Advanced Robot Navigation: Lecture 8

Lecture Notes

Victor Risager

October 3, 2023

1 Recap

Bayes filter branch out into particle filter and kalman filter. EKF local linear approximation.
These filters are used for localisation.

1.1 Map building

Landmark: a landmark is a distinguishable feature in the environment. Camera → spot in a certain color Leads into **SLAM** which can brach out into:

- EKF, include landmarks in state → landmarks^2
- PF SLAM → FastSLAM → Linear scaling

Action model ← **mobility model** which is applied to close a time gap. Other times we do a perception update, where we use a sensor model, which incorporates information
Only update the perceptions if there is a new measurement ready. You can use action updates and perception updates independently.

1.2 Random walk model

$$x_{n+1} = x_n + v_n \tag{1}$$

where v_n is some measurement noise. A very imprecise model
if your odometry malfuncitons, then you cannot just use your motion model in the action update.

2 Path Planning

Initial assumptions:

- Holonomic
- Point robot

We will apply Minkowski-sum later on.

2.1 Holonomic robots

A car is non-holonomic, because the number of global degrees of freedom, are higher than the free configuration degrees of freedom.
A train is holonomic, because we have reduced the global degrees of freedom.
you "expand" the obstacle to confine the width of the robot and make that our new obstacle.
If we dont do the conservative approximation of the robot by taking e.g. a circle of the robot.

2.2 C-Obstacles

How do we map the workspace obstacles to the C-space?
We define an anchor point on the robot, which is the point that abstracts the robot, and that point will be outside the minkowsky sum. we check if:

$$p + r \in O \tag{2}$$

where P is the translation of the r vector in the Robot.
Then

$$\exists o \in O, P + r = o \rightarrow p = o - r \tag{3}$$

where

$$C = \{P = o + r | o \in O, r \in -R\} \tag{4}$$

matlab: Polyshape which gives a polygonal shape based on a set of coordinates.
The minkowsky sum is a 3'rd party library. The minkowsky sum is computed with the best possible heading, and is therefore not as conservative.
Discretisising the headings and recomputing the minkowsky sum, gives a set of maps which combined on top of each other, gives a way to search position and heading.
The star algorithm produces the same result.

3 Graph Search

A graph

$$G = (V, E) \tag{5}$$

where $V = \{v_1, ..., v_n\}$ is vertecies, and $E = \{(v_i, v_j) | v_i, v_j \in V\}$ is edges.
Let $W = E \rightarrow R_+$ be the positive weights of an edge.
A path is a sequence of edges. where the cost of the path is

$$c(P) \sum_j w(e_j) \tag{6}$$

Conceptually

- Vertices are junctions
- Edges are roads that connect junctions
- W is the travel length of roads/edges

3.1 Visibility Graph

Workload is quite modest. The path gracefully touches the obstacles.

3.2 Astar

Heuristic

3.3 D*

An incremental replanning of version of A*

3.4 Value itteration

Algebraic version to graph search.

$$E(v) = C(v, goal) \tag{7}$$

which is optimal.
From v to *goal* there may be some stepping stone through v' which is part of some path.

Sidenote Bellmann optimality principle. If a path between v and *goal* is optimal, then every subpath of this path is also optimal.
$\boxed{\text{If } w(e) + c(p) = E(v) \text{ then } c(p) = E(v'), \text{ thus } E(v) = w(e) + E(v')}$
Thus generally

4 Potential Fields

let

$$U : Q \to \mathbb{R} \tag{8}$$

$$\nabla U : Q \to \mathbb{R}^n \tag{9}$$

$$\nabla U(q) = v \tag{10}$$

So we let the velocity be proportional to force

$$\frac{dq}{dt} = -\zeta \cdot \nabla U(q) \tag{11}$$