CourseWare Wiki

🏠 / b181 / courses / b4m36uir / labs / lab03

NAVIGATION ⊟

- B4M36UIR
- Classification
- Tasks
- Laboratory Exercises
  - Lab01 - Introduction to V-REP and Open-Loop Robot Locomotion Control
  - Lab02 - Exteroceptive sensing and Reactive-based Obstacle Avoidance
  - Lab03 - Grid and Graph-based Path Planning Methods
  - Lab04 - Incremental Path Planning
  - Lab05 - Randomized Sampling-based Algorithms
  - Lab06 - Curvature-constrained local planning in RRT
  - Lab07 - Asymptotically Optimal Randomized Sampling-based Path Planning
  - Lab08 - Multi-goal path planning and data collection path planning - TSP-like formulations
  - Lab09 - Data Collection Path Planning with Remote Sensing (TSPN)
  - Lab10 - Curvature-constrained Data Collection Path Planning ((D)TSPN)
  - Lab11 - Game Theory in Robotics
  - Lab12 - Game Theory in Robotics contd.
  - Lab13 - Game Theory in Robotics contd.
- Lectures
- For students
- Tutorials

ALL COURSES ⊟

- Winter 2018 / 2019
- Summer 2017 / 2018
- Older

B4M36UIR & BE4M36UIR

−Table of Contents

Lab03 - Grid and Graph-based Path Planning Methods

Motivations and Goals
Become familiar with grid based path planning
Understand the basic steps of the mobile robot navigation
Be able to implement simple path planning algorithms and deploy them in robot navigation
Tasks (teacher)
Task03 **(3 Points)** Implement selected search algorithm such as BFS, Dijkstra, and A* in grid-based path planning
Implement robot embodiment and extension of the grid map to configuration space
Lab resources
Task03 resource package

Occupancy Grid Maps

Occupancy grid maps represent an example of environment representation in probabilistic robotics which address the problem of generating maps from noisy and uncertain sensor measurement data, with the assumption that the robot pose is known. [1] The goal of the occupancy mapping is to estimate the posterior probability over maps given the data: $p(m|z_{1:t}, x_{1:t})$ where m is the map, $z_{1:t}$ is the set of measurements from time 1 to t, and $x_{1:t}$ is the set of robot poses from time 1 to t. Typically, the problem decomposes to estimation of $p(m_i|z_{1:t}, x_{1:t})$, where m_i is a single cell of the occupancy grid map.

In general, the grid maps divide the action space of the robot into discrete cells which carry the information about the occupancy of the designated area (occupied, free, unknown). The cell dimensions are user defined and usually is a balance between the environment size, that influence the overall memory consumption, and geometrical precision, that should be proportional to the robot dimensions. The data from the sensors are fused to the occupancy grid using either Bayesian or Non-bayesian way.

The Bayesian occupancy grid update is defined as:

$$P(m_i = occupied|z) = \frac{p(z|m_i = occupied)P(m_i = occupied)}{p(z|m_i = occupied)P(m_i = occupied) + p(z|m_i = free)P(m_i = free)},$$

where $P(m_i = occupied|z)$ is the probability of cell m_i being occupied after the fusion of the sensory data; $P(m_i = occupied)$ is the previous probability of the cell being occupied and $p(z|m_i = occupied)$ is the model of the sensor which is usually modeled as:

$$p(z|m_i = occupied) = \frac{1 + S^z_{occupied} - S^z_{free}}{2},$$

$$p(z|m_i = free) = 1 - p(z|m_i = occupied),$$

where $S^z_{occupied}$ and S^z_{free} are the sensory models for the occupied and free space respectively.

For the LIDAR sensor, the simplified sensory model yields:

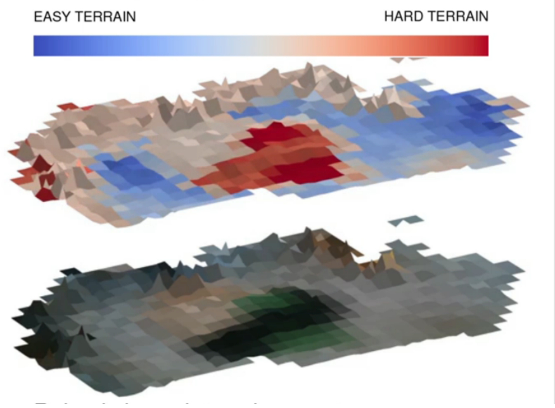
$$S^z_{occupied} = \begin{cases} 1 & \text{for} \quad d \in [r - \epsilon, r + \epsilon] \\ 0 & \text{otherwise} \end{cases}, \qquad S^z_{free} = \begin{cases} 1 & \text{for} \quad d \in [0, r - \epsilon] \\ 0 & \text{otherwise} \end{cases},$$

where r is the distance of the measured point and ϵ is the precision of the sensor. I.e. it is assumed that the space between the sensor and the measured point is free and the close vicinity of the measured point given by the sensor precision is occupied.

An example of the occupancy grid mapping: https://www.youtube.com/watch?v=zjl7NmutMIc

Grid-Based Planning for Mobile Robot Navigation

The grid representation of the world introduces discretization of the robot's action space into individual cells. These cells carry the information whether they are free or occupied and possibly further information including traversability, or suitability of the area for foot placement, etc. An example of a terrain map and its underlying representation as a traversability map is visualized in the following figure.



A typical pipeline of the grid-based path planning consists of following steps:

1. **Obstacle growing** - The robot has non-zero dimensions that represent its embodiment which has to be taken into account when planning. Moreover, the plan execution in real experiments has a limited precision which further enlarges demands on the clearance to the obstacles for the robot to safely navigate the environment. The planning with a large robot in grid environment is complex, therefore the trick is to grow the borders of the obstacles by a diameter of the robot and plan in the resulting map that ensures a sufficient clearance to the obstacles.
2. **Path planning** - Proper selection of the grid path planning algorithm depends on the performance metrics and usage of the plan. Among others, following grid-based planning algorithms are worth mentioning:
    1. Breadth First Search (BFS), Depth First Search (DFS)
    2. Dijkstra
    3. A*
    4. Jump Point Search (JPS)
    5. Theta* (any-angle path planning - returns shortest path)
    6. ARA* (Any-time repairing A*)
    7. D* lite
    8. Floyd-Warshall
3. **Trajectory smoothing** - The grid-based planners usually return path as a list of neighboring cells (?which of them doesn't?). The trajectory smoothing is for simplifycation of the path representation, possible shortening of the path and taking into consideration the path following approach deployed on the particular mobile robot. Typical approach to trajectory smoothing is to connect the neighboring segments one by one using straight-line segments up to the point where the straight-line segment collide with an obstacle (grown obstacle of course) and then follow with another straight-line segment.

Evaluation of path planning methods in robotics

Typical steps of planning algorithm evaluation in robotics consists of following steps

1. Software testing of the planning algorithm without execution
2. Simulation of execution with perfect information and perfect execution
3. Simulation in realistic simulator with perfect information and imperfect execution given by simulated physics
4. Deployment on a real robot with imperfect information and imperfect execution

For evaluation of the path, different metrics can be used. In general following properties are examined:

- **Feasibility** - whether the path is feasible for particular robot given its embodiment
- **Length** - the overall path length
- **Planning time** - in the real-time applications and multi-goal path planning, the overall planning time is a crucial parameter
- **Clearance** - how far from the obstacles the robot navigate. Low distance may result in robot damage in case of inaccurate path execution

[1] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics. MIT press, 2005.

courses/b4m36uir/labs/lab03.txt · Last modified: 2018/10/21 22:19 by cizekpe6