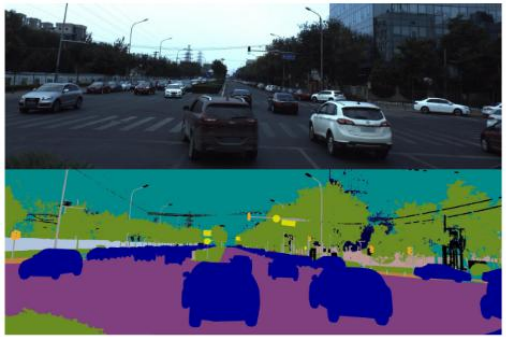
FreiCAR
Block 3: Planning
Lecture 1: Dijkstra, A*, PRM, RRT

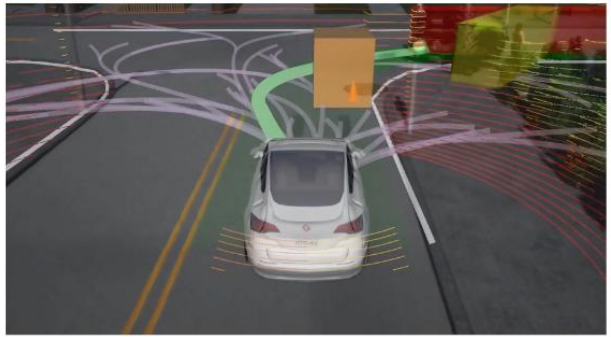
Eugenio Chisari

AIS Autonomous Intelligent Systems

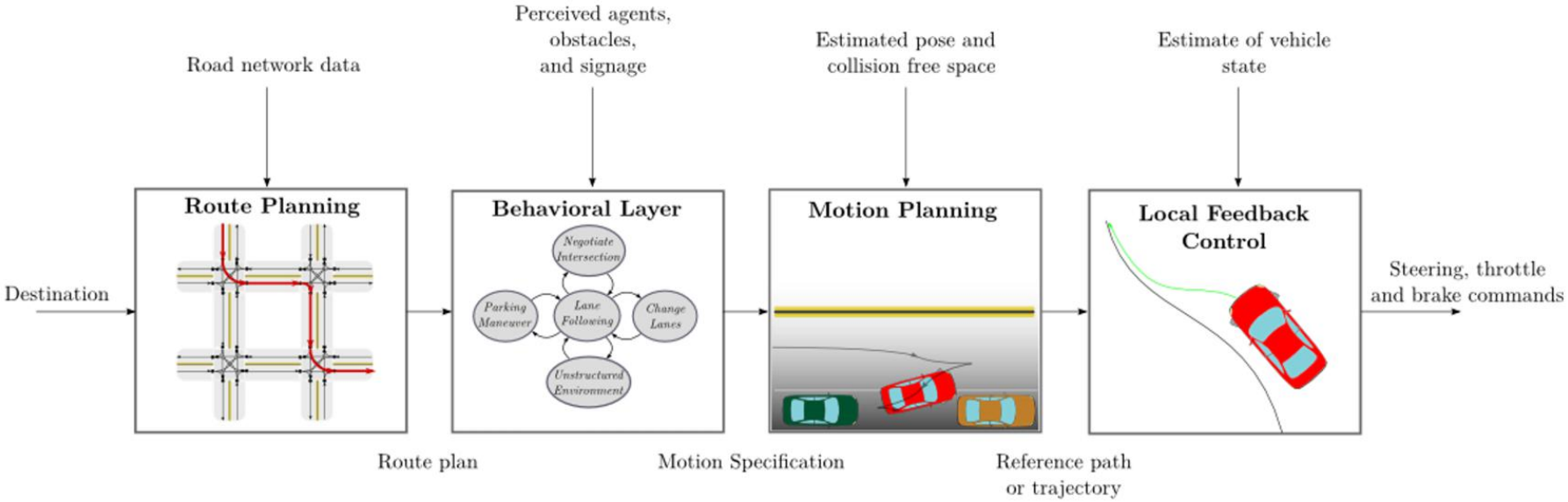
Planning Problem: Context

- Perception: collecting information
- Localization: organizing information
- Planning: decision making





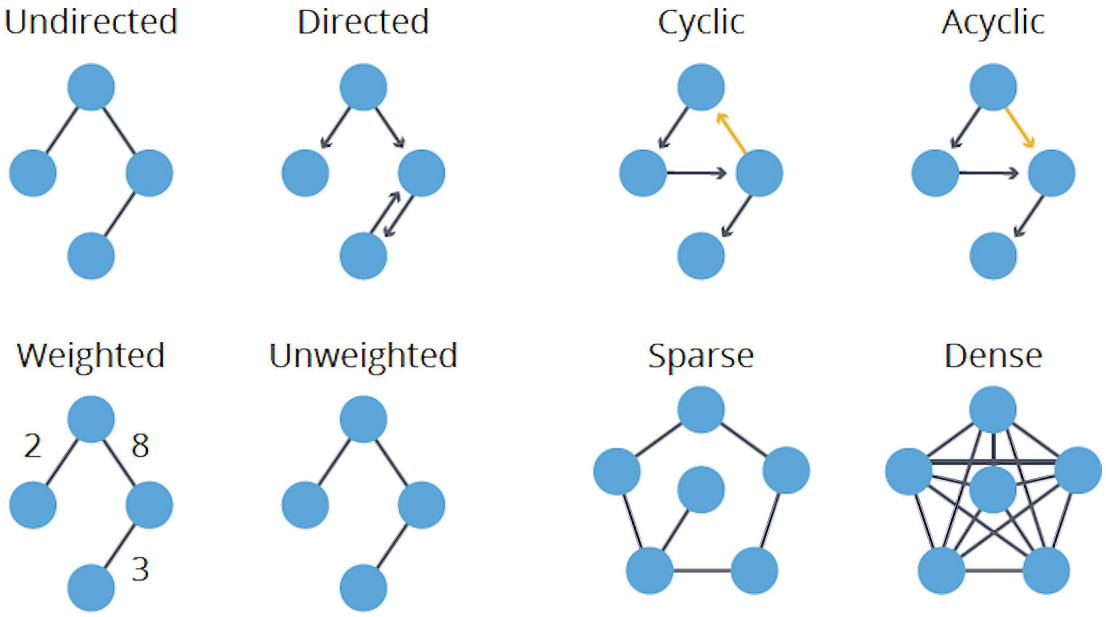
Hierarchical Planning



Route Planning

- Find the fastest route between point A and B
- High level of abstraction, doesn’t consider lower level issues like kinemathic feasibility or obstacle avoidance.
- The map can be represented as a graph, where each road is an edge and each intersection is a node
- The solution can be found via shortest path algorithms, like Dijkstra and A*

Graph Recap



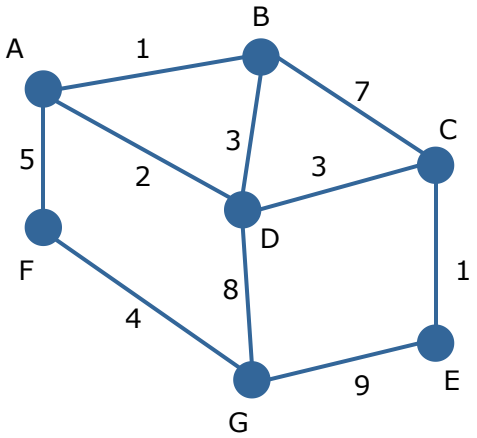
A graph is a set of nodes (vertices) + a set of edges: $G = (V + E)$.
It can have different properties.

Pathfinding Algorithms

Given a graph, different shortest path algorithms exists:

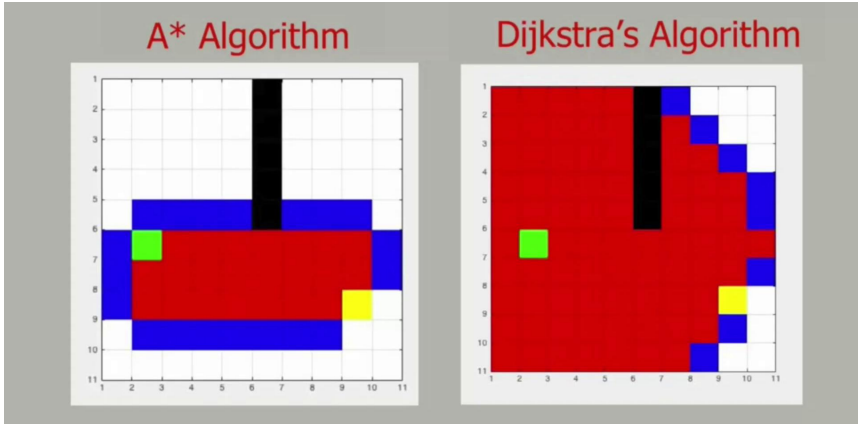
- Unweighted Graph → Breadth First Search, $O(V + E)$
- Directed Acyclic Graph → Topological Order Traversal, $O(V + E)$
- Non-negative weighted Graph → Dijkstra Algorithm, $O(E \bullet \log V)$
- General Graph → Bellman-Ford Algorithm, $O(E \bullet V)$

Dijkstra Algorithm



- Keep a Distance vector which stores for each node the distance of the shortest path found so far.
- Keep a Priority queue of (index, distance) pairs of the nodes to be visited next, ordered by lowest distance.
- At each iteration, visit the next node in the priority queue and update it's distance vector if the new path shorter.
- When goal node reached, stop search.
- For implementation details, many tutorials available online, e.g. https://www.youtube.com/watch?v=pSqmAO-m7Lk&t=494s.

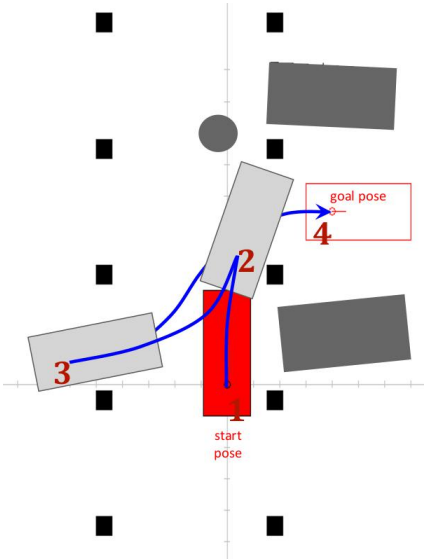
Extensions: A* and D*



- A* (A star):
  - Extends Dijkstra by incorporating a heuristic to guide the search towards the goal. The heuristics used is a lower bound of the remaining cost to reach the goal node from any given vertex in the graph.

- D* (D star):
  - Extension of A* that allows efficient replanning when a small portion of the graph changes.

Motion Planning



Motion Planning Objectives:

- Shortest Path (or minimal time)
- Smoothess

Motion Planning Constraints:

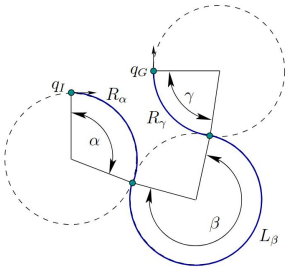
- Collision free motion: avoid all static and moving obstacles
- Vehicle kinematics and dynamics constraints. In the case of a car, non-holonomic! (i.e. cannot move sideways or rotate on the spot, also called “Differential Constraints”)

Challenge:

- In free space, no straightforward graph representation! We need to build one.

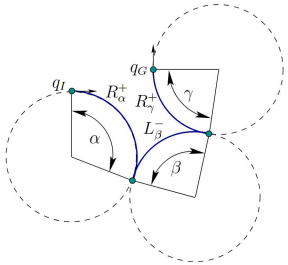
Kinematic Feasibility

Simplified car models commonly used as steering functions, i.e. to check for kinematic feasibility (they have closed form solutions):



Dubin Car:

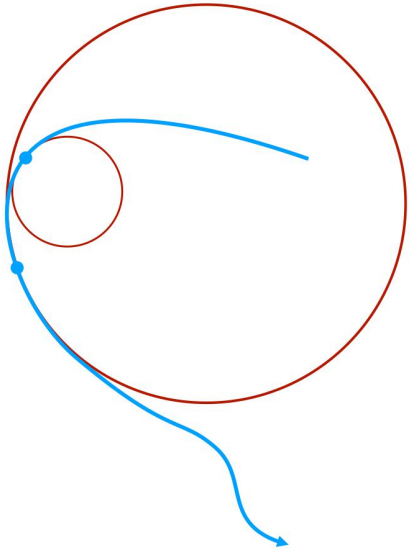
- moves only forward
- steering curvature is upper bounded



Reeds-Shepp Car:

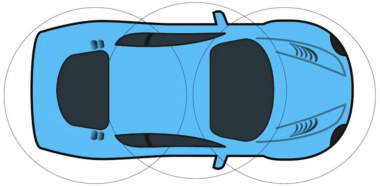
- can move forward and backward
- steering curvature is upper bounded

Reminder: curvature of a path



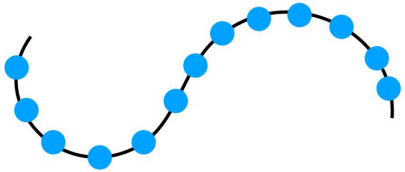
- Curvature of a path at a given point is the inverse of the radius of a tangent circle: $k = 1/R$
- For straight paths, the curvature is 0 (tangent to a circle with infinite radius)
- The curvature profile $k(s)$ uniquely describe a planar path (see Frenet-Serret representation of curves)

Collision Checking



Collision checking between two objects:

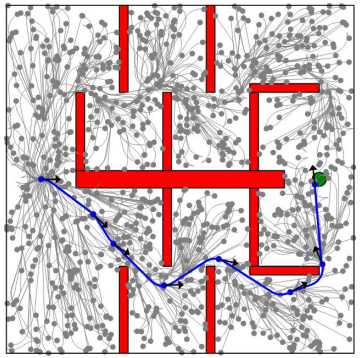
- Approximate both with circles
- Check that distance between centers larger than radius of circles



Collision checking along a path:

- Sample points (i.e. poses) uniformly along the path (densely enough)
- Check collision for each pose

Sampling Based Planning



Schmerling, E., Janson, L., & Pavone, M. (2015, May).
Optimal sampling-based motion planning under differential constraints: the driftless case.
In 2015 IEEE International Conference on Robotics and Automation (ICRA)

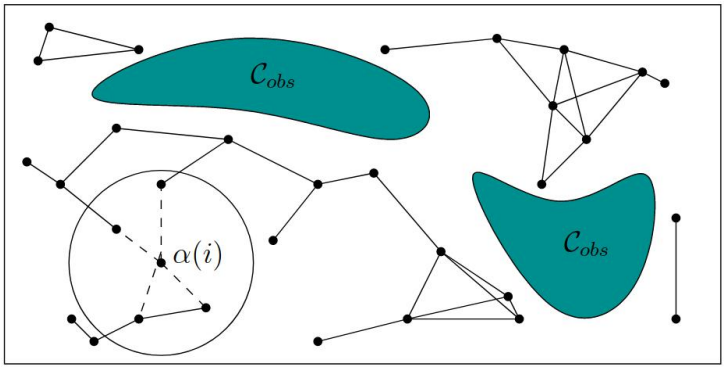
General Recipe

- Seed the graph
- Sample a new node
- Choose to which other node it might connect to
- Decide which edges to add
- Decide which edges to remove

Sampling-based methods advantages:

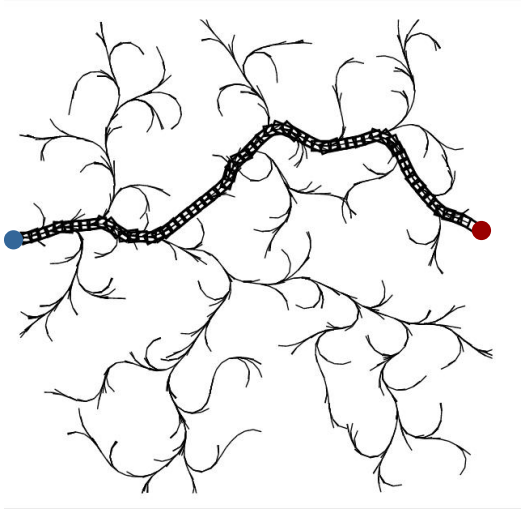
- Easy and flexible to implement: different collision checking and steering functions can be used for different scenarios and vehicle kinematics
- Robust: can use conservative collision checking and steering functions
- Scalable: scale well with dimensions

PRM - Probabilistic Road Map



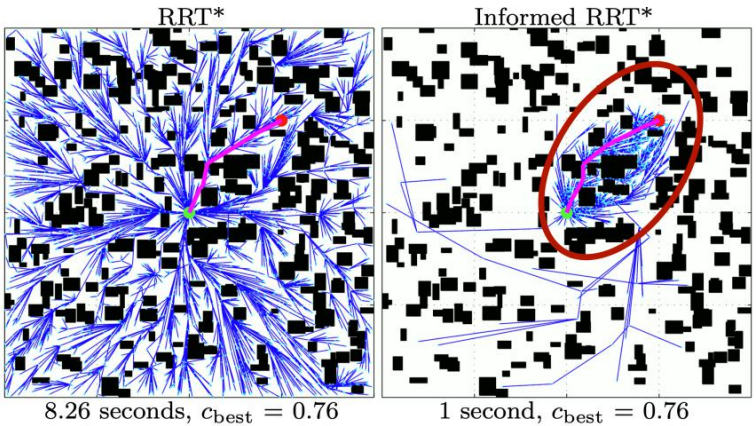
- Idea: construct a dense enough road map first, use it for search second. Efficient if we need multiple queries on the same graph.
- New pose samples are connected to neighbors within radius R or to k-nearest neighbors.
- New connections valid only if path feasible (check both collisions and kinematics).
- After graph is constructed, can apply A*. You might want to smooth the found path using a local optimization method.
- Optimal extensions exist (i.e. PRM*).

RRT - Rapidly exploring Random Tree



- Idea: build a tree with root at the start pose. Much more efficient than PRM if we only need a single query
- Iterations:
  - Sample a point in free space
  - Find closest vertex of the tree to that point
  - Grow the tree from that vertex in the direction of the sampled point.
  - Once in a while, use the goal position instead of a random sample

RRT Extensions

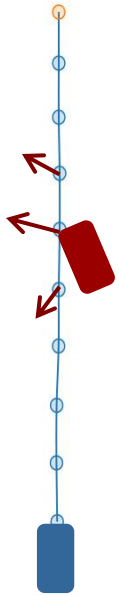


https://www.youtube.com/watch?v=nsl-5MZfwu4

Many extensions and improvements exist. A few examples:

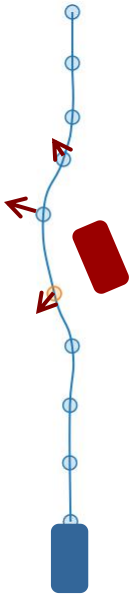
- RRT*: optimal version of RRT. To achieve optimality, after adding a point the graph needs to be rewired.
- Informed RRT*: heuristic based, biases the search toward goal

Local Planner - Gradient methods



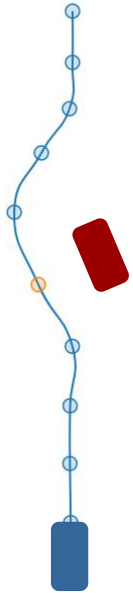
- Local Planner: given a nominal unfeasible path, find a feasible alternative
- Gradient based methods, also called variational methods
- Example: line following
- Path usually parametrized, e.g. a spline
- Do gradient updates to minimize an objective function
- Also useful to smooth a path found via RRT or PRM

Local Planner - Gradient methods



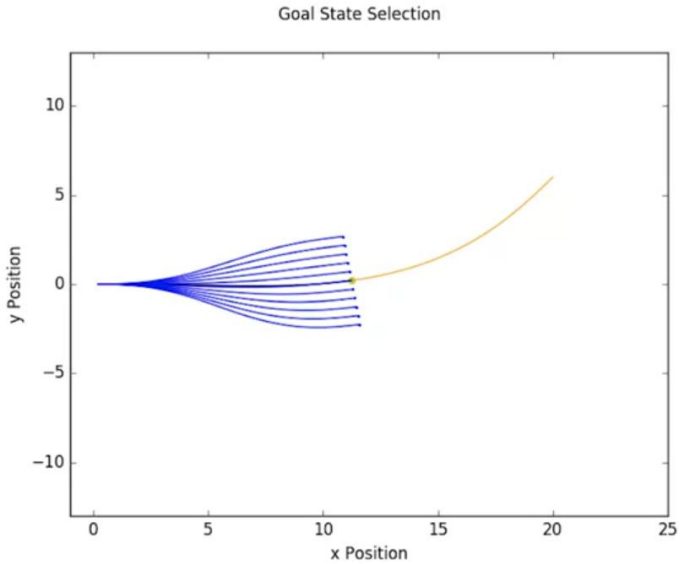
- Local Planner: given a nominal unfeasible path, find a feasible alternative
- Gradient based methods, also called variational methods
- Example: line following
- Path usually parametrized, e.g. a spline
- Do gradient updates to minimize an objective function
- Also useful to smooth a path found via RRT or PRM

Local Planner - Gradient methods



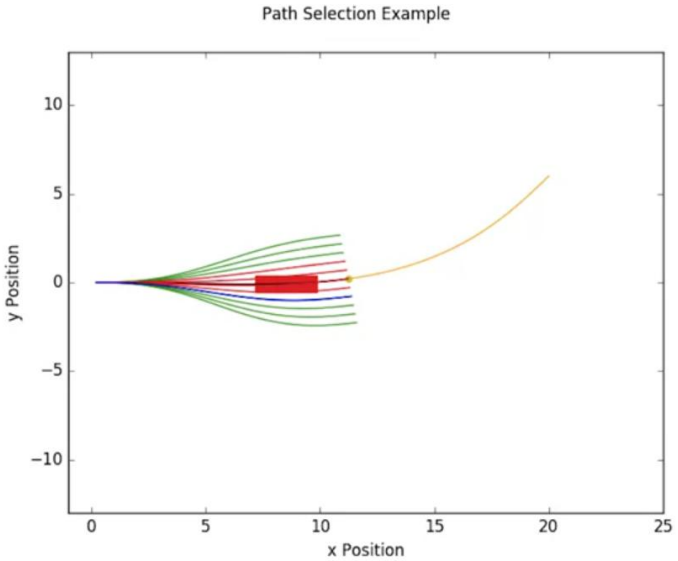
- Local Planner: given a nominal unfeasible path, find a feasible alternative
- Gradient based methods, also called variational methods
- Example: line following
- Path usually parametrized, e.g. a spline
- Do gradient updates to minimize an objective function
- Also useful to smooth a path found via RRT or PRM

Local Planner - Lattice methods



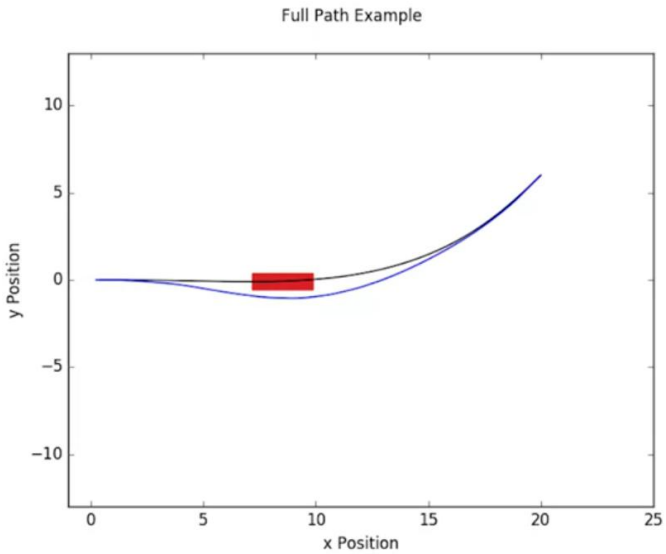
- Lattice Methods: evaluate a set of trajectories and pick the best one
- Example: Conformal Lattice Planning (see figure)
- At given horizon length, pick a set of points perpendicular to the nominal path
- Connect them to the start position (usually via a parametrized curve, e.g. a spline)
- Evaluate all of them with the objective function and pick the best one

Local Planner - Lattice methods



- Lattice Methods: evaluate a set of trajectories and pick the best one
- Example: Conformal Lattice Planning (see figure)
- At given horizon length, pick a set of points perpendicular to the nominal path
- Connect them to the start position (usually via a parametrized curve, e.g. a spline)
- Evaluate all of them with the objective function and pick the best one

Local Planner - Lattice methods



- Lattice Methods: evaluate a set of trajectories and pick the best one
- Example: Conformal Lattice Planning (see figure)
- At given horizon length, pick a set of points perpendicular to the nominal path
- Connect them to the start position (usually via a parametrized curve, e.g. a spline)
- Evaluate all of them with the objective function and pick the best one

Conclusions

- Planning is a crucial building block of autonomous vehicles.

- Planning in general is a very difficult problem! Things we didn’t consider:
  - Uncertainty in the map and in the state estimation
  - Modeling errors in the vehicle dynamics
  - Interaction with other agents (game theory)
  - Anytime planning: how to handle tight computation time constraints

- There is no general method to solve the complete problem!

References

- LaValle, S. M. (2006)
  "Planning algorithms".
  Cambridge university press.

- Paden B. et al. (2016).
  "A survey of motion planning and control techniques for self-driving urban vehicles."
  IEEE Transactions on intelligent vehicles

- Introduction to mobile robotics, SS20 lecture
  http://ais.informatik.uni-freiburg.de/teaching/ss20/robotics/slides/19-pathplanning-long.pdf

- The Duckietown Foundation
  https://www.duckietown.org/