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

$\boldsymbol{Q} \leftarrow \langle \boldsymbol{start} \rangle$ *// initialize queue with start*

while $\boldsymbol{Q} \neq \emptyset$:

pick (and remove) the path $\boldsymbol{P}$ with the lowest estimated cost $(\boldsymbol{f}(\boldsymbol{P}) = \boldsymbol{g}(\boldsymbol{P}) + \boldsymbol{h}(\boldsymbol{head}(\boldsymbol{P})))$ from Q

if $\boldsymbol{head}(\boldsymbol{P}) = \boldsymbol{x}_{\boldsymbol{goal}}$ then return $\boldsymbol{P}$ *// Reached the goal*

for each vertex $\boldsymbol{v}$ such that $(\boldsymbol{head}(\boldsymbol{P}), \boldsymbol{v}) \in \boldsymbol{E}$, do *// for all neighbors*

add $\langle v, P \rangle$ to Q *// Add expanded paths*

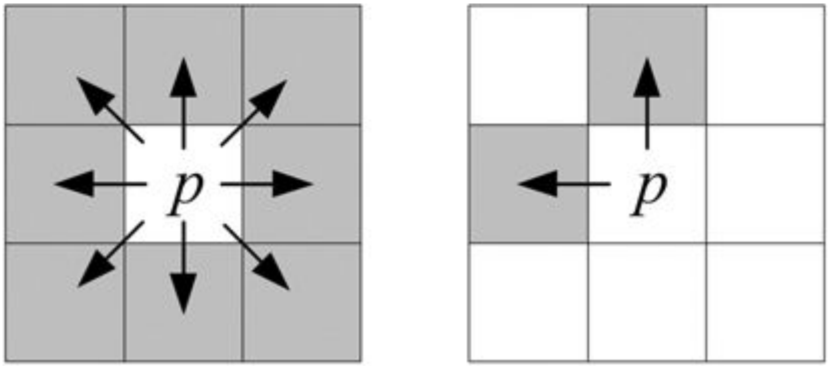
Return FAILURE ***// nothing left to consider***

Heuristic

- Admissible heuristic: h(v) <= 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) <= 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?



(a) (b)

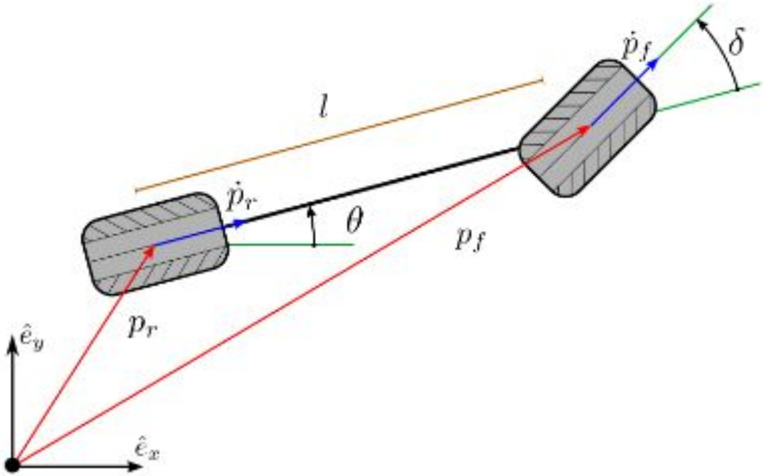
https://www.researchgate.net/figure/Neighbors-of-a-pixel-p-a-8-neighbors-and-b-two-causal-neighbors_fig7_224343002

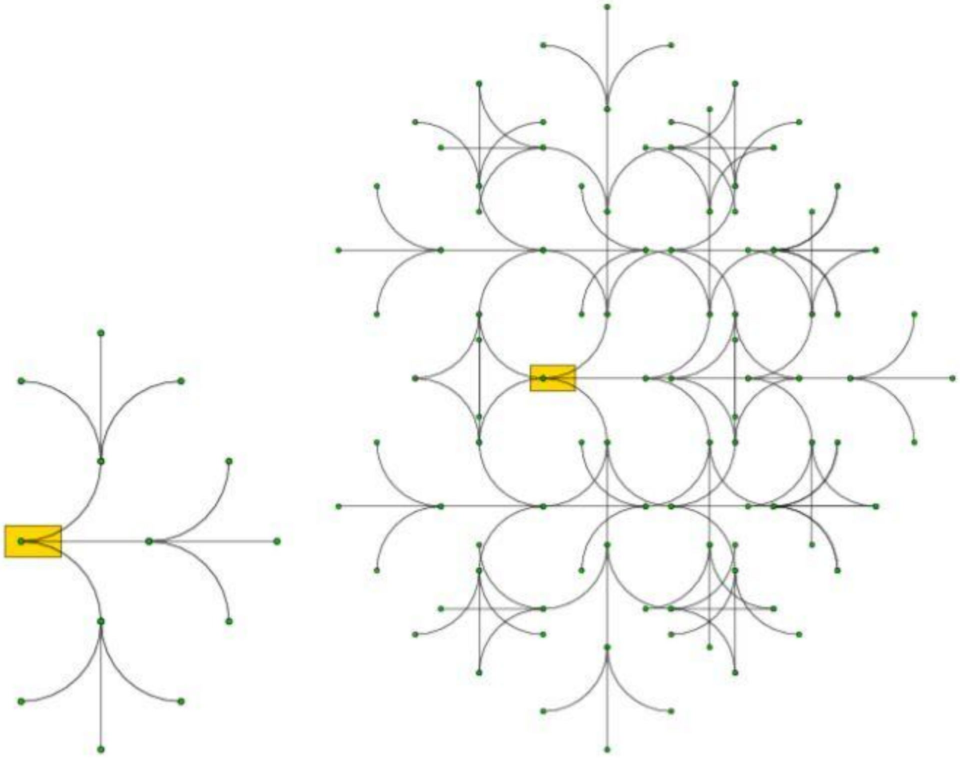
Nodes and Heap/ Priority Queue

- 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

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



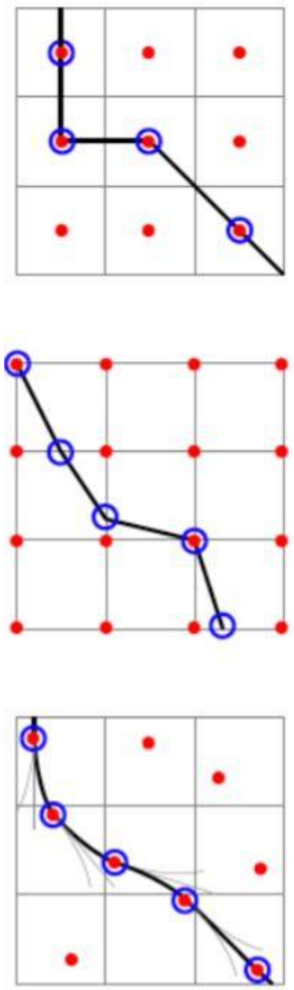


Two stages

Four stages

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, θ)
  - Continuous position state: (xc, yc, θc)
  - 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, θ) and continuous state (xc, yc, θc)
- We apply a control u = (v, δ) to (xc, yc, θc) from all the possible actions
- Now we get a continuous state (xc', yc', θc')
- We discretize (xc', yc', θc') to (x', y', θ') by rounding (or you can define your own resolution)
- (x', y', θ') 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°) is the same as (3.72, 6.22, 33.8°) 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