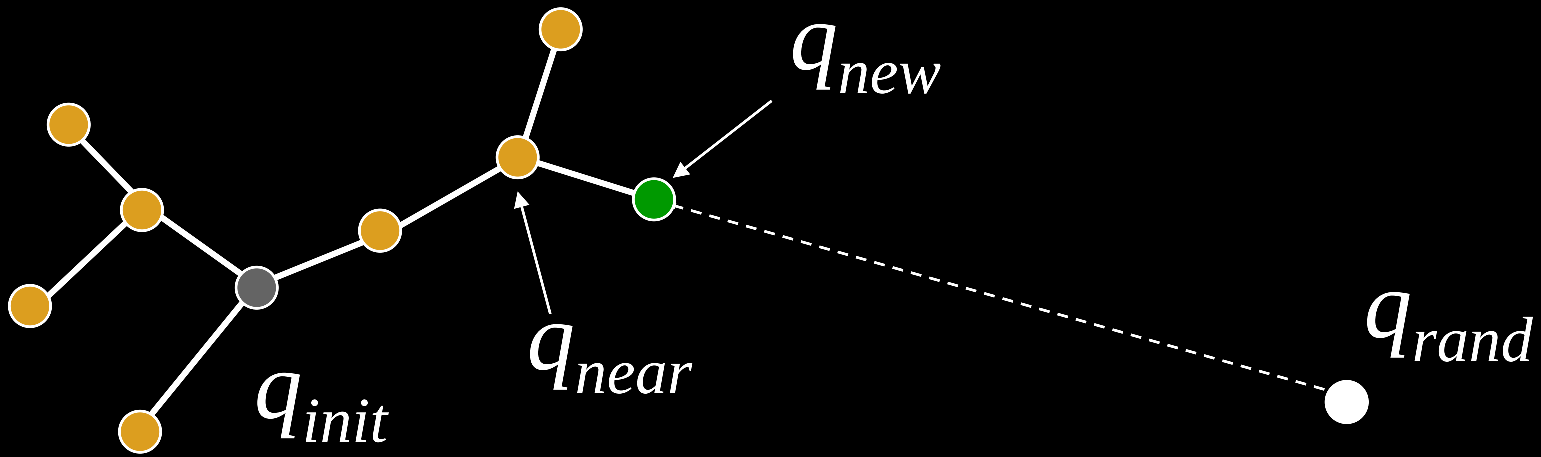
非完整规划

Motion Planning IV – Non-Holonomic Motion Planning

Some material from Howie Choset, J. Kuffner, M. Pivtoraiko, Matt Mason

Last time…

- We learned about RRTs….



- But the standard version of sampling-based planners assume the robot can move in any direction at any time
- What about robots that can't do this?

Outline

- Non-Holonomic definition and examples
- Discrete Non-Holonomic Planning
- Sampling-based Non-Holonomic Planning

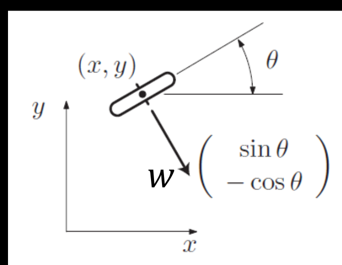
Holonomic vs. Non-Holonomic Constraints

- Holonomic constraints depend only on configuration
  - $F(q, t) = 0$ (note that they can be time-varying!)
  - Technically, these have to be bilateral constraints (no inequalities)
    - In robotics literature we ignore this so we can consider collision constraints as holonomic

- Non-holonomic constraints are constraints that **cannot** be written in this form

Holonomic vs. Non-Holonomic Constraints

- Example: The kinematics of a unicycle
  - Can move forward and back
  - Can rotate about the wheel center
  - Can't move sideways



$$\dot{q} = (\dot{x}, \dot{y}, \dot{\theta})^T$$
$$w = (\sin\theta, -\cos\theta, 0)$$
$$w\dot{q} = 0 \longleftarrow \text{Constraint}$$

- But wait, why can't we just integrate them to get a holonomic constraint?
- Can still reach any (x,y,θ) (so no constraint on configuration)
  - But may not be able to move in a certain direction *instantaneously*

Holonomic vs. Non-Holonomic Constraints

- Non-holonomic constraints are **non-integrable**, i.e. can't re-write them as holonomic constraints
  - Thus non-holonomic constraints **must** contain derivatives of configuration
  - They are sometimes called non-integrable **differential** constraints

- Thus, we need to consider how to move between configurations (or states) when planning
  - Previously we assumed we can move between arbitrary nearby configurations using a straight line

Constraint Taxonomy

bilateral	two-sided constraint, which can be expressed by equations of the form $F(\ldots) = 0$.
unilateral	a one-sided constraint, requiring an inequality $F(\ldots) \geq 0$.
holonomic	a constraint that can be expressed as an equation in just the configuration variables, and possibly time, but independent of the rate variables, $F(\mathbf{q}, t) = 0$.
nonholonomic	a constraint that <u>cannot</u> be expressed in the form $F(\mathbf{q}, t) = 0$, requiring either inequalities or rate variables.
scleronomic	a stationary constraint, expressible independent of time $F(\mathbf{q}, \dot{\mathbf{q}}) = 0$.
rheonomic	a moving constraint, involving time $F(\mathbf{q}, \dot{\mathbf{q}}, t) = 0$.

Mechanics of Robotic Manipulation, **Matthew T. Mason, MIT Press, August 2001.**

- Note: uses the standard definition of holonomic (no inequalities allowed)

State Space vs. Control Space

- State Space

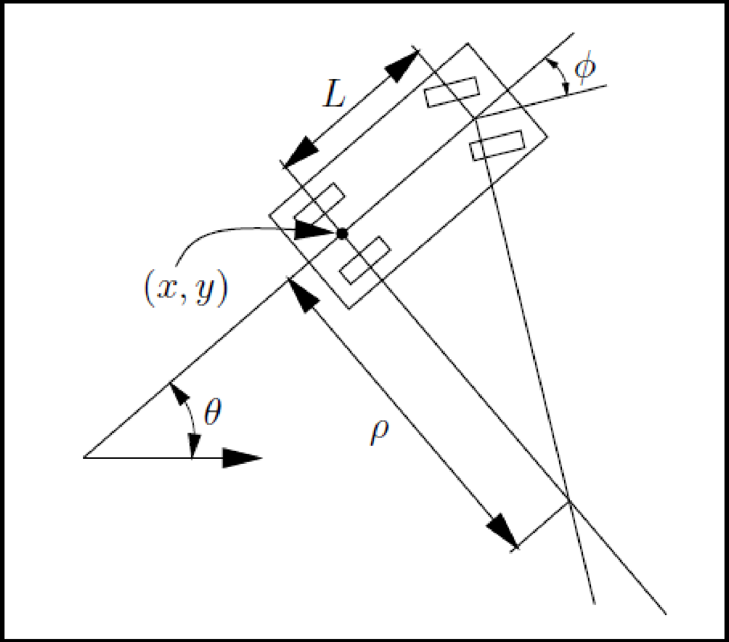
$$x, y, z, \psi, \varphi, \theta$$
$$\dot{x}, \dot{y}, \dot{z}, \dot{\psi}, \dot{\varphi}, \dot{\theta}$$

- Control space
  - Speed or Acceleration
  - Steering

Example: Simple Car

- Non-holonomic Constraint:

$$-\dot{x}\sin\theta + \dot{y}\cos\theta = 0.$$



- Motion model:

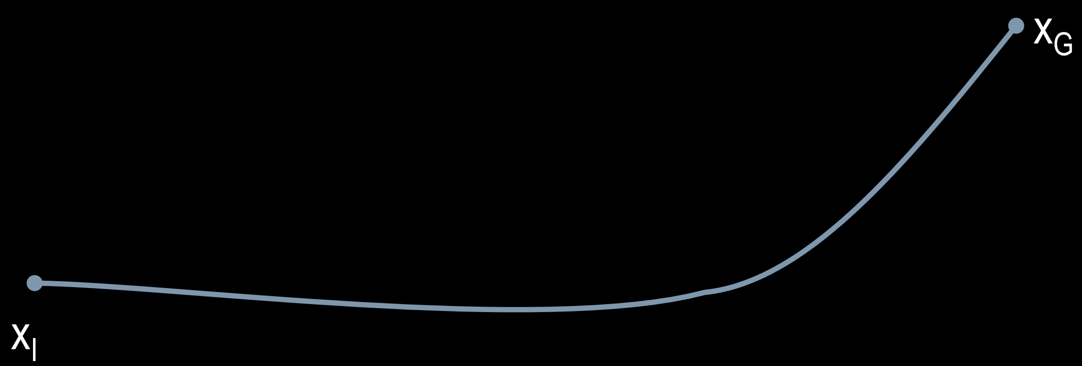
$$\begin{aligned}\dot{x} &= u_s \cos\theta \\ \dot{y} &= u_s \sin\theta \\ \dot{\theta} &= \frac{u_s}{L}\tan u_\phi.\end{aligned}$$

u_s = speed

u_ϕ = steering angle

Moving between states (with no obstacles)

- Two-Point Boundary Value Problem (BVP): Find a control sequence to take system from state x_I to state x_G while obeying kinematic constraints.



- LOTS of methods for this
  - Shooting method: Pick initial guess, iteratively get closer to goal
  - Many “Steering” methods can be used (see LaValle Chapter 15.5)
- For motion planning, we only use this locally
  - Because it doesn’t account for obstacles

Methods for Planning for Non-holonomic Systems

- Discrete search
  - Sequencing primitives
  - State lattice
- Sampling-based
  - PRM-style
  - RRT-style

Discrete Planning for Non-Holonomic Systems

- Two Well-known Alternatives:
  1. Search for sequence of primitives to get to a goal state
  2. Compute *State Lattice*, search for sequence of states in lattice
     - By construction of state lattice, can always get between these states

Discrete Planning Option 1: Sequencing Primitives

- Discretize control space
  - Barraquand & Latombe, 1993:
    - Motion primitives: 3 arcs (+ reverse) at κ_{max}
    - Cost = number of reversals
    - Search using Dijkstra's Algorithm
    - Disadvantage: Discontinuous curvature

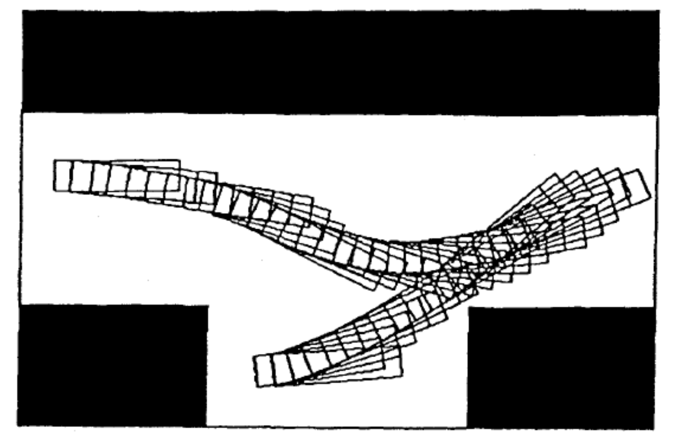


Fig. 4. Parking a car.

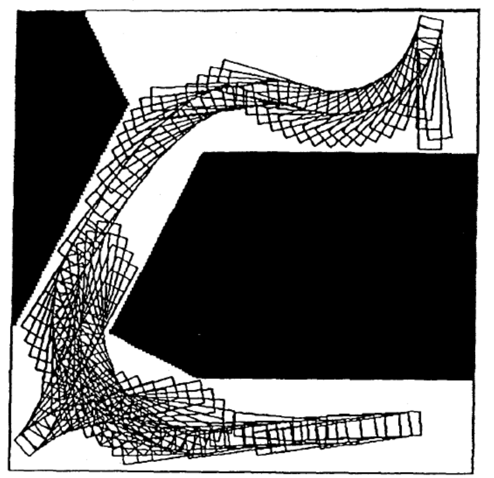
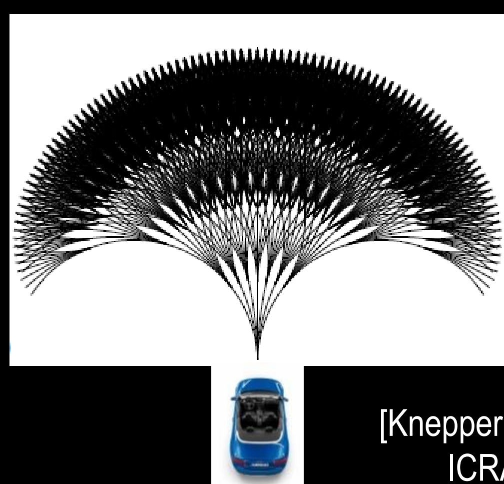


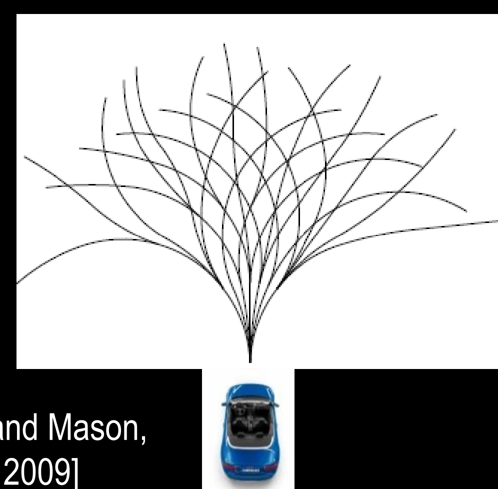
Fig. 5. Car maneuvering in a cluttered workspace.

Discrete Planning Option 1: Sequencing Primitives

- Choice of set of primitives affects
  - Completeness
  - Optimality
  - Speed
- Some algorithms build good (small) sets of primitives

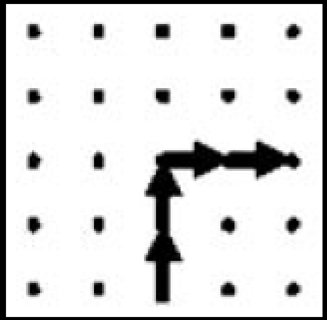


[Knepper and Mason, ICRA 2009]

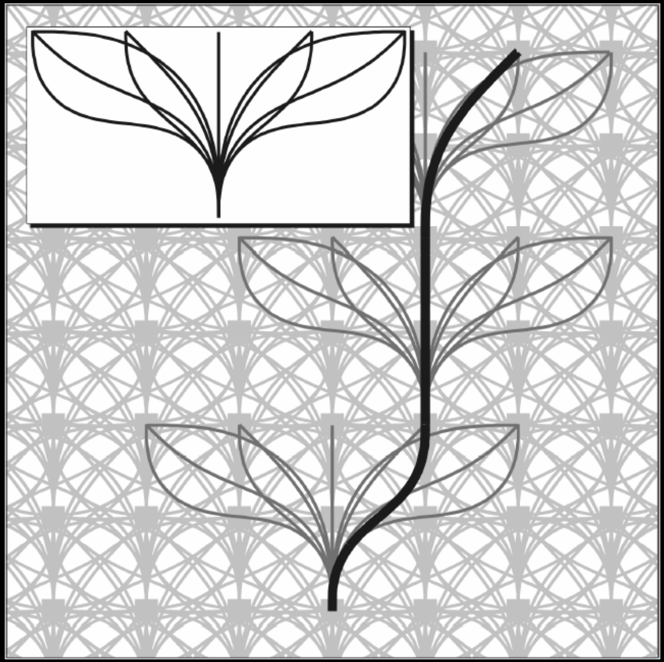


Discrete Planning Option 2: State Lattice

- Pre-compute state lattice
- Two methods to get lattice:
  - Forward: For certain systems, can sequence primitives to make lattice
  - Inverse: Discretize space, use BVP solvers to find trajectories between states
- Impose continuity constraints at graph vertices
- Search state lattice like any graph (i.e. A*)



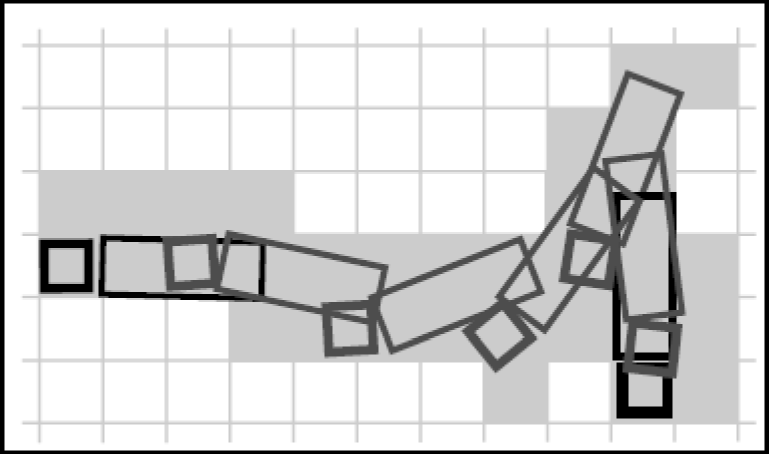
Traditional lattice yields discontinuous motion



Pivtoraiko et al. 2009

Discrete Planning Option 2: State Lattice

- Pre-compute swept volume of robot for each transition for faster collision check



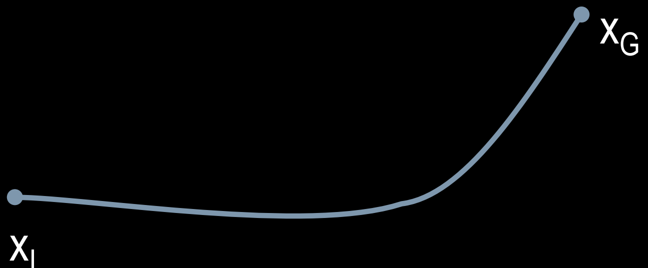
Pivtoraiko et al. 2009

Sampling-Based Planning for Non-Holonomic Systems

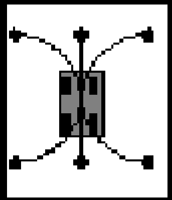
- Forming a full state lattice is impractical for high dimensions, so sample instead.
- **IMPORTANT**: We are now sampling **state space** (position and velocity), not C-space (position only)
- Why is this hard?
  - Dimension of the space is doubled
  - Moving between points is harder (can't go in a straight line)
  - Distance metric is unclear
    - We usually use Euclidian, even though it's not the right metric

PRM-style Non-Holonomic Planning

- Sampling, graph building, and query strategies are all the same as regular PRM
- Problem: Local planner needs to reach an EXACT state (i.e. a given node) while obeying non-holonomic constraints
- In general: BVP problem, use general solver (slow)



- In practice local planner specialized to system type
- Example: For Reeds-Shepp car, can compute optimal path between nodes quickly



Break

RRT-style Non-Holonomic Planning

- RRT became famous partly because of its success as a method for non-holonomic planning
- Sampling and tree building is the same as regular RRT
- Problem: Not all straight lines are valid, can't extend toward nodes
  - One solution: use motion primitives to get as close to target node as possible

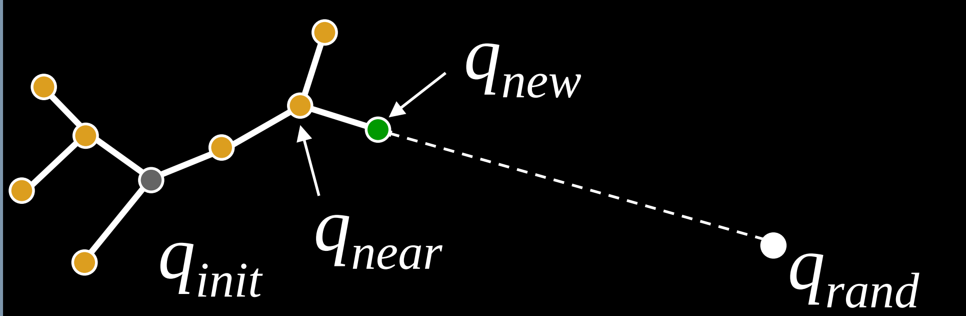
RRTs for Non-Holonomic Systems

- Apply motion primitives (i.e. simple actions) at q_{near}

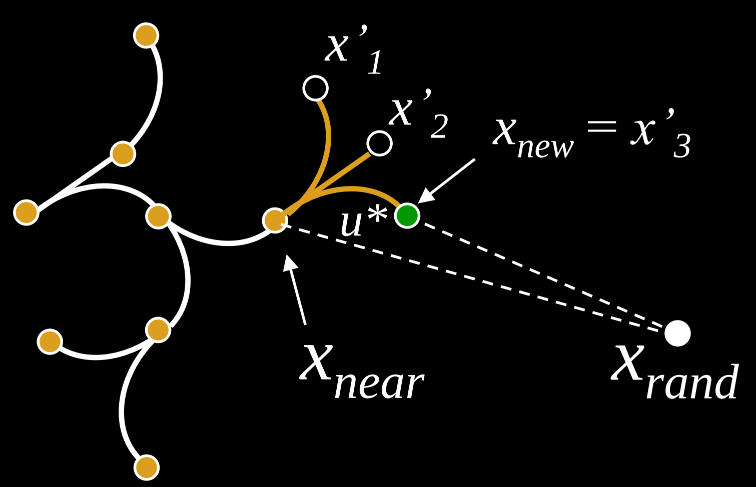
$$x' = f(x,u) \text{ --- use action } u \text{ from } x \text{ to arrive at } x'$$

$$\text{chose } u_* = \arg\min_{u_i} d(f(x,u_i), x_{rand})$$

Holonomic RRT



Non-Holonomic RRT[1]

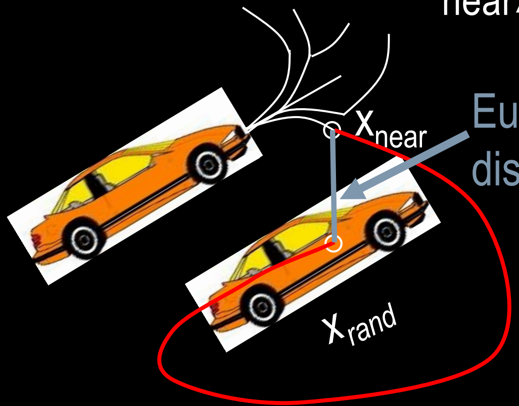


- You probably won't reach x_{rand} by doing this
  - Key point: No problem, you're still exploring!

[1]often called *Kinodynamic RRT*

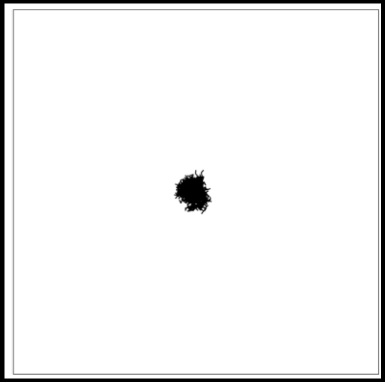
RRTs and Distance Metrics

- Hard to define *d*, the distance metric (needed to determine x_{near})
  - Mixing velocity, position, rotation, etc.



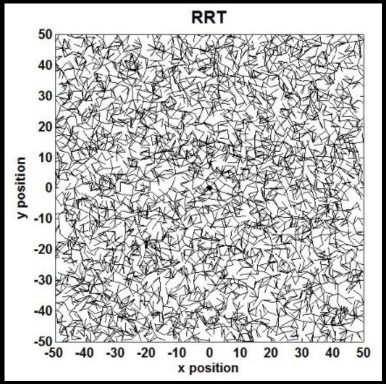
Configurations are close according to Euclidian metric, but actual distance is large

- How do you pick a good x_{near}?



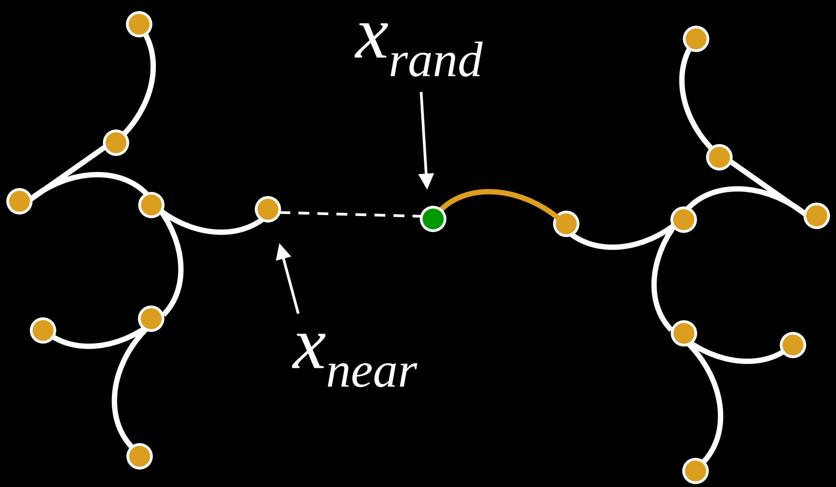
Random Node Choice
(bad distance metric)

Euclidean is somewhere in the middle



RRT Node Choice
(good distance metric)

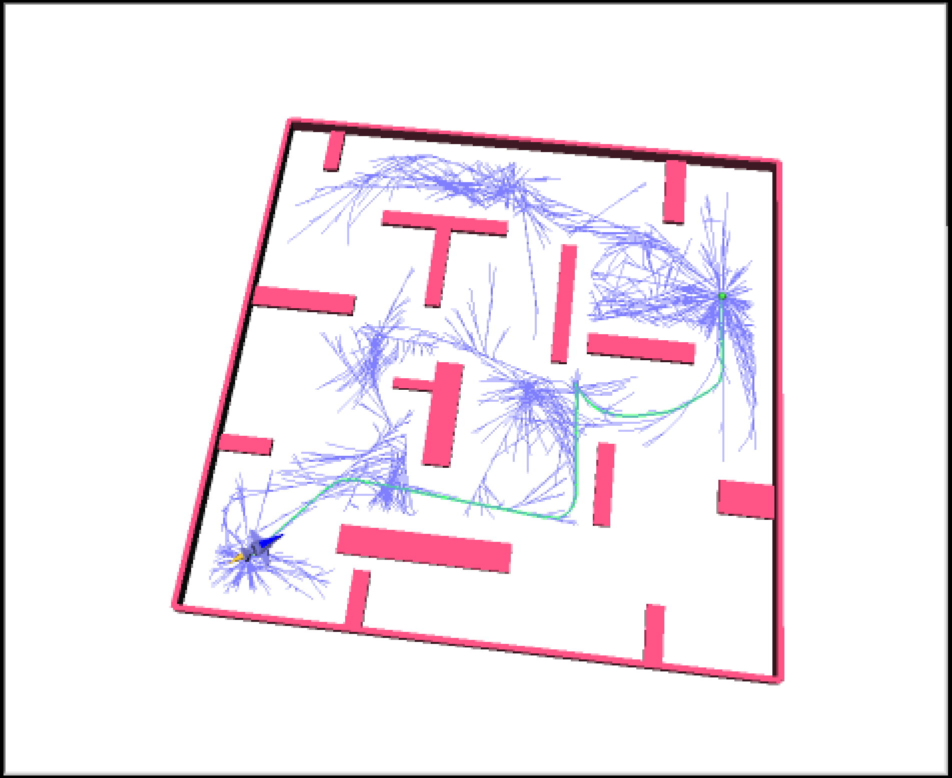
BiDirectional Non-Holonomic RRT



- How do we bridge these two points?

Non-holonomic Smoothing

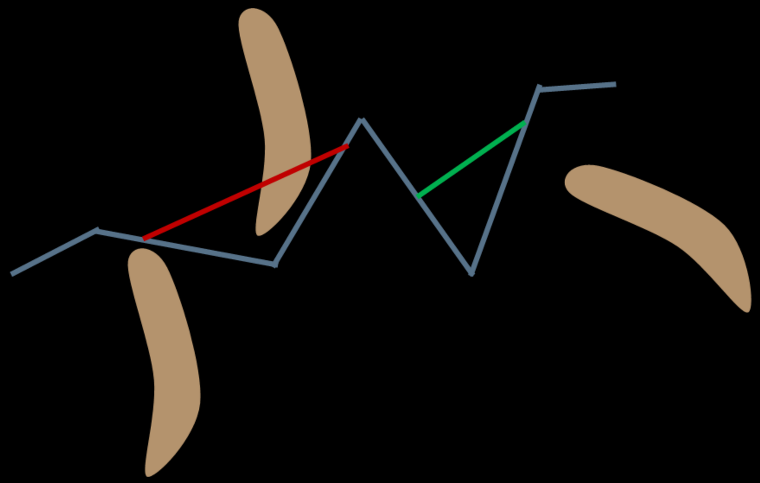
- Similar to holonomic case, paths produced can be highly suboptimal



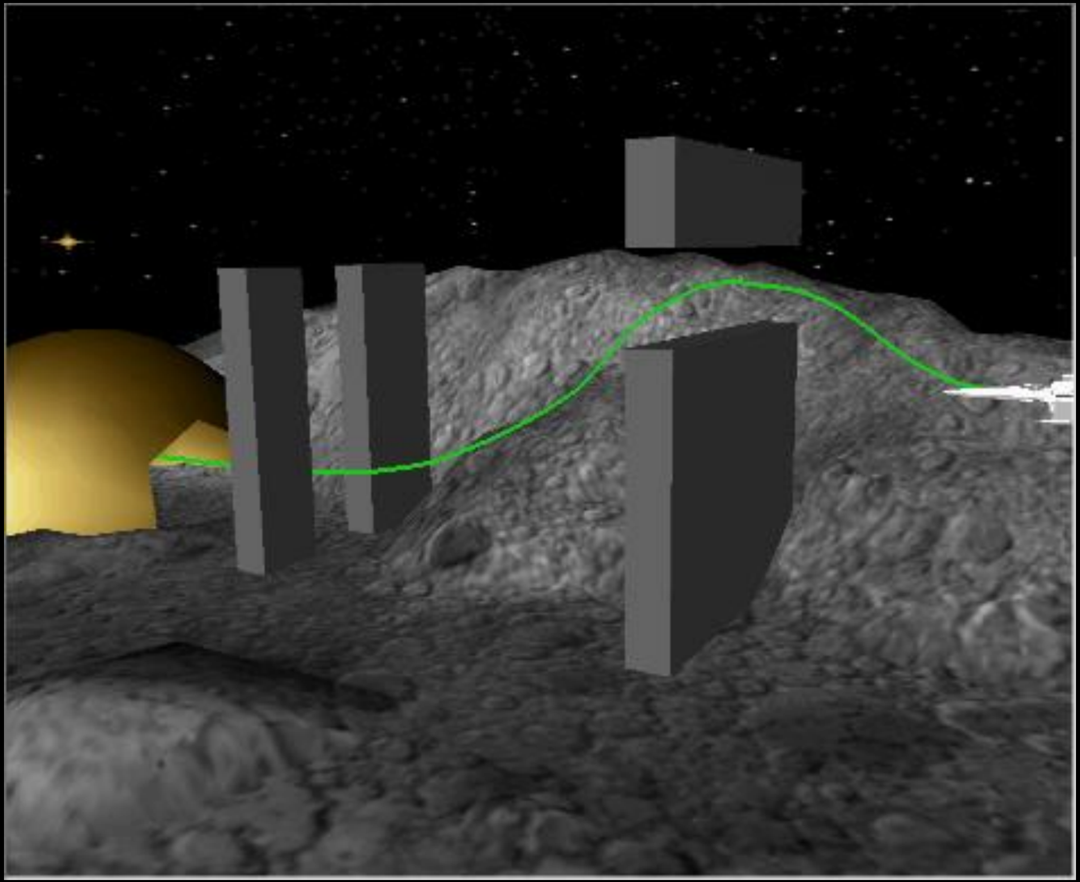
Hovercraft with 2 Thrusters in 2D

Non-Holonomic Smoothing

- Smoothing methods:
  - General trajectory optimization
  - Convert path to cubic B-spline
    - Be careful about collisions
- Can we use shortcut smoothing?



RRTs can Handle High DOF



12DOF Non-Holonomic Motion Planning

Summary

- Non-holonomic constraints are constraints that must involve derivatives of position variables

- Discrete Non-Holonomic Planning
  - Option 1: Search for sequence of primitives to get to a goal state
  - Option 2: Compute *State Lattice*, search for sequence of states in lattice

- Sampling-based Non-Holonomic Planning
  - Adapt PRM to use BVP solver
  - Adapt RRT to use motion primitives (+ BVP solver for BiDirectional case)

Homework

- Read IK Survey
- HW 3 is posted