MP4: Path Planning

Eric Liang
Overview

● Demo due 4/29/2021, Report due 4/30/2021
● 3 Written Questions, 4 Implementation Questions
● Written Questions:
  ○ A* Search
  ○ Probabilistic Road Maps
  ○ Value Iteration
● Implementation Questions
  ○ A* Planner v.s. Hybrid A* Planner
  ○ Design Choices
  ○ Video
  ○ Demo
Module Architecture

- **a_star.py**: You need to implement A*
- **hybrid_a_star.py**: You need to implement Hybrid A*
- **controller.py**: Dubin’s Car Controller
- **mp4.py**: Main Function/Wrapper
- **reset.py**: Resets Car Position (You need to reset to various different positions in this MP!)
A* Planner

- Keep track of both the cost of the partial path to get to a vertex $g(v)$ and the heuristic function estimating the cost to reach the goal from a vertex $h(v)$
- Choose a “ranking” function to be the sum of the two costs: $f(v) = g(v) + h(v)$
- $g(v)$: cost-to-arrive (from the start to $v$)
- $h(v)$: cost-to-go estimate (from $v$ to the goal)
- $f(v)$: estimated cost of the path (from the start to $v$ and then to the goal)
A* Planner

\[
Q \leftarrow \langle \text{start} \rangle \\
\text{while } Q \neq \emptyset: \\
\quad \text{pick (and remove) the path } P \text{ with the lowest estimated cost } (f(P) = g(P) + h(head(P))) \text{ from } Q \\
\quad \text{if } head(P) = x_{goal} \text{ then return } P \\
\quad \text{for each vertex } v \text{ such that } (head(P), v) \in E, \text{ do} \\
\qquad \text{add } \langle v, P \rangle \text{ to } Q \\
\text{Return FAILURE} \\
\]

// initialize queue with start
// Reached the goal
// for all neighbors
// Add expanded paths
// nothing left to consider
Heuristic

- Admissible heuristic: \( h(v) \leq h^*(v) \)
- \( h^*(v) \): The actual cost from node to goal
- \( h(v) \): Needs to be at most \( h^*(v) \) to guarantee optimality
- But it doesn’t mean \( h(v) \) should be as small as possible!
  - When \( h = 0 \), A* is equivalent to UCS
  - \( h(v) \) should be as close to \( h^*(v) \) as possible (Review partial order and domination)
- Consistent heuristic: \( h(u) \leq w(u, v) + h(v) \)
  - Satisfies triangle inequality
  - \( f(v) = g(v) + h(v) \) is non-decreasing along paths
  - No need to compare costs or revisit expanded nodes
Euclidean distance

- Admissible?
- Consistent?
- Can you think about a better heuristic?
- Does it make sense to assign same edge weights/lengths to all neighbors?

https://www.researchgate.net/figure/Neighbors-of-a-pixel-p-a-8-neighbors-and-b-two-causal-neighbors_fig7_224343002
Nodes and Heap/ Priority Queue

● A* node
  ○ Cost-to-arrive g
  ○ Estimated cost of the path: \( f = g + h \) (heuristic to goal)
  ○ Position state: \((x, y)\)
  ○ You can store path here also. Just need the ability to backtrack.

● Does popping smallest element sound familiar?
  ○ Heap (Python heapq)

● How do you construct nodes to be used in heap?
  ○ Easiest: Tuple with the key as the first element (f)
  ○ Cleaner: __lt__ comparator in a class
Dubin’s Car

- $x_{\text{new}} = x + v \cos(\theta)$
- $y_{\text{new}} = y + v \sin(\theta)$
- $\theta_{\text{new}} = \theta + v \tan(\delta) / l$
Hybrid A* Planner

- Problem with A*: Not good with continuous state space in real world
- Discrete path may not be executable by the vehicle dynamics
- Solution: Use representative continuous state on top of discrete nodes
- Hybrid A* Node
  - Cost-to-arrive g
  - Estimated cost of the path: $f = g + h$ (heuristic to goal)
  - Discrete position state: $(x, y, \theta)$
  - Continuous position state: $(x_c, y_c, \theta_c)$
  - Path or other information for backtracking

http://robots.stanford.edu/papers/junior08.pdf
Transition and Discretization

- Say we start from a node with discrete state \((x, y, \theta)\) and continuous state \((x_c, y_c, \theta_c)\)
- We apply a control \(u = (v, \delta)\) to \((x_c, y_c, \theta_c)\) from all the possible actions
- Now we get a continuous state \((x_c', y_c', \theta_c')\)
- We discretize \((x_c', y_c', \theta_c')\) to \((x', y', \theta')\) by rounding (or you can define your own resolution)
- \((x', y', \theta')\) is now the discrete state of your new node
- Cost \(g\) and \(h\) and path are still computed with continuous state
- Discrete state checks whether you’ve explored similar states (or goal)
- \((4.1, 5.9, \ 34.1^\circ)\) is the same as \((3.72, 6.22, 33.8^\circ)\) in terms of if you have visited it before or not
Demo

- The TA will pick a few test cases from the lab manual.
- Students need to show their A* and Hybrid A* planners
  - Finish planning within reasonable amount of time
  - A* planner needs to be optimal
  - Hybrid A* planner needs to have reasonably good performances
Questions?

- Again, start early