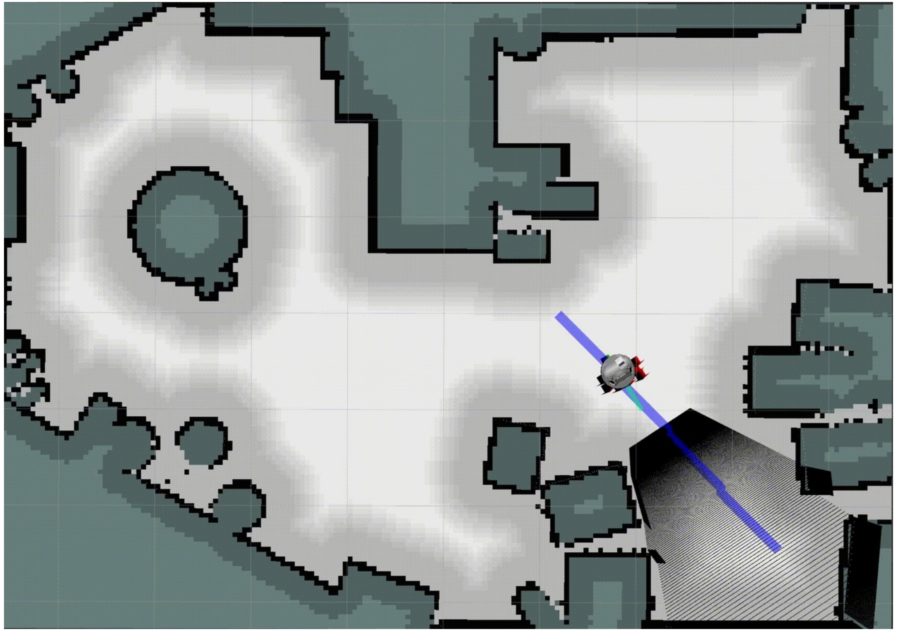
- obstacles detected by the robot and not included in the map make the localization difficult
- **solution:** do not update tf from /odom to /map if localization is inaccurate





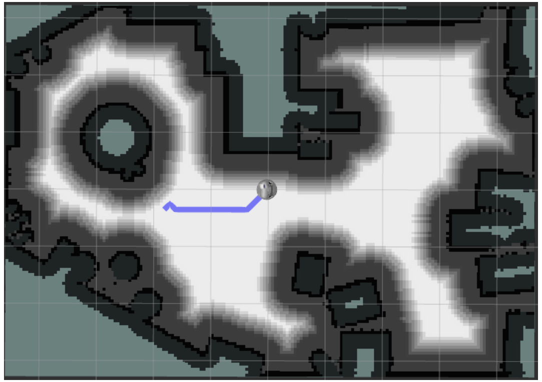
MCL - Adaptive Variance

- the variance of the gaussian noise is adapted to the performance of MCL
- performance of localization is bad
  - variance of gaussian noise is increased
  - particles spread out in order to capture the robot pose again

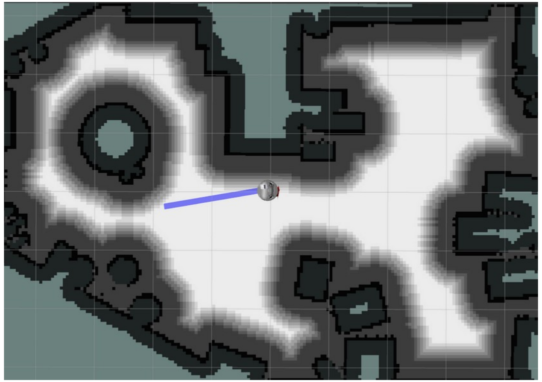


Global Planning - Bidirectional A*

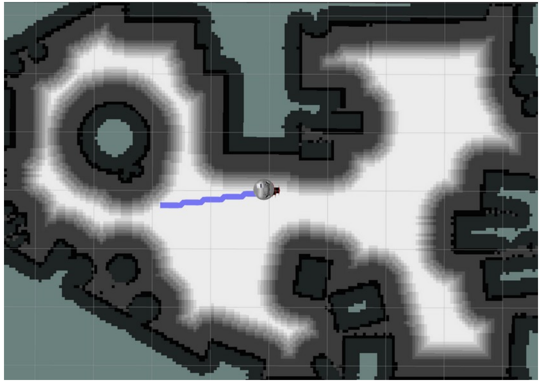
bidirectional search



edge cutting



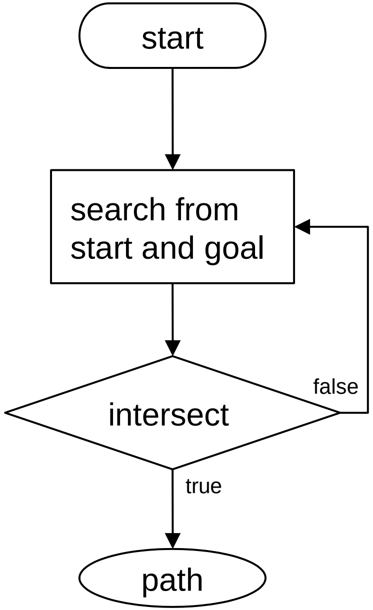
dense path

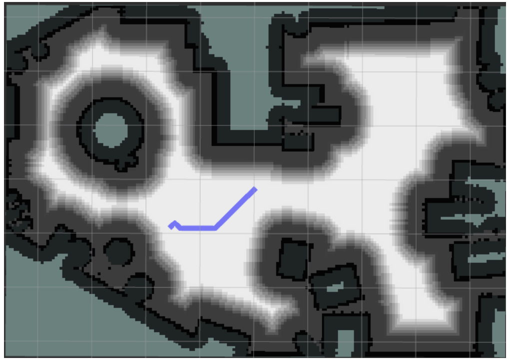


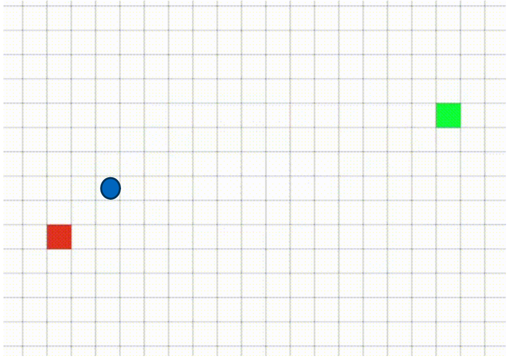
Global Planning - Bidirectional A*

- **processes**:
  - search from both start and goal
  - check intersection
  - connect two paths
- add costmap value in heuristic cost to maximize collision avoidance

```
cost_f = cost_g + cost_h
```



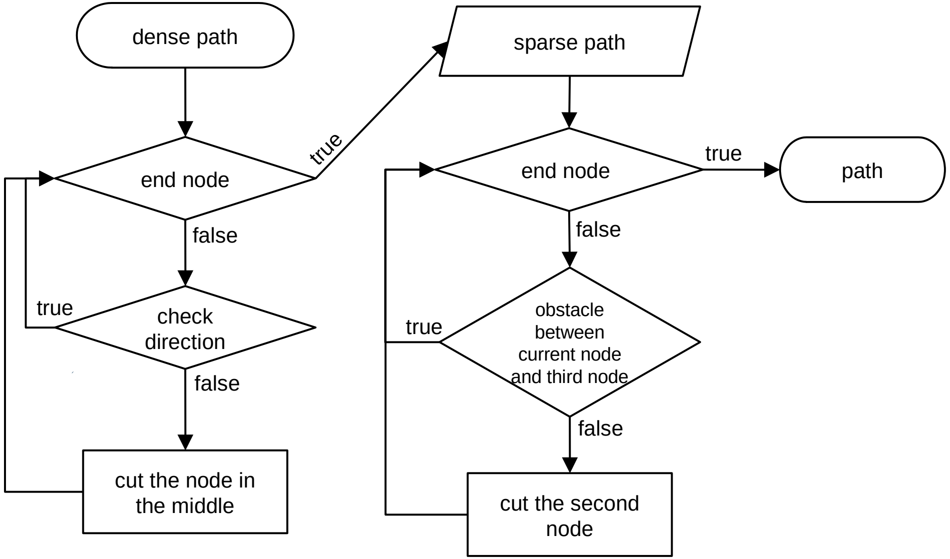


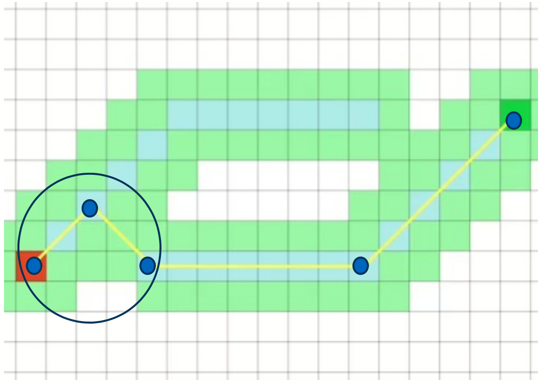


http://qiao.github.io/PathFinding.js/visual/

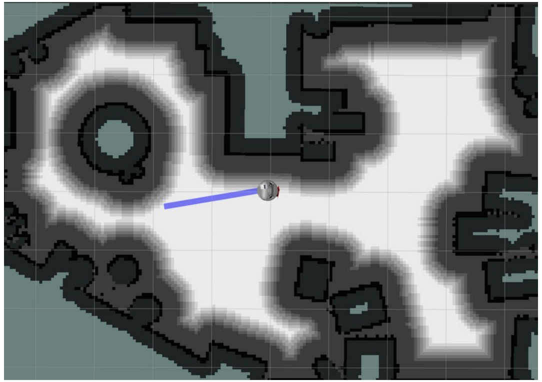
Global Planning - Edge Cutting

- to cut unnecessary edges



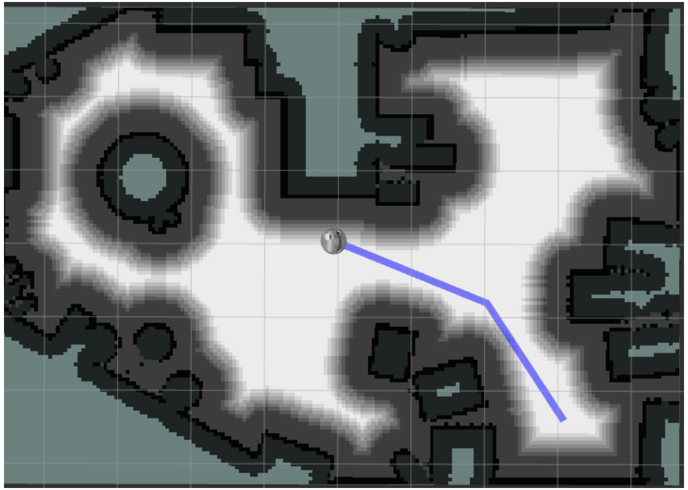


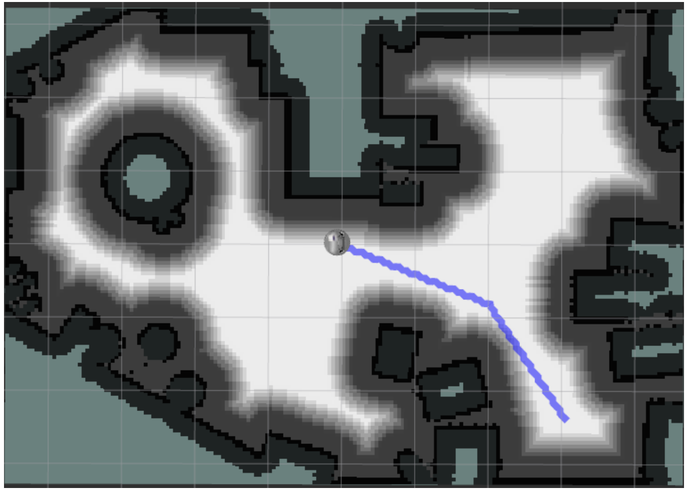
retrieved from http://qiao.github.io/PathFinding.js/visual/



Global Planning - Dense Path

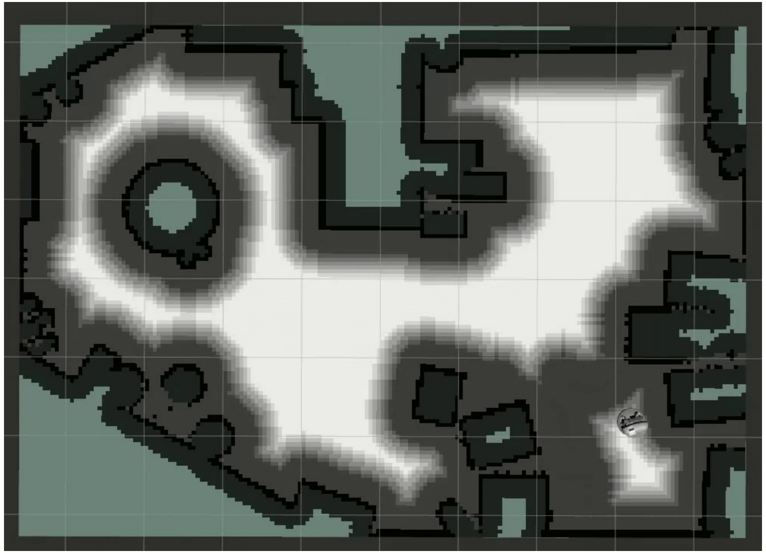
- project the sparse path from last step to grid map and get a dense path





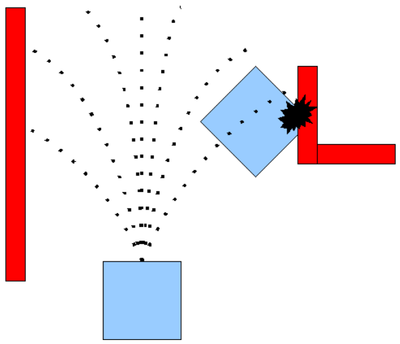
Global Planning - No Path Available

- **problem**: adaption of the global cost map might not allow the global planner to find a path even if there might be the possibility
- **solution**: call the service clear_map to reset global costmap to initial one

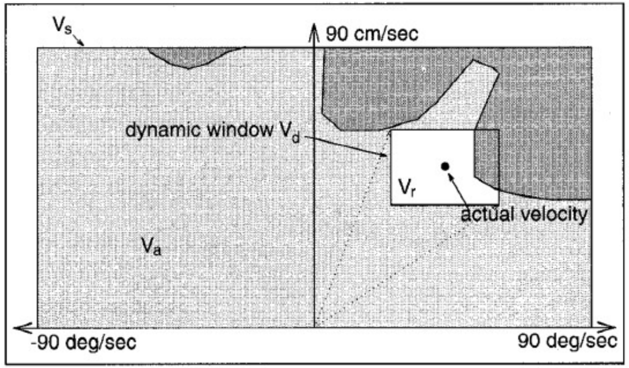


Local Planner - Dynamic Window Appr.

- online collision avoidance strategy for local planning
- samples circular trajectories from a search space (dynamic window)
- discards invalid trajectories
- selects the best trajectory based on a cost function
- publishes a linear and an angular velocity



retrieved from www.wiki.ros.org/dwa_local_planner

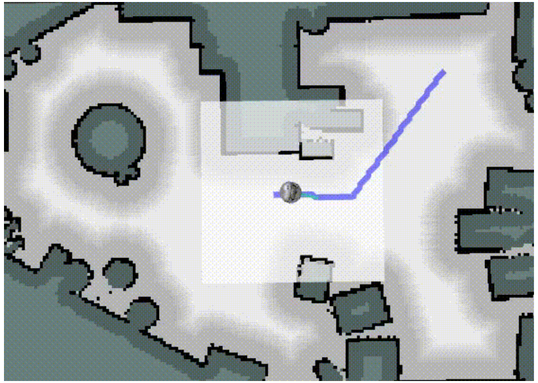


Thrun, S. and Burgard, W. and Fox, D. (1997). IEEE Robotics & Automation Magazine.
The Dynamic Window Approach for Collision Avoidance

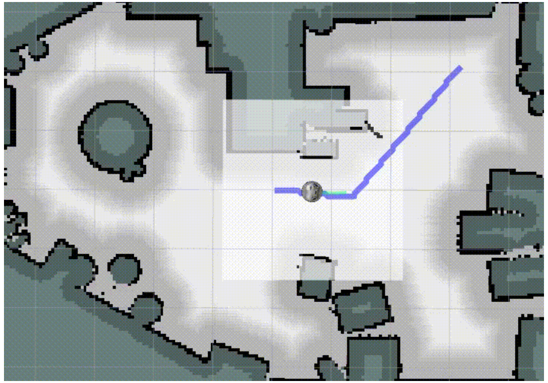
Local Planner - The Cost Function

- based on the sum of four individual costs:
  - cost based on linear velocity
  - cost based on angle towards the goal
  - cost based on proximity to the global path
  - cost based on proximity to obstacles
- each individual cost has its own gain factor
- change of behaviour based on change of gain factors

```
cost = cost_vel * gain_vel +
       cost_angle_to_goal * gain_angle_to_goal +
       cost_prox_to_path * gain_prox_to_path +
       cost_prox_to_obst * gain_prox_to_obst
```

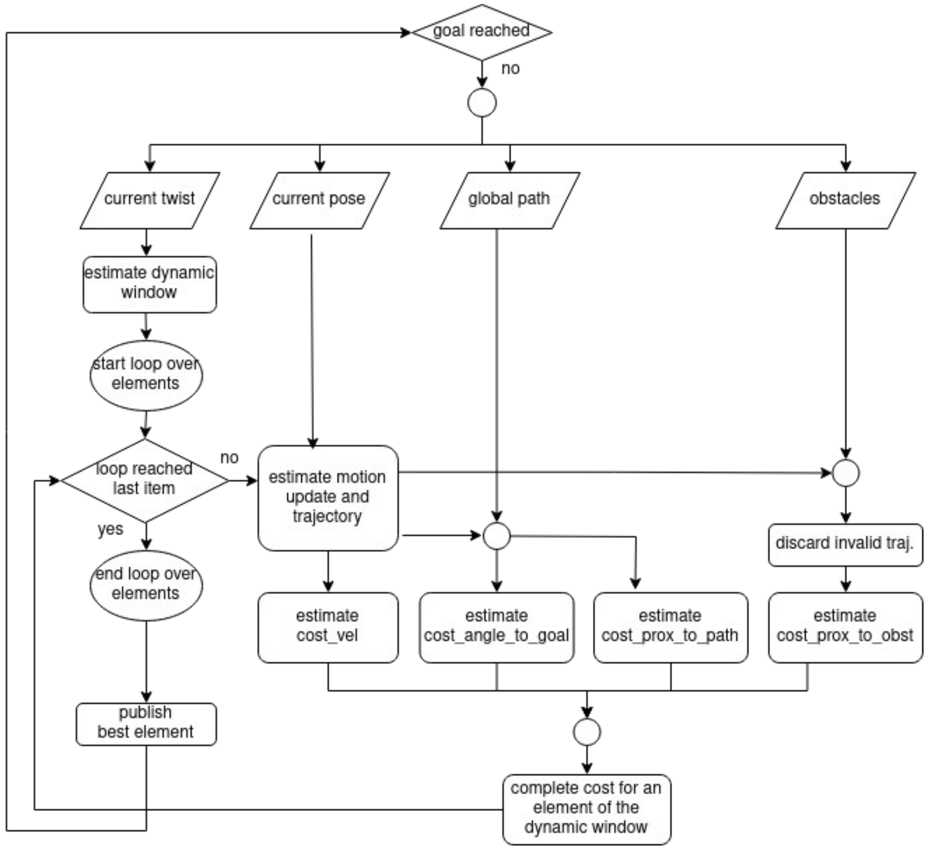


gain factors: 18, 12, 15, 15

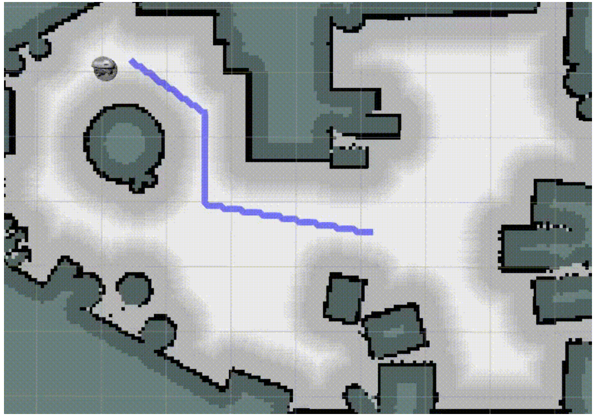


gain factors: 28, 2, 80, 1

Local Planner - The Algorithm

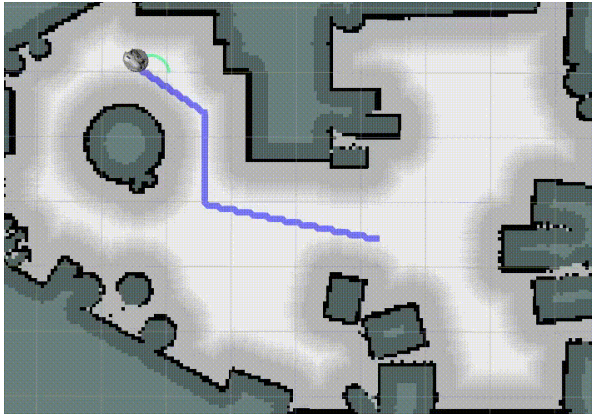


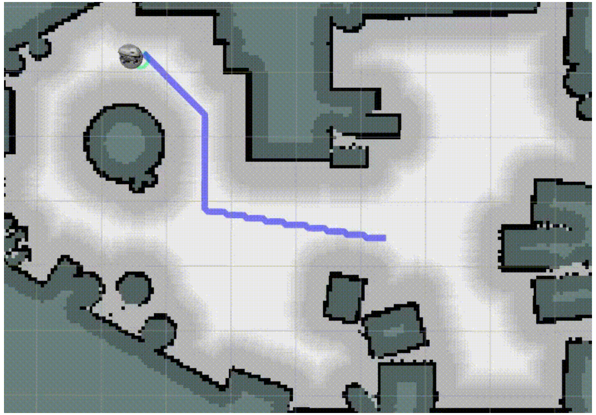
Local Planner - The Lookahead



lookahead of 1s

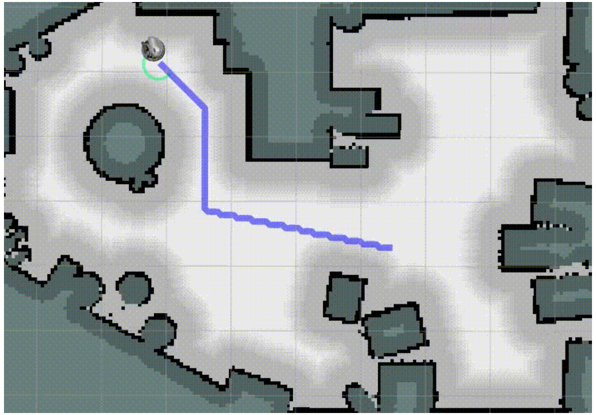
lookahead of 3s





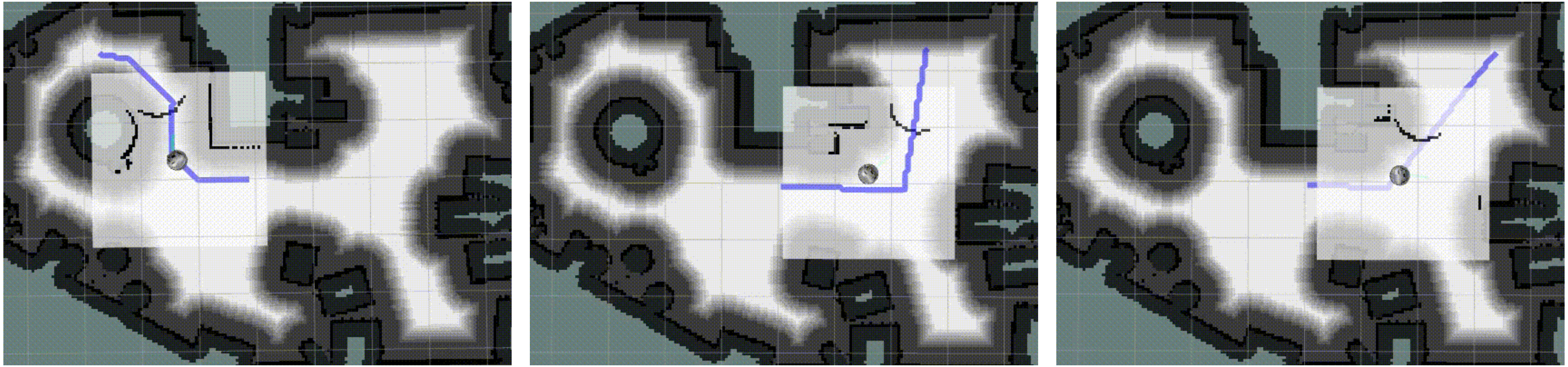
lookahead of 6s

lookahead of 10s



Local Planner - Recovery Behaviour

- local planner is also responsible for detection of critical situations
- necessary to react to dynamic changes of the environment
- measures to initialize a recovery behaviour:
  - small **linear velocity** for a certain amount of time
  - robots **circles** for a certain amount of time
  - estimated **execution time** of the path is exceeded



Challenges and Outlook

- challenges:
  - efficient implementation in Python
  - real-time capability of the system
  - memory leaking
  - message synchronization
  - integration of packages and adaptation of parameters

- next steps:
  - parallelize code to allow the use of more particles
  - try to make the navigation stack work outside of the simulation environment
  - think about more advanced recovery behaviours
  - diagnostics

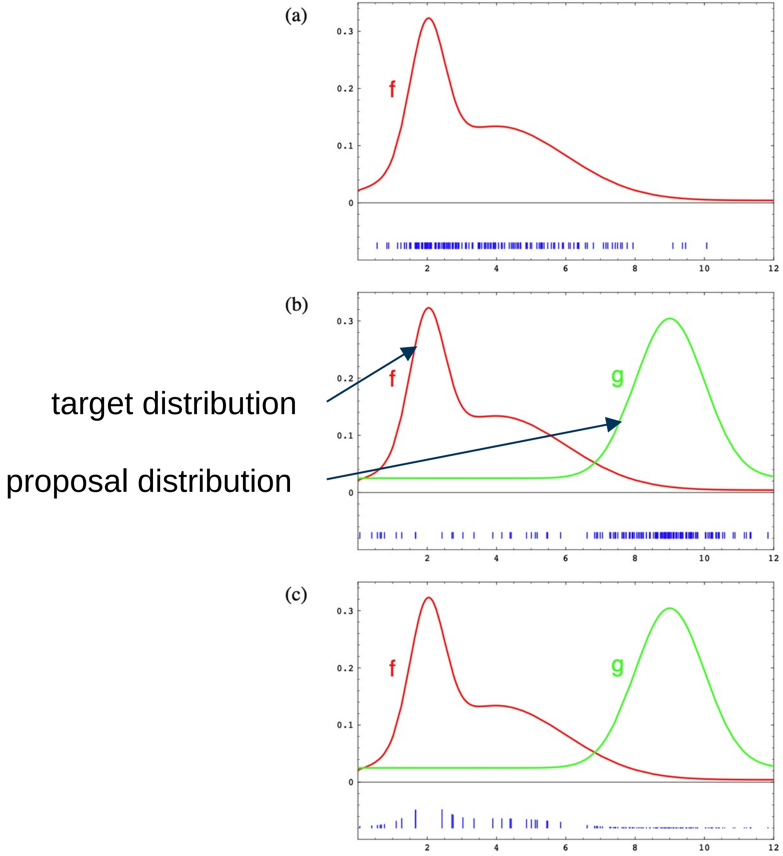
Thank you for your attention!

Live Demonstration
and Questions

Backup

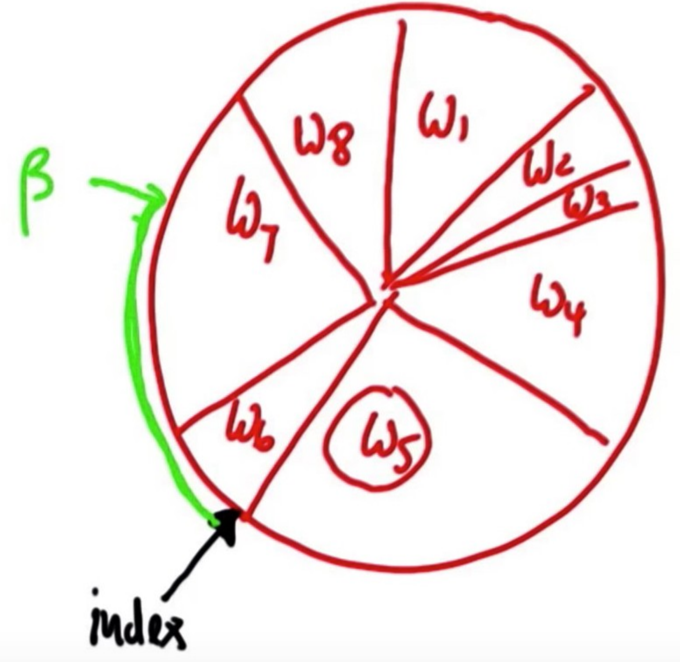
MCL - Importance Sampling

- **proposal distribution**: current particle distribution
- importance weights are assigned to each particle showing the consistency of the particles with the sensor measurement
- weighted particles converge towards the target distribution
- **target distribution:** updated particle distribution by resampling from the weighted particles



Thrun, S. and Burgard, W. and Fox, D. and Arkin, R.C. (2005). Probabilistic Robotics. https://books.google.de/books?id=2Zn6AQAAQBAJ

MCL - Resampling wheel



Karunakran, D. (2018.March.2014).Kidnapped vehicle project using Particle Filters-Udacity's Self-driving Car Nanodegree. Website. retrieved from: https://medium.com/intro-to-artificial-intelligence/kidnapped-vehicle-project-using-particle-filters-udacitys-self-driving-car-nanodegree-aa1d37c40d49