Introduction to Mobile Robotics

Path Planning and Collision Avoidance

Cyrill Stachniss

Motion Planning

Latombe (1991):
“...eminently necessary since, by definition, a robot accomplishes tasks by moving in the real world.”

Goals:
- Collision-free trajectories.
- Robot should reach the goal location as fast as possible.

... in Dynamic Environments
- How to react to unforeseen obstacles?
  - efficiency
  - reliability

  Dynamic Window Approaches
  [Simmons, 96], [Fox et al., 97], [Brock & Khatib, 99]
  Grid map based Planning
  [Konolige, 00]
  Nearness Diagram Navigation
  [Minguez et al., 2001, 2002]
  Vector-Field-Histogram+
  [Ulrich & Borenstein, 98]
  A*, D*, D* Lite, ARA*, ...

Dynamic Window Approach (1)
- Here: robot moves on circular arcs.
- Motion commands \((v,\omega)\).
- Which \((v,\omega)\) are admissible?
- Collision Avoidance: Check with geometric operations which trajectories are collision-free!
Dynamic Window Approach (2)

- Example for the Search-Space:

  ![Image]

- $V_s$ = all possible speeds of the robot.
- $V_{a}$ = obstacle free area.
- $V_{d}$ = possible speeds based on possible accelerations.

$$Space = V_s \cap V_{a} \cap V_{d}$$

Dynamic Window Approach (3)

- How to choose $<v,\omega>$ now?
- Steering commands are chosen by a heuristic navigation function.
- This function tries to minimize the travel-time by:
  “drive fast in the correct direction”.

Dynamic Window Approach (4)

- Heuristic navigation function.
- Planning restricted to $<x,y>$-space.
- No planning in the velocity space.

Navigation Function: [Brock & Khatib, 99]

$$NF = \alpha \cdot vel + \beta \cdot nf + \gamma \cdot \Delta nf + \delta \cdot goal$$

Dynamic Window Approach (4)

- Heuristic navigation function.
- Planning restricted to $<x,y>$-space.
- No planning in the velocity space.

Navigation Function: [Brock & Khatib, 99]

$$NF = \alpha \cdot vel + \beta \cdot nf + \gamma \cdot \Delta nf + \delta \cdot goal$$
Dynamic Window Approach (4)

- **Heuristic** navigation function.
- Planning restricted to \( <x,y>-space \).
- No planning in the velocity space.

**Navigation Function:** [Brock & Khatib, 99]

\[
NF = \alpha \cdot vel + \beta \cdot nf + \gamma \cdot \Delta nf + \delta \cdot goal
\]

- Maximizes velocity.
- Considers cost to reach the goal.

Dynamic Window Approach (5)

- **Reacts fast.**
- Low CPU power required.
- Guides a robot on a collision free path.
- Successfully used in a lot of real world experiments.
- But not always good trajectories.
- Local Minima, sometimes no solution!
Problems of DWAs

\[ NF = \alpha \cdot vel + \beta \cdot nf + \gamma \cdot \Delta nf + \delta \cdot goal \]

Robot's velocity.

Preferred direction of \( NF \).

13

14

15

16
The robot drives too fast at $c_0$ to enter corridor facing south.
Problems of DWAs

- Same situation as in the beginning.
  - DWAs have problems to reach the goal.

- Robot does not slow down early enough to enter the doorway.

Robot Path Planning with A*

- What about using A* to plan the path of a robot?
  - Finds the shortest path
  - Requires a graph structure
  - Limited number of edges
  - In robotics: planning on a 2d occupancy grid map

A*: Minimize the estimated path costs

- $g(n) = $ actual cost from the initial state to $n$.
- $h(n) = $ estimated cost from $n$ to the next goal.
- $f(n) = g(n) + h(n)$, the estimated cost of the cheapest solution through $n$.
- Let $h^*(n)$ be the actual cost of the optimal path from $n$ to the next goal.
- $h$ is admissible if the following holds for all $n$:
  $$ h(n) \leq h^*(n) $$
- We require that for A*, $h$ is admissible (the straight-line distance is admissible in the Euclidean Space).
Example: Path Planning for Robots in a Grid-World

Deterministic Value Iteration

- To compute the shortest path from every state to one goal state, use (deterministic) value iteration.
- Very similar to Dijkstra’s Algorithm.
- Such a cost distribution is the optimal heuristic for A*.

Typical Assumption in Robotics for A* Path Planning

- A robot is assumed to be localized.
- Often a robot has to compute a path based on an occupancy grid.
- Often the correct motion commands are executed (but no perfect world!).

Is this always true?

Problems

- What if the robot is slightly delocalized?
- Moving on the shortest path guides often the robot on a trajectory close to obstacles.
Convolve the Grid Map

- Convolution blurs out the map.
- Obstacles are assumed to be bigger than in reality.
- Perform an A* search in such a convolved map.
- Robots keeps distance to obstacles and moves on a short path!

Example: Map Convolution

- 1-d environment, cells $c_0, \ldots, c_5$

![Graph showing before and after convolution]

- Cells before and after 2 convolution runs.

Convolution

- Consider an occupancy map. Then the convolution is defined as:

  \[
  P(occ_{x_i,y}) = \frac{1}{4} \cdot P(occ_{x_{i-1},y}) + \frac{1}{2} \cdot P(occ_{x_i,y}) + \frac{1}{4} \cdot P(occ_{x_{i+1},y})
  \]

  \[
  P(occ_{x_0,y}) = \frac{2}{3} \cdot P(occ_{x_{0},y}) + \frac{1}{3} \cdot P(occ_{x_1,y})
  \]

  \[
  P(occ_{x_{n-1},y}) = \frac{1}{3} \cdot P(occ_{x_{n-2},y}) + \frac{2}{3} \cdot P(occ_{x_{n-1},y})
  \]

- This is done for each row and each column of the map.
- “Gaussian blur”

A* in Convolved Maps

- The costs are a product of path length and occupancy probability of the cells.

- Cells with higher probability (e.g. caused by convolution) are shunned by the robot.

- Thus, it keeps distance to obstacles.

- This technique is fast and quite reliable.
Key Ideas of the Presented Approach (5d-Planning)

- Plans in the full \( <x,y,\theta,v,\omega> \) configuration space using A*.
  \( \Rightarrow \) considers the robot's kinematic constraints.

- Generates a sequence of steering commands to reach the goal location.

- Maximizes tradeoff between driving time and distance to obstacles.

The Search Space (1)

- What is a state in this space?
  \( <x,y,\theta,v,\omega> = \) current position and speed of the robot

- How does a state transition look like?
  \( <x_1,y_1,\theta_1,v_1,\omega_1> \rightarrow <x_2,y_2,\theta_2,v_2,\omega_2> \)
  with motion command \( (v_2,\omega_2) \) and
  \( |v_1-v_2| < a_v, |\omega_1-\omega_2| < a_\omega \).
  Pose of the Robot is a result of the motion equations.

The Search Space (2)

**Idea:** Search in discretized \( <x,y,\theta,v,\omega> \)-space.

**Problem:** The search space is too huge to be explored within the time constraints (.25 secs for online control).

**Solution:** Restrict the full search space.

The Main Steps of Our Algorithm

1. Update (static) grid map based on sensory input.

2. Use A* to find a trajectory in the \( <x,y> \)-space using the updated grid map.

3. Determine a restricted 5d-configuration space based on step 2.

4. Find a trajectory by planning in the restricted \( <x,y,\theta,v,\omega> \)-space.
Updating the Grid Map

- The environment is represented as a 2d-occupancy grid map.
- Use convolved map.
- All detected obstacles are added.
- We reset cells discovered free.

Find a Path in the 2d-Space

- Use A* to search for the optimal path in the 2d-grid map.
- Use heuristic based on a deterministic value iteration within the static map.

Restricting the Search Space

**Assumption:** The projection of the 5d-path onto the $<x,y>$-space lies close to the optimal 2d-path.

**Therefore:** Construct a restricted search space (channel) based on the 2d-path.

Space Restriction

- Resulting search space = $<x, y, \theta, v, \omega>$ with $(x,y) \in$ channel.
- Choose a subgoal lying on the 2d-path within the channel.
Find a Path in the 5d-Space

- Use A* in the restricted 5d-space to find a sequence of steering commands to reach the subgoal.

- To estimate cell costs: perform a deterministic 2d-value iteration within the channel.

Examples

Timeouts

- Online robot control:
  new steering command every .25 secs.

  Abort search after .25 secs.

  How to find an admissible steering command?

Alternative Steering Command

- Previous trajectory still admissible? ➔ OK

- If not, drive on the 2d-path or use DWA to find new command.
Timeout Avoidance

- Reduce the size of the channel if the 2d-path has high cost.

Examples

B21r robot Albert.  Planning state.

Comparison to the DWA (1)

- DWAs often have problems entering narrow passages.

DWA planned path.  Our Approach.

Comparison to the DWA (1)

- DWAs often have problems entering narrow passages.

DWA planned path.  Our Approach.
Comparison to the DWA (2)

The presented approach results in significantly faster motion when driving through narrow passages!

Comparison to the Optimum

- Channel: with length=5m, width=1.1m we are close to the optimal solution.

Summary

- New approach to reactive collision avoidance.
- Considers the robot’s kinematic constraints and plans in the velocity space.
- Shows better results than the DWA in a variety of situations.
- The quality of the trajectory scales with the performance of the underlying hardware.
- The resulting paths are often close to the optimal ones.

What’s Next?

- More complex vehicles (e.g., cars).
- Moving obstacles, motion prediction.
- Approximative Search (ARA*: Not-admissible heuristics for faster planning).
- ...