

Research topics: Navigation + Algorithms (SLAM)

Useful links

- [Gitlab](#)
- [Autonomous Navigation with Deep Reinforcement Learning Using ROS2](#) - Useful YouTube Video
- [What is SLAM \(Video Explanation\)](#)

Articles

- [2007] <https://www.witpress.com/Secure/elibrary/papers/SAFE07/SAFE07030FU1.pdf>
 - Autonomous exploration for search and rescue robots
- [2019] <https://isprs-archives.copernicus.org/articles/XLII-2-W13/857/2019/isprs-archives-XLII-2-W13-857-2019.pdf>
 - Reinforcement learning and SLAM based approach
- [2019] <https://www.tandfonline.com/doi/full/10.1080/23311916.2019.1632046>
 - A comprehensive study for robot navigation techniques
- [2023] <https://arxiv.org/pdf/2308.00331.pdf>
 - Target Search and Navigation in Heterogeneous Robot Systems with Deep Reinforcement Learning
- [2023] <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10062604/>
 - A search and rescue robot search method based on flower pollination algorithm and Q-learning fusion algorithm

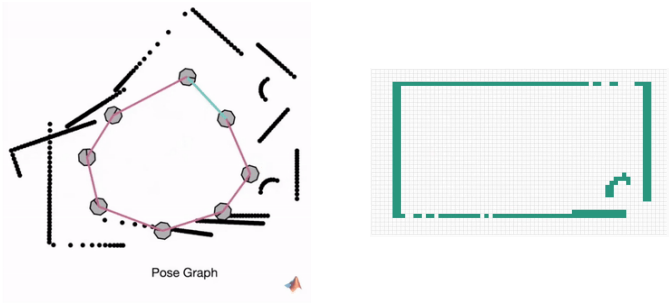
Main types of navigation algorithms (target search)

- traditional algorithms:
 - A*
 - Artificial potential field (APF)
 - Dijkstra's algorithm
- intelligent algorithms
 - Genetic Algorithm (GA)
 - Ant colony optimization (ACO) - probabilistic?
 - Particle swarm algorithm
- reinforcement learning algorithms
 - Optimises actions to obtain the maximum reward
 - To quote: 'artificial intelligence algorithms that learn to achieve goals in uncertain, potentially complex environments by training a machine learning model to make a series of decisions that enable the robot to choose the actions to perform based on the environment'

A* VS Dijkstra algorithms

- Dijkstra
 - Only requires distance from starting node
 - Checks every possible route
 - Not efficient in general
 - Works in scenario of no possible paths
- A*
 - Requires both distance from starting and end node
 - More efficient and uses less resources
 - Used in google map

SLAM

- Purpose: construct a map of an unknown environment while keeping track of a robot's **pose** within it
 - Pose is defined as the position and orientation of the robot, eg. x , y , θ
 - Visual SLAM: Uses images acquired from cameras
 - LiDAR SLAM (light detection and range): Uses laser sensor (or distance sensor)
 - Ideal situation (perfect **odometry**): Take measurements of how far away obstacles are to build a map, use perfect odometry to determine where the robot is in the map
 - In reality (large uncertainty in odometry): Each measurement will be inaccurate since the estimated pose will be different to the real pose
 - Odometry is the use of data from sensors to estimate the change in position of a vehicle over time relative to a starting point
 - Pose graph optimization
 - => improves estimated poses to create map
 - Binary Occupancy Grid
 - => use this grid for navigation
- 
- Pose graph optimization
 - Nodes: poses
 - Edges: spatial constraints or measurements between the poses
 - Loop closure essentially 'pulls' the nodes together through 'tension'
 - Loop closure requires an absolute sensor, not relative
 - Binary Occupancy Grid
 - Is a grid of cells which represents the environment
 - True value represents occupied space and false value represents empty space
 - Grid is used for navigation

Main points

- 2 main points
 - Need a way to navigate through a given map
 - Need a way to create the map and determine where the robot currently is on the map
- 3 main types of navigation algorithms
 - Go through examples briefly, explain reinforcement learning
 - Go into detail of A* and Dijkstra's
- SLAM
 - What is SLAM and its purpose
 - Types of SLAM
 - Brief explanation on how it works in by comparing ideal situation vs reality
 - Briefly mention how problems are overcome in reality (pose graph optimization)
 - Briefly mention binary occupancy grid
- What should we do
 - Use SLAM for creating the map and determining pose
 - Choose an algorithm for navigation