→ Navigation

The knowledge about the environment and situation is usually only partial and uncertain.

- Where am I → localization
- Where have I been → mapping
- Where should I go → decision
- What's the best way to go there → path planning
- How do I get there → path following + object avoidance (motion)

→ Path planning

Task of computing a path without collisions.

- Path → only geometric considerations
- Trajectory → includes geometric and time considerations
- Motion → mobile robots and manipulators (considers constraints like object avoidance)

• Configuration space (C-space): set of possible valid configurations (poses) of the robot. (search space + allowable paths)

• Free space (F-space): set of valid configurations that do not intercept obstacles in the environment (robot shape).

Assumptions: robot is a dot, good enough representation of a map to compute a path, there is a good estimation of robot pose.

→ Environment representation

Two ways: topological and geometric maps.

↳ Geometric: continious geometric description

↳ Topological: graphs with nodes and arcs.

Path planning algorithms are only applicable to discrete maps. Decomposition

- Road map: identify routes within the free space.
- Cell decomposition: divide space into cells that can be free or occupied.
- Potential field: apply math function over the space.

↳ Visibility graph

Objects are treated as poligons.

With the detected nodes (polygon vertices and start/finish) and edges (connected nodes that don't go through object) a connectivity graph (visibility) is generated.

Problem: may cause the robot to move too close to obstacles

Solution: grow obstacles.

↳ Voronoi diagrams

For each point in free space its distance to nearest obstacle is computed. Then, set of equidistant points from the nearest two or more obstacle boundaries are extracted.

Result is path of maximum distance from obstacles.

→ Cell decomposition

Divide space into simple connected regions called cells. Adjacency graph of free cells.

↳ nodes: free cells

↳ edges: edge every pair of nodes whose cells are adjacent.

Two possible decomposition methods:

↳ Exact cell decomposition:

Free cells correspond exactly to free space. The free space F is divided into non-overlapping convex cells.

A connectivity graph can be constructed where nodes represent the free cells and every adjacent cell is connected by an edge.

Good for sparse environments.

↳ Approximate cell decomposition:

Some free space is included in partially occupied cells. Free space is divided into non-overlapping cells. These have simpler shapes. Two approaches:

- variable size: decomposing with a variable sized cell with rectangular shape. Some areas can be free but considered as occupied.
- fixed size: Same but using square shape.

Cell size is not dependent on the obstacle size in the environment. Low computational complexity for path planning.

- quad-tree: variable-size square cells.

Partially occupied cells are subdivided until a given granularity. Resolution can vary and depend on free space.

→ Search algorithms

- **Wavefront expansion:** path in fixed-size cell arrays. Starting at the goal, many in each adjacent cell its distance to goal (Manhattan). This is repeated until start pos is reached. Planner links cells together that are adjacent and closer to goal.

- **Dijkstra's algorithm:** start marking adjacent nodes with cost to get to them. Goes to lowest cost and does the same. If all visited, go to lowest and continue. Path to goal may be obtained starting from goal and following edges pointing to lower cost.

- **A* algorithm:** Use a cost function to many choices, using the costs of going to nodes and the estimates to the goal (heuristic function).

→ Potential field path planning (downhill)

Think of the robot as a particle in a potential field, defining a potential function over the free space (global minimum at goal). Define high potentials for obstacles, the robot will follow the steepest descent of the potential function.

The goal generates an attractive potential, obstacles a repulsive potential.

Problems: oscilating motion, trapped in local min.

→ Obstacle avoidance

Environment is not fully modelled, might change dynamically.

Avoid colisions while pursuing goal, relying on info about goal, current pos and recent sensory info.

Covered methods:

- Bug 1: Go towards goal, if reached finish.. Do a full turn around obstacle and show closest point to goal. Go to that point, repeat.

Inefficient but does the job.

- Bug 2: Go towards goal, if reached finished. If obstacle, contour until reach line between start and goal. repeat, keeping same side.

More efficient but can fail.

- Vector field histogram: local map around robot. (occupancy grid). Each cell holds confidence that there is an obstacle. Grid is updated with recent sensory data. Histogram grid is converted to polar histogram, storing statistical info. A threshold converts that histogram into binary diagram with possible regions. The one chosen depends on target.