Introduction to Mobile Robotics

Path Planning and Collision Avoidance

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.

Discretization of Continuous Maps
Extract a discrete search space / graph from the continuous map.
- Discrete grids
- Visibility graphs / roadmaps
- Probabilistic roadmaps
- Voronoi diagrams

A*: Minimization of the Total Estimated Path Costs
- \( g(n) = \text{actual cost} \) from the initial state to \( n \).
- \( h(n) = \text{estimated cost} \) from \( n \) to the closest goal.
- \( f(n) = g(n) + h(n) \), the estimated cost of the cheapest solution through \( n \).
- Let \( h^*(n) \) be the cost of the optimal path from \( n \) to the closest goal.
- \( h \) is admissible if \( h(n) \leq h^*(n) \) for all \( n \):
- A* is optimal when \( h \) is admissible.
**A* on a Grid Graph**

Costs are calculated base on the occupancy probability

---

**Properties of A***

- Optimal Path (on the grid graph)
- Multiple solutions
- Paths can lead the robot close to obstacles
- Does not incorporate the non-determinism of the actions

---

**Markov Decision Process (MDP)**

Given:
- Finite set of states $S$
- Finite set of actions $A$
- Transition probabilities $p(s'|s,a)$
- Reward function $r(s)$

Wanted:
- Policy $\pi(s)$ that maximizes the future expected reward
Value Iteration

- Value function of policy $\pi$
  $$V_{\pi}(s) = E \left[ \sum_{\tau} r(s_\tau) \mid s_0 = s, a_\tau = \pi(s_\tau) \right]$$
- Bellman equation for optimal value function
  $$V(s) = r(s) + \max_a \gamma \sum_s p(s' \mid s, a) V(s')$$
- Value iteration: recursively estimating the value function
  $$V(s) \leftarrow r(s) + \max_a \gamma \sum_s p(s' \mid s, a) V(s')$$
- Greedy policy: $\pi(s) = \text{argmax}_a \gamma \sum_s p(s' \mid s, a) V(s')$

[Bellman 57] [Howard 60] [Sutton/Barto 98]

Properties of MDP-Planning

- Can incorporate the uncertainty in the actions of the robot
- Calculates the cost-optimal paths.
- Requires a model.

Properties of Grid-based Techniques

- Finds the shortest/cost optimal paths
- Only applicable to low-dimensional configuration spaces?
- In practice, they have been mostly applied in two dimensional state spaces only.
Visibility Graph

- Reduce path planning to graph search
- Connect all corners by a direct line as long as they go through free space.
- Add the starting location S and the goal location G.
- Once the Graph has been constructed apply A* search for shortest path
- Leads to optimal paths
- They are, however, only semi-free.

Probabilistic Roadmaps

- For robots with many DOFs (degrees of freedom).
- Sample points in the configuration space and retain those in free-space.
- Alternatively, one can sample on the surface of objects.
- Two points are connected if path between them can be found by local, deterministic planner (e.g., A*).

[Amato and Wu, 96]
Real-World Example

Voronoï Diagrams
- Generalized Voronoï Diagram is the graph containing all points equidistant from the closest two or more obstacle boundaries.
- Starting and goal location are connected to this graph.
- Advantage: Very efficient, especially in high dimensions.
- Disadvantage: Maximizes the distance to obstacles and can lead to suboptimal paths.

Voronoï Example

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)
- Assumption: 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:
- \(V_s\) = all possible speeds of the robot.
- \(V_o\) = obstacle free area.
- \(V_d\) = possible speeds based on possible accelerations.

\[ Space = V_s \cap V_o \cap V_d \]

Dynamic Window Approach (3)
- Which pair \(<v,\omega>\) should we choose from the admissible ones?
- Steering commands are chosen by a heuristic navigation function.
- This function tries to minimize the travel-time base on the heuristic "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]
\[ NF = \alpha \cdot \text{vel} + \beta \cdot nf + \gamma \cdot \Delta nf + \delta \cdot \text{goal} \]
**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.
- Follows grid based path computed by A*.

---

**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.
- Follows grid based path computed by A*.

---
Dynamic Window Approach (5)

- Reacts quickly.
- Low CPU power required.
- Guides a robot on a collision free path.
- Successfully used in a lot of real world installations.
- Resulting trajectories can be sub-optimal.
- Local Minima, sometimes no solution!

Problems of DWAs

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

* The robot drives too fast at \( c_0 \) to enter corridor facing south.
Problems of DWAs

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

Problems of DWAs

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

Problems of DWAs

- Also problems in real world situations:
  - Robot does not slow down early enough to enter the doorway.
Solution (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 \((x,y) \in \text{channel}\).
- 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.

**Timeouts**

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

  ➔ Abort search after .25 secs.

  How to find an admissible steering command?

**Examples**

**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.

Comparison to the DWA (1)

- DWAs often have problems entering narrow passages.
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).
- ...