16-350
Planning Techniques for Robotics

Case Study:
Planning for Autonomous Driving

Maxim Likhachev
Robotics Institute
Carnegie Mellon University
Typical Planning Architecture for Autonomous Vehicle

- **Route Planner**
  - Input: world model
  - Output: next road segment to follow

- **Lane Trajectory Planner**
  - Input: world model, perception data
  - Output: trajectory represented as series of \( <x, y, \theta, v> \) points

- **Path/Motion Planner for Free Spaces**
  - Input: world model, perception data
  - Output: trajectory represented as series of \( <x, y, \theta, v> \) points

- **Trajectory Follower & Low-level Collision Avoidance**
  - Input: world model, perception data
  - Output: Control inputs (e.g., speed and steering angle) for execution
Typical Planning Architecture for Autonomous Vehicle

How do you think the graph is constructed?
Typical Planning Architecture for Autonomous Vehicle

- **Route Planner**
  - next road segment to follow
- **Lane Trajectory Planner**
  - world model
- **Path/Motion Planner for Free Spaces**
  - world model

Planning states defined by: \( x, y, \Theta, v \)

Tartanracing, CMU
Typical Planning Architecture for Autonomous Vehicle

Lane Trajectory Planner

- World model perception data
- Trajectory represented as series of \( <x, y, \theta, v> \) points

Trajectory Follower & Low-level Collision Avoidance

- World model perception data
- Control inputs (e.g., speed and steering angle) for execution
- Trajectory represented as series of \( <x, y, \theta, v> \) points

for Free Spaces

perception data
How do you think the graph is constructed?

Lane Trajectory Planner

Trajectory represented as series of \( x, y, \theta, v \) points

Trajectory Follower & Low-level Collision Avoidance

Control inputs (e.g., speed and steering angle) for execution

world model perception data

perception data

perception data

trajectory represented as series of \( x, y, \theta, v \) points
planning states defined by: discretization along a lane (=x) and perpendicular to it (=y), lane ID, v, time
Typical Planning Architecture for Autonomous Vehicle

We’ll look into the version used for Urban Challenge in ‘07 [Likhachev & Ferguson, ‘09]
Motivation

• Planning **long complex maneuvers** for the Urban Challenge vehicle from CMU (Tartanracing team)

![Urban Challenge vehicle](image)

• Planner suitable for
  – autonomous parking in very large (200m by 200m) cluttered parking lots
  – navigating in off-road conditions
  – navigating cluttered intersections/driveways
Desired Properties

• Generate a path that can be tracked well (at up to 5m/sec):
  – path is a 4-dimensional trajectory:
    \[(x, y, \theta, v)\]
    - orientation
    - speed
Desired Properties

• Generate a path that can be tracked well (at up to 5m/sec):

  - path is a 4-dimensional trajectory:
    \[
    (x, y, \theta, v)
    \]
    \[\text{orientation} \quad \text{speed}\]

  Orientation of the wheels is not included. When will that be a problem?
Desired Properties

• Fast (2D-like) planning in trivial environments:

200 by 200m parking lot
Desired Properties

- But can also handle large non-trivial environments:

200 by 200m parking lot
Desired Properties

- Anytime property: finds the best path it can within X secs and then improves the path while following it.
Desired Properties

- Fast replanning, especially since we need to avoid other vehicles

*planning a path that avoids other vehicles*
Desired Properties

- Fast replanning, especially since we need to avoid other vehicles

*Time is not part of the state-space. When will that be a problem?*
The Approach

• Define an Implicit graph
  – multi-resolution version of a lattice graph

• Search the graph for a least-cost path
  – Anytime D* (ARA* + D* Lite)
Building the Graph

- Lattice-based graph:

  - Each transition is feasible (constructed beforehand)
  - Outcome state is the center of the corresponding cell

(action template)

\[ (x, y, \theta, v) \]
Building the Graph

- Lattice-based graph:
  - Outcome state is the center of the corresponding cell
  - Each transition is feasible (constructed beforehand)
  - Action template

(x, y, \theta, v) \rightarrow \text{replicate it online}
Building the Graph

• Lattice-based graph:

  outcome state is the center of the corresponding cell

  each transition is feasible (constructed beforehand)

  we will be searching this graph for a least-cost path from \( s_{\text{start}} \) to \( s_{\text{goal}} \)

\[ (x, y, \theta, v) \]

replicate it online
Building the Graph

• Multi-resolution lattice:
  – high density in the most constrained areas (e.g., around start/goal)
  – low density in areas with higher freedom for motions

*most constrained areas*
Building the Graph

• The construction of multi-resolution lattice:
  – the action space of a low-resolution lattice is a strict subset of the action space of the high-resolution lattice

*reduces the branching factor for the low-res. lattice*
Building the Graph

- The construction of multi-resolution lattice:
  - the action space of a low-resolution lattice is a strict subset of the action space of the high-resolution lattice
    - reduces the branching factor for the low-res. lattice
  - the state-space of a low-resolution lattice is discretized to be a subset of the possible discretized values of the state variables in the high-resolution lattice
    - reduces the size of the state-space for the low-res. lattice
  - both allow for seamless transitions
Building the Graph

- Multi-resolution lattice used for Urban Challenge:

  dense-resolution lattice
  
  36 actions, 32 discrete values of heading 0.25m discretization for x,y

  low-resolution lattice
  
  24 actions, 16 discrete values of heading 0.25m discretization for x,y
Building the Graph

• Properties of multi-resolution lattice:
  – utilization of low-resolution lattice: every path that uses only the action space of the low-resolution lattice is guaranteed to be a valid path in the multi-resolution lattice

  – validity of paths: every path in the multi-resolution lattice is guaranteed to be a valid path in a lattice that uses only the action space of the high-resolution lattice
Building the Graph

• Benefit of the multi-resolution lattice:

<table>
<thead>
<tr>
<th>Lattice</th>
<th>States Expanded</th>
<th>Planning Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-resolution</td>
<td>2,933</td>
<td>0.19</td>
</tr>
<tr>
<td>Multi-resolution</td>
<td>1,228</td>
<td>0.06</td>
</tr>
</tbody>
</table>
Searching the Graph

• Anytime D*:
  – anytime incremental version of A*

  – **anytime**: computes the best path it can within provided time and improves it while the robot starts execution.

  – **incremental**: it reuses its previous planning efforts and as a result, re-computes a solution much faster
Searching the Graph

• Anytime D*: computes a path reusing all of the previous search efforts
  
  - set $\varepsilon$ to large value;
  - until goal is reached
    ComputePathwithReuse();
    publish $\varepsilon$-suboptimal path for execution;
    update the map based on new sensory information;
    update current state of the agent;
  - if significant changes were observed
    increase $\varepsilon$ or replan from scratch;
  - else
    decrease $\varepsilon$;

  makes it improve the solution

guarantees that 
$\text{cost}(\text{path}) \leq \varepsilon \text{ cost(optimal path)}$

desired bound on the
Searching the Graph

- Anytime behavior of Anytime D*:

![Diagram showing solution cost and time](image)

- Solution cost and time graph:
  - Cost: 133,736
  - Time: 0.3 s
  - ε: 3.0
  - # expands: 1,715

- Solution cost and time graph:
  - Cost: 77,345
  - Time: 0.6 s
  - ε: 1.0
  - # expands: 14,132
Searching the Graph

- Incremental behavior of Anytime D*: 

  initial path

  a path after re-planning
Searching the Graph

• Performance of Anytime D* depends strongly on heuristics $h(s)$: estimates of cost-to-goal

should be consistent and admissible (never overestimate cost-to-goal)
Searching the Graph

- Performance of Anytime D* depends strongly on heuristics $h(s)$: estimates of cost-to-goal

$S = (x, y, \theta, v)$

$h(s)$

$S_{goal}$

Any ideas?

should be consistent and admissible (never overestimate cost-to-goal)
Searching the Graph

- In our planner: $h(s) = \max(h_{\text{mech}}(s), h_{\text{env}}(s))$, where
  - $h_{\text{mech}}(s)$ – mechanism-constrained heuristic
  - $h_{\text{env}}(s)$ – environment-constrained heuristic

$h_{\text{mech}}(s)$ – considers only dynamics constraints and ignores environment

$h_{\text{env}}(s)$ – considers only environment constraints and ignores dynamics
Searching the Graph

- In our planner: $h(s) = \max(h_{mech}(s), h_{env}(s))$, where
  - $h_{mech}(s)$ – mechanism-constrained heuristic
  - $h_{env}(s)$ – environment-constrained heuristic

$h_{mech}(s)$ – considers only dynamics constraints and ignores environment

$h_{env}(s)$ – considers only environment constraints and ignores dynamics

pre-computed as a table lookup for high-res. lattice

computed online by running a 2D A* with late termination
Searching the Graph

- In our planner: $h(s) = \max(h_{mech}(s), h_{env}(s))$, where
  - $h_{mech}(s)$ – mechanism-constrained heuristic
  - $h_{env}(s)$ – environment-constrained heuristic

$h_{mech}(s)$ – considers only dynamics constraints and ignores environment

$h_{env}(s)$ – considers only environment constraints and ignores dynamics

pre-computed as a table lookup for high-res. lattice

computed online by running a 2D A* with late termination
Searching the Graph

• In our planner: \( h(s) = \max(h_{mech}(s), h_{env}(s)) \)

• \( h_{mech}(s) \) – admissible and consistent

• \( h_{env}(s) \) – admissible and consistent

• \( h(s) \) – admissible and consistent

**Theorem.** The cost of a path returned by Anytime D* is no more than \( \varepsilon \) times the cost of a least-cost path from the vehicle configuration to the goal configuration using actions in the multi-resolution lattice, where \( \varepsilon \) is the current value by which Anytime D* inflates heuristics.
Searching the Graph

• Benefit of the combined heuristics:

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>States Expanded</th>
<th>Planning Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment-constrained only</td>
<td>26,108</td>
<td>1.30</td>
</tr>
<tr>
<td>Mechanism-constrained only</td>
<td>124,794</td>
<td>3.49</td>
</tr>
<tr>
<td>Combined</td>
<td>2,019</td>
<td>0.06</td>
</tr>
</tbody>
</table>
Optimizations

- Pre-compute as much as possible
  - convolution cells for each action for each initial heading
Results

• Plan improvement

*Tartanracing, CMU*
Results

- Replanning in a large parking lot (200 by 200m)

*Tartanracing, CMU*
What You Should Know…

• Typical hierarch of planners used in self-driving

• Multi-resolution lattice

• Benefits of anytime and incremental planning

• Ways to generate informative heuristics for motion planning for self-driving