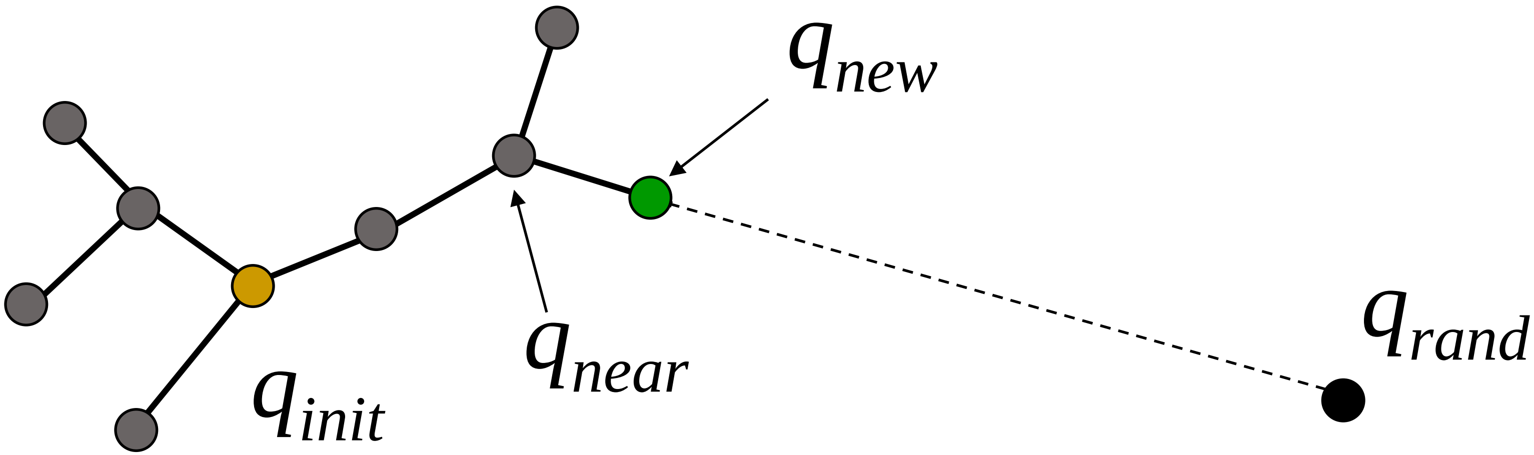
# Non-holonomic Planning

Jane Li
Assistant Professor
Mechanical Engineering & Robotics Engineering
http://users.wpi.edu/~zli11

# Recap

- We have learned about RRTs….




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

# Outline

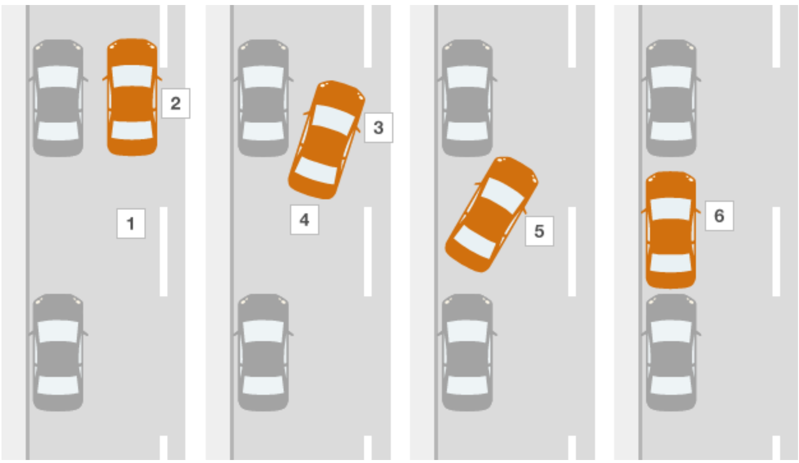
- Non-Holonomic definition and examples
- Discrete Non-Holonomic Planning
- Sampling-based Non-Holonomic Planning

# Holonomic vs. Non-Holonomic Constraints

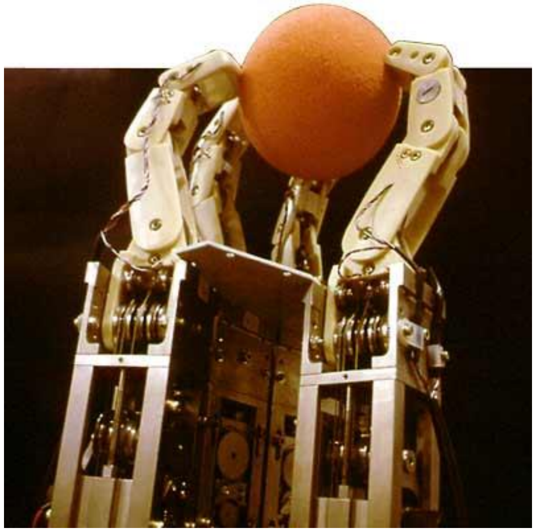
- **Holonomic** constraints depend only on **configuration**
  - $F(q, t) = 0$ (note they can be **time-varying**!)
  - Technically, these have to be **bilateral constraints** (no inequalities)
    - In robotics literature we ignore this so we can consider collision constraints as holonomic

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

# Example of Non-holonomic Constraint




Parallel Parking



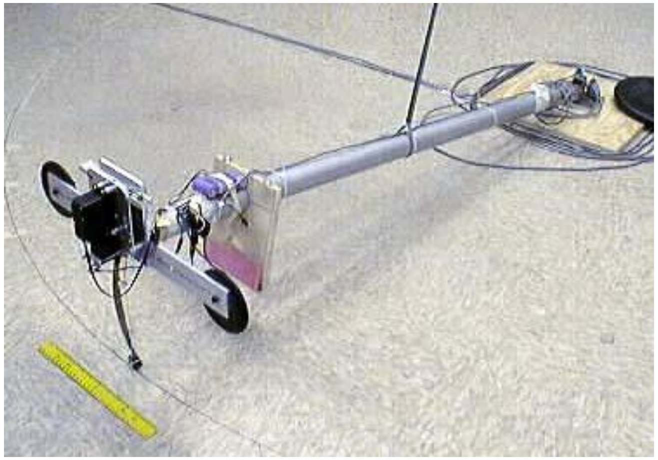
Manipulation with a robotic hand

Multi-fingered hand from Nagoya University

**Rolling without contact**

# Example of Non-holonomic Constraint

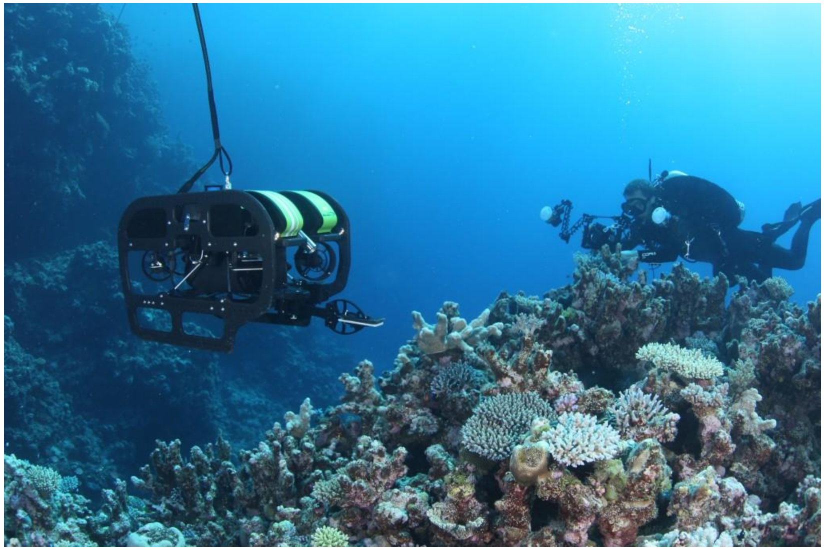
Hopping robots – RI's bow leg hopper (CMU)



AERcam, NASA - Untethered space robots

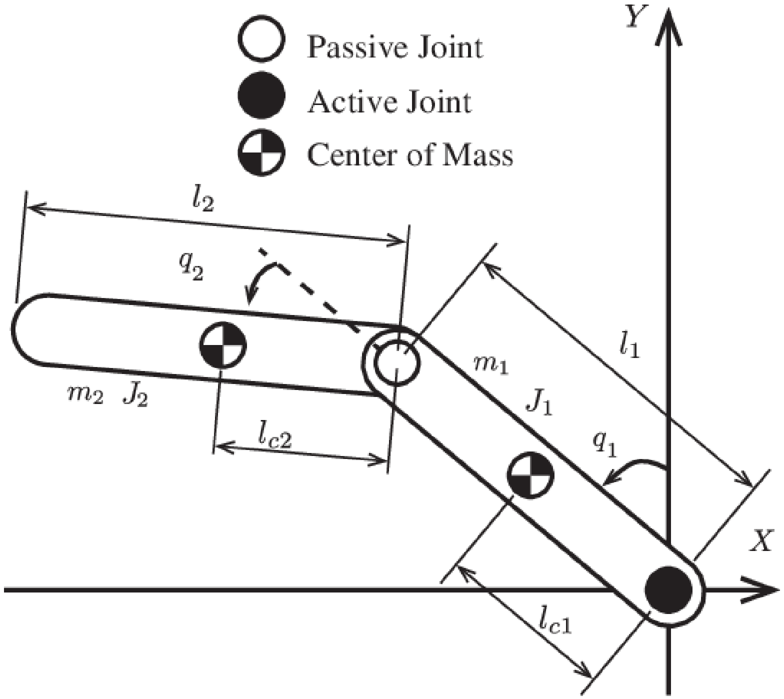
**Conservation of angular momentum**

# Example of Non-holonomic Constraint



Underwater robot
Forward propulsion is allowed
only in the pointing direction

Robotic Manipulator
with passive joints



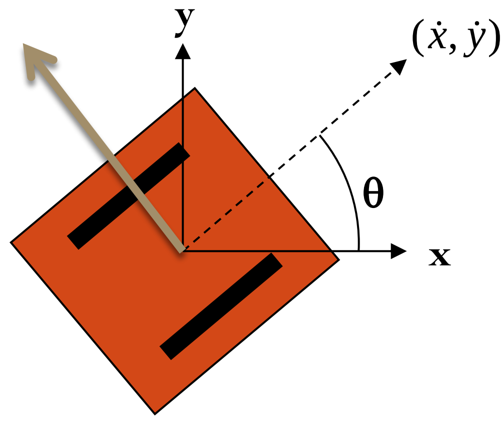

**A Chosen actuation strategy**

# How to Represent the Constraint Mathematically?

- Constraint equation

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

- What does this equation tell us?
  - The direction we can't move in
    - If θ=0, then the velocity in y = 0
    - If θ=90, then the velocity in x = 0
  - Write the constraint in matrix form




$$q = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \quad \dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$$

Position & Velocity Vectors

$$w_1(q) = [-\sin\theta \;\; \cos\theta \;\; 0\,]$$

Constraint Vector

$$w_1(q)\cdot\dot{q} = 0 = [-\sin\theta \;\; \cos\theta \;\; 0]\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \Rightarrow -\dot{x}\sin\theta + \dot{y}\cos\theta = 0$$

# 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

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

- Can we just integrate them to get a holonomic constraint?
  - Intermediate values of its trajectory matters
- Can we still reach any configuration (x,y,θ)?
  - No constraint on configuration, but …
  - May not be able to go to a (x,y,θ) **directly**

# 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. But now …

# 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 angle

# Example – Simple Car

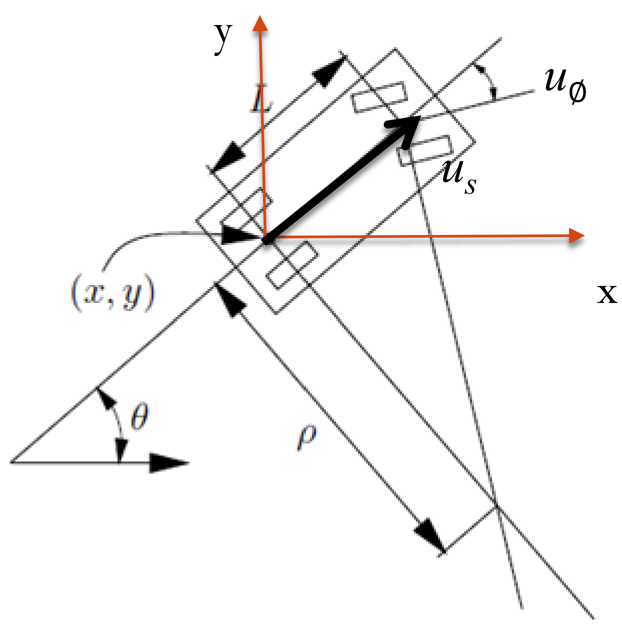
- Non-holonomic Constraint:

Dimension of configuration space?

  - In a small time interval, the car must move approximately in the **direction that the rear wheels are pointing**.

$$\Delta T \rightarrow 0, \quad \frac{dx}{dy} = \frac{\dot{x}}{\dot{y}} = \tan\theta \quad \Rightarrow \quad \dot{y}\cos\theta - \dot{x}\sin\theta = 0$$

- Motion model
  - $u_s$ = speed
  - $u_\phi$ = steering angle



# Example – Simple Car

Dimension of configuration space?

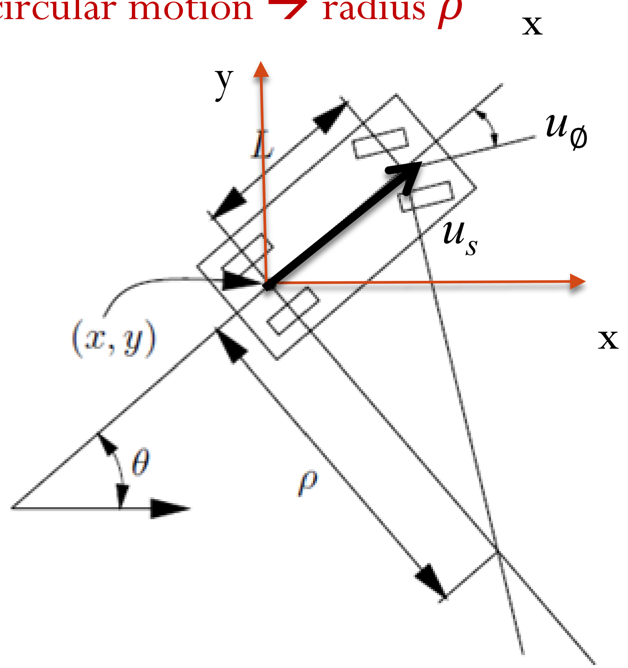
- Motion model
  - $u_s$ = speed

$$\dot{x} = u_s \cos\theta, \quad \dot{y} = u_s \sin\theta$$

  - $u_\phi$ = steering angle
    - If the steering angle is fixed, the car travels in a circular motion → radius $\rho$
    - Let $\omega$ denote the distance traveled by the car

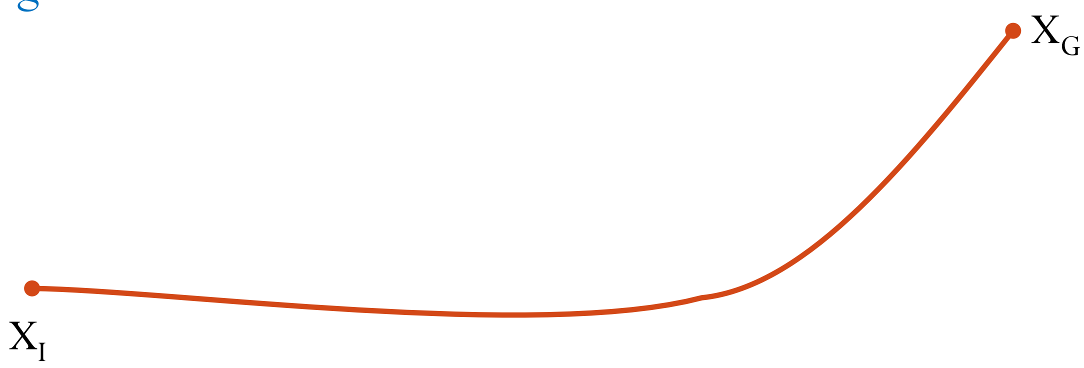
$$d\omega = \rho d\theta, \quad \frac{L}{\rho} = \tan u_\phi \;\Rightarrow\; d\theta = \frac{\tan u_\phi}{L} d\omega, \quad \dot{\omega} = u_s$$

$$\dot{\theta} = \frac{u_s}{L} \tan u_\phi$$



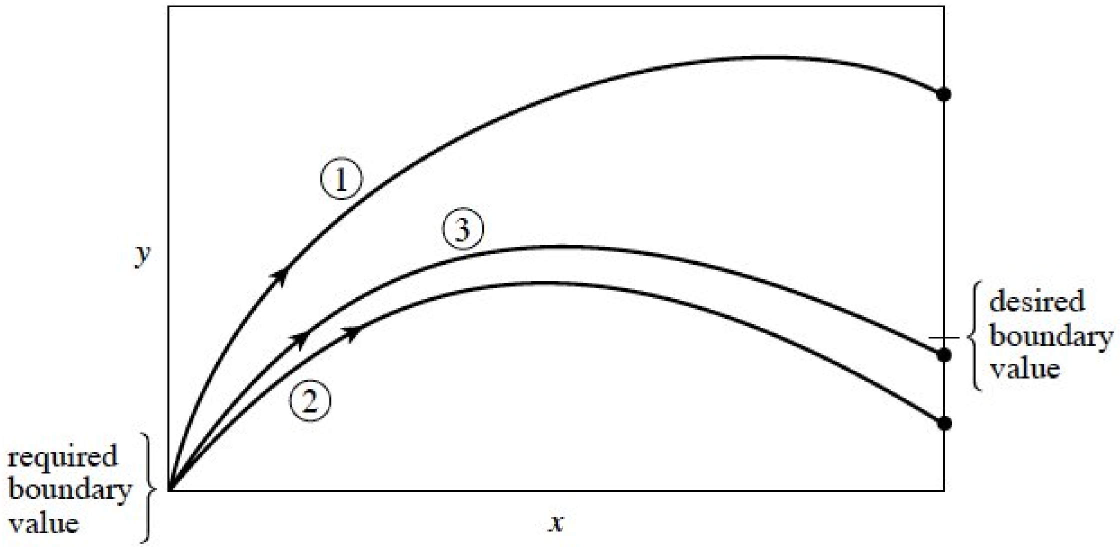
# Moving Between States (with No Obstacles)

- 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.



# Shooting Method

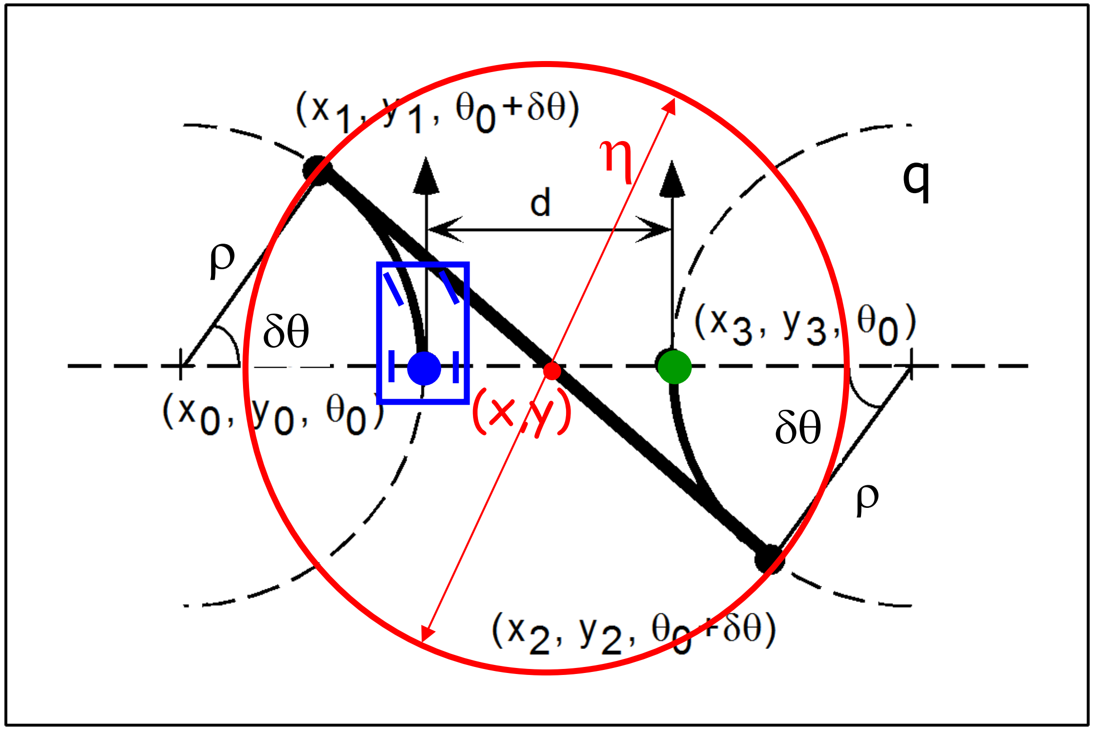
- Basically, we 'shoot' out trajectories in different directions until we find a trajectory that has the desired boundary value.
  - System $\frac{d\mathbf{y}}{dx} + \mathbf{f}(x, \mathbf{y}) = 0$
  - Boundary condition $y(0) = 0,\ y(1) = 1$



# Alternative Method

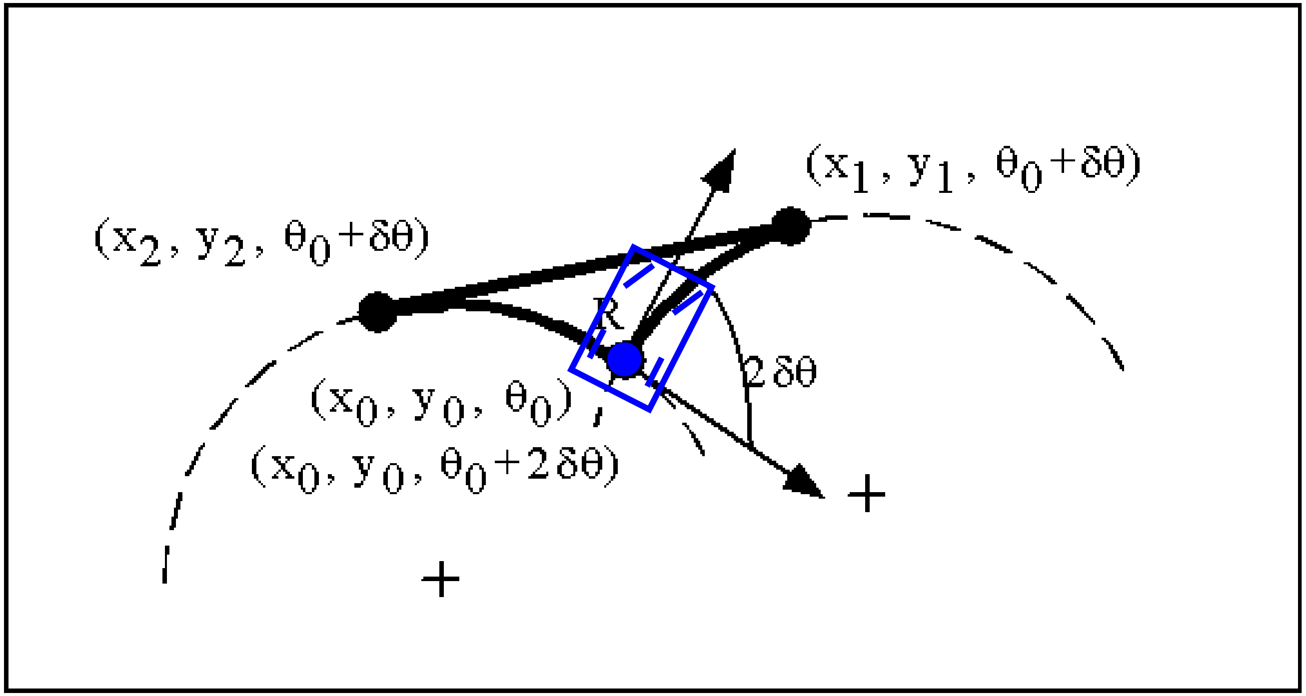
- Due to non-holonomic constraint
  - Direct (sideway) motion is prohibited, but can be approximated by a series of forward/backward and turning maneuvers
- Therefore, what we can do …
  - Plan a path ignoring the car constraints
  - Apply sequence of allowed maneuvers

# Type 1 Maneuver



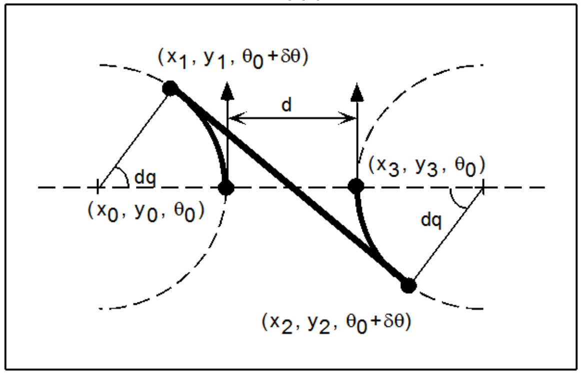

→ **Allows sidewise motion**

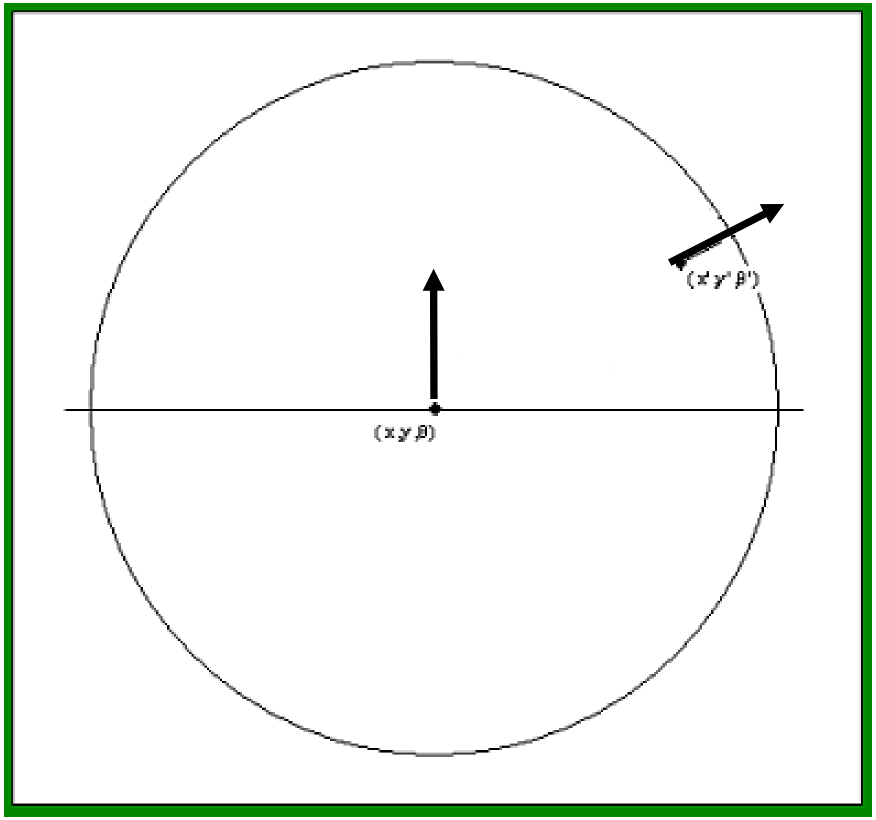
# Type 2 Maneuver

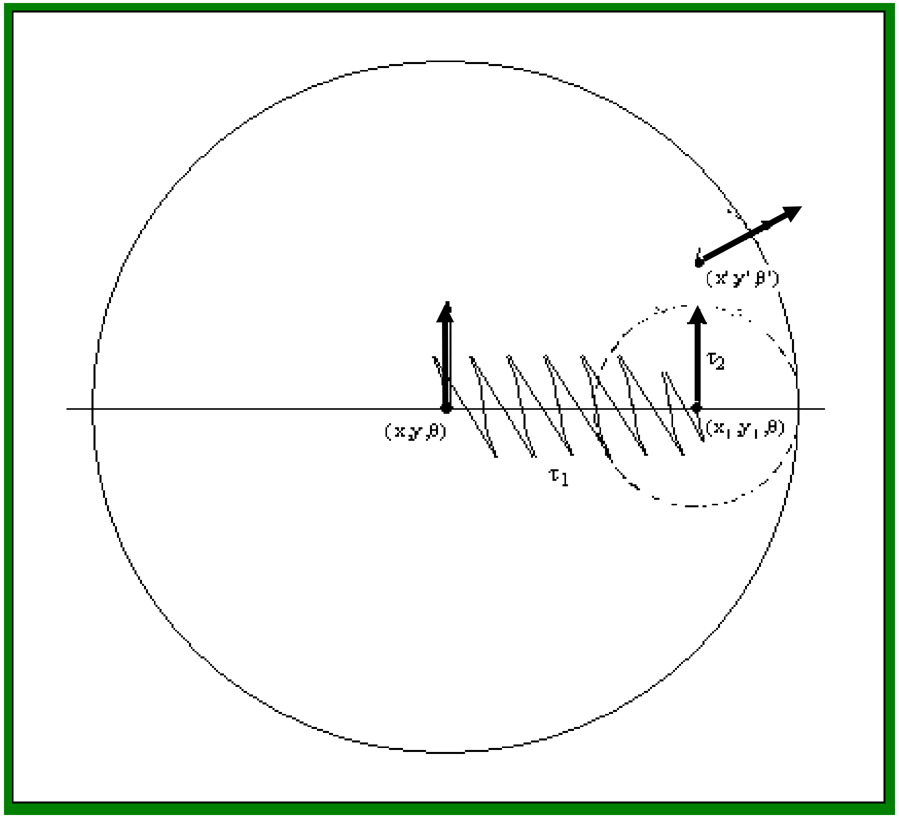



→ **Allows pure rotation**

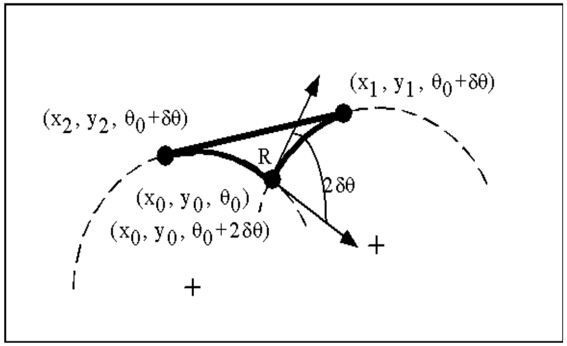
# Combination

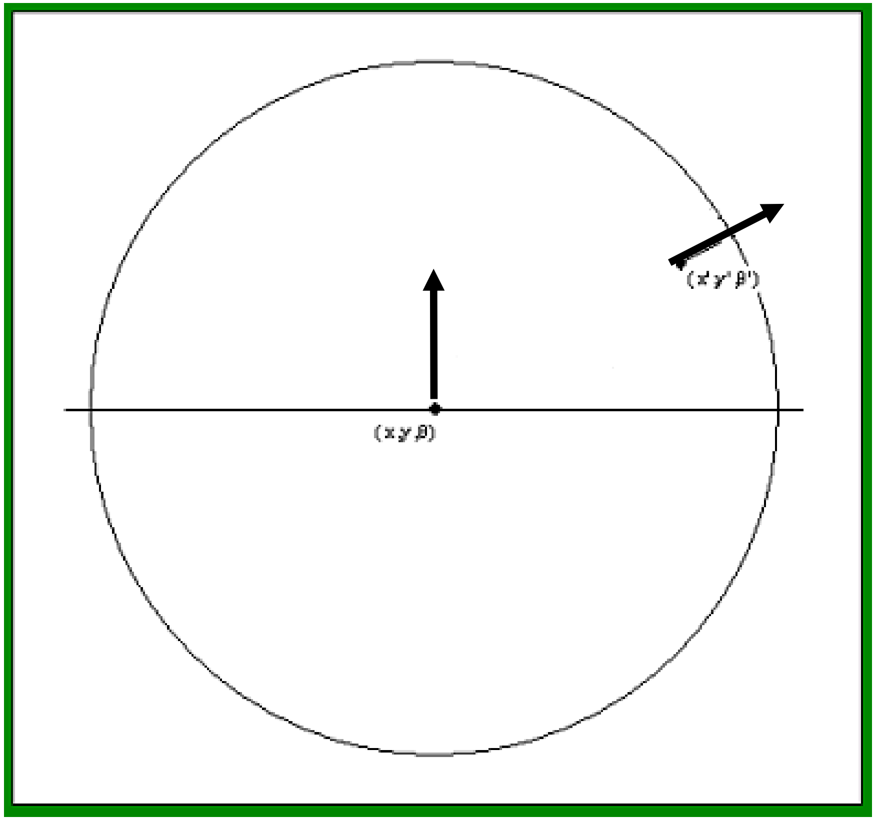


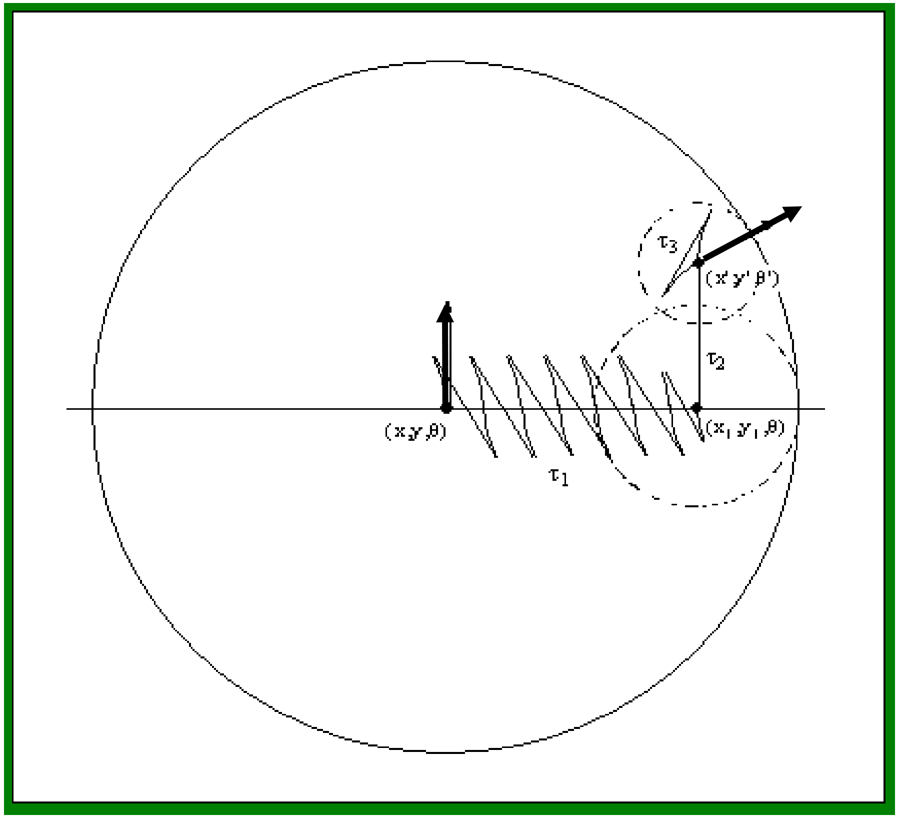





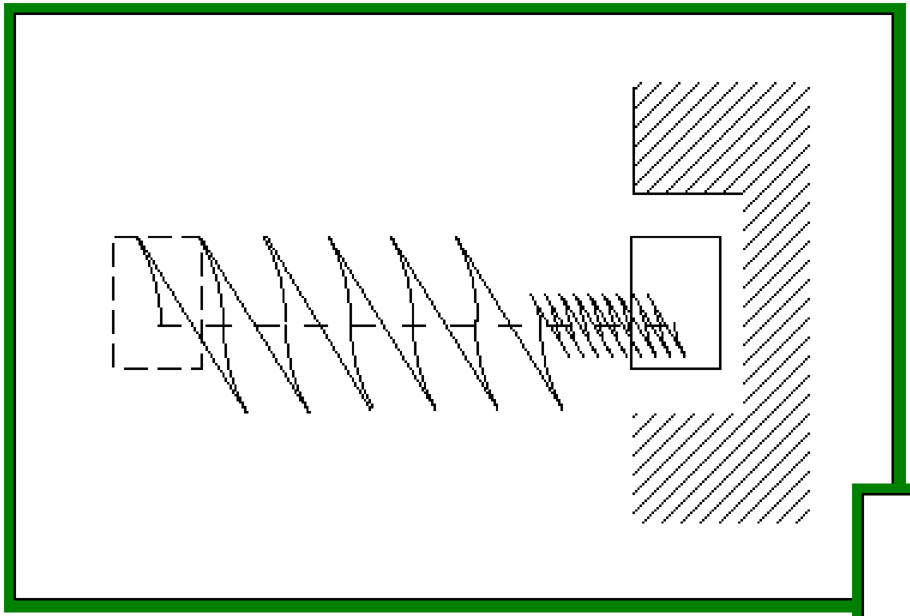
# Combination

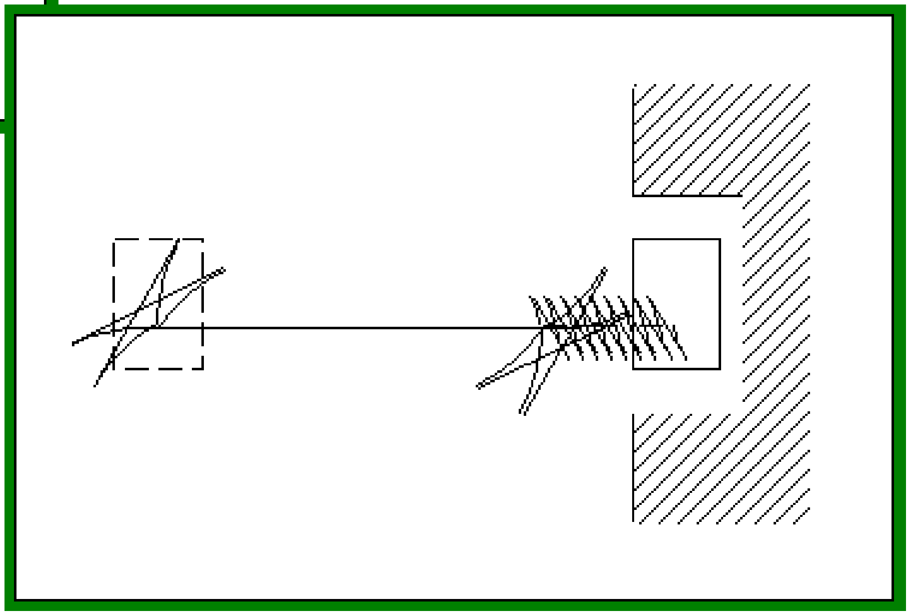


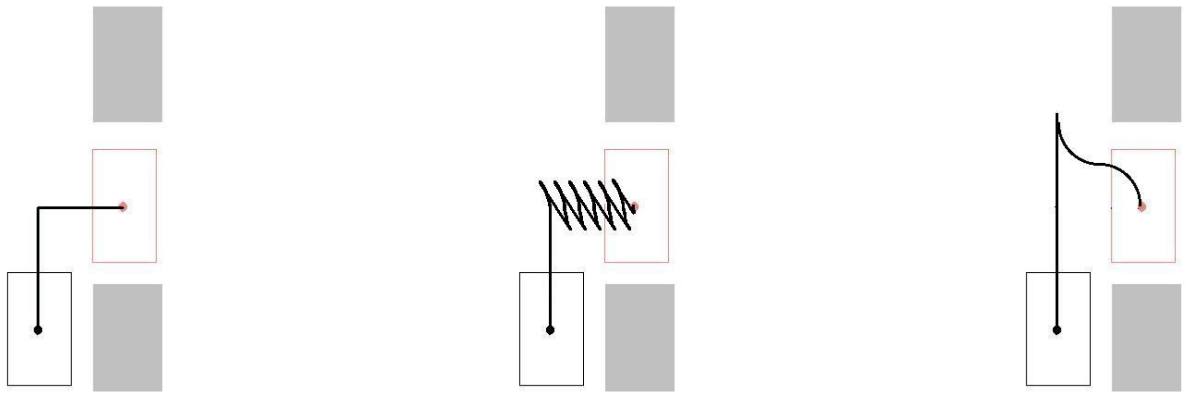
# Path Examples





# Drawbacks

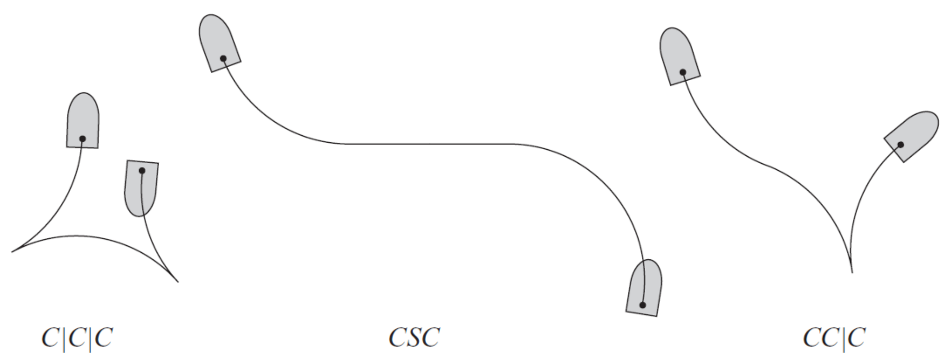
- Final path can be far from optimal



- Not applicable to car that can only move forward
  - e.g., think of an airplane

# Optimal Solution?

- Reed and Shepp (RS) Path
  - Optimal path must be one of **a discreet and computable set of curves**
  - Each member of this set consists of sequential straight-line segments and circular arcs at the car's **minimum turning radius**
- Notation
  - C – curve
  - S – straight line
  - "|" – switch direction
  - Subscript – traverse distance



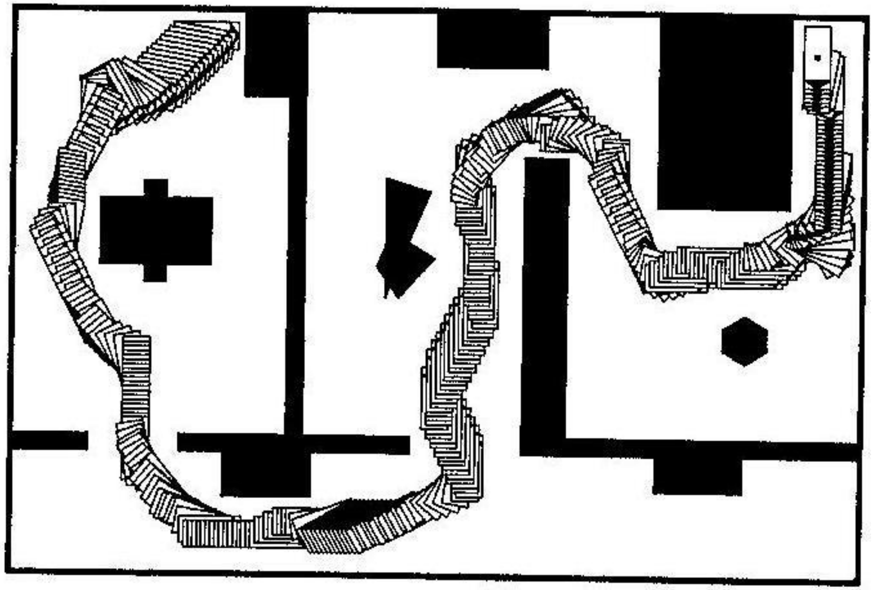
# Reeds and Shepp Paths

- Given any two configurations
  - The shortest RS paths between them is also the **shortest** path
  - The optimal path is guaranteed to be contained in the following set of path types

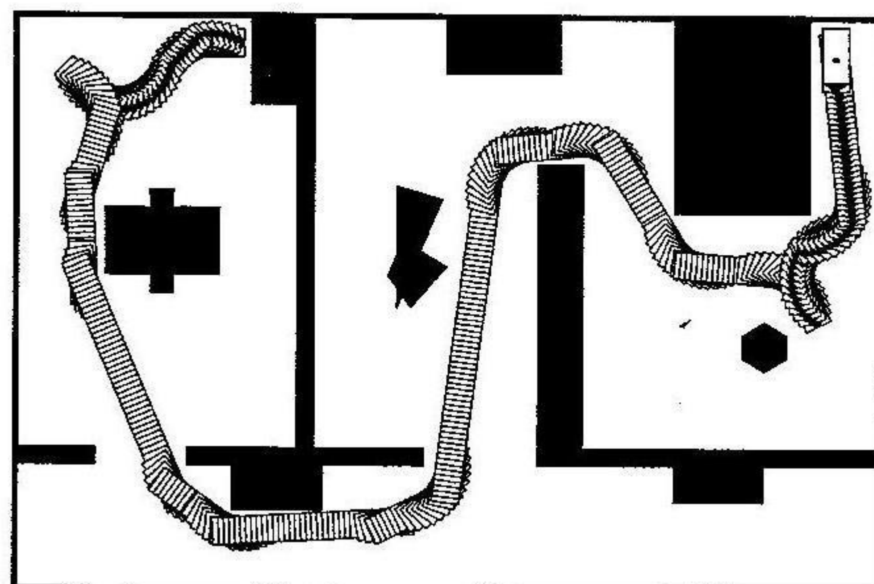
$$\{C\,|\,C\,|\,C,\;\; CC\,|\,C,\;\; C\,|\,CC,\;\; CC_a|C_aC,\;\; C\,|\,C_aC_a\,|\,C,$$
$$C\,|\,C_{\pi/2}SC,\;\; CSC_{\pi/2}\,|\,C,\;\; C\,|\,C_{\pi/2}SC_{\pi/2}\,|\,C,\;\; CSC\}$$

- Strategy
  - **In the absence of obstacles**, look up the optimal path from the above set using a map indexed by the goal configuration relative to the initial configuration
  - Shortest path may not be unique

# Example of Generated Path



Holonomic



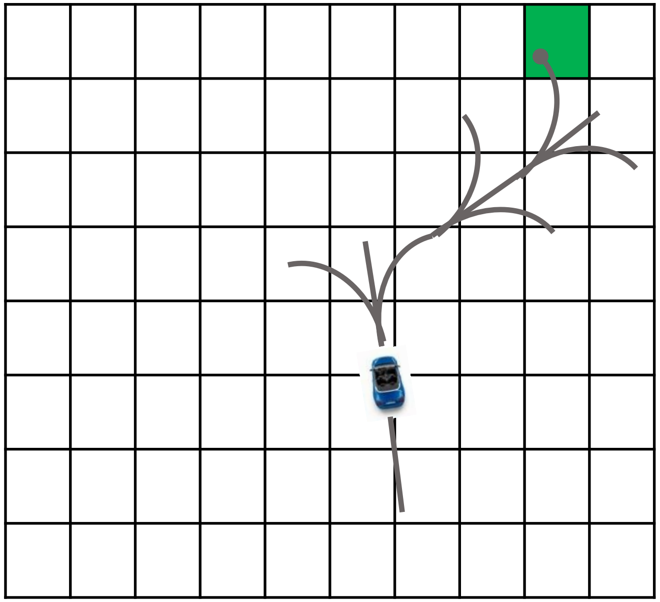
Nonholonomic

# Discrete Planning

- Strategies
  - Search for **sequence of primitives** to get to a goal state
  - Compute **State Lattice**, search for sequence of states in lattice
    - By construction of state lattice, can always get between these states

# Sequencing of Primitives

- Discretize control space
  - Barraquand & Latombe, 1993
    - 3 arcs (+ reverse) at $\kappa_{max}$
    - Discontinuous curvature
    - Cost = number of reversals
    - Dijkstra's Algorithm



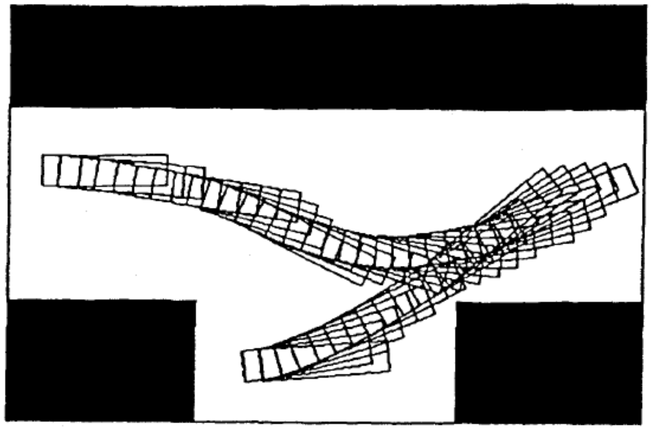


Fig. 4. Parking a car.

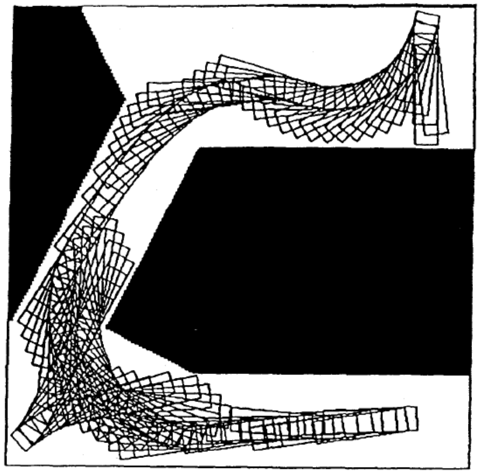
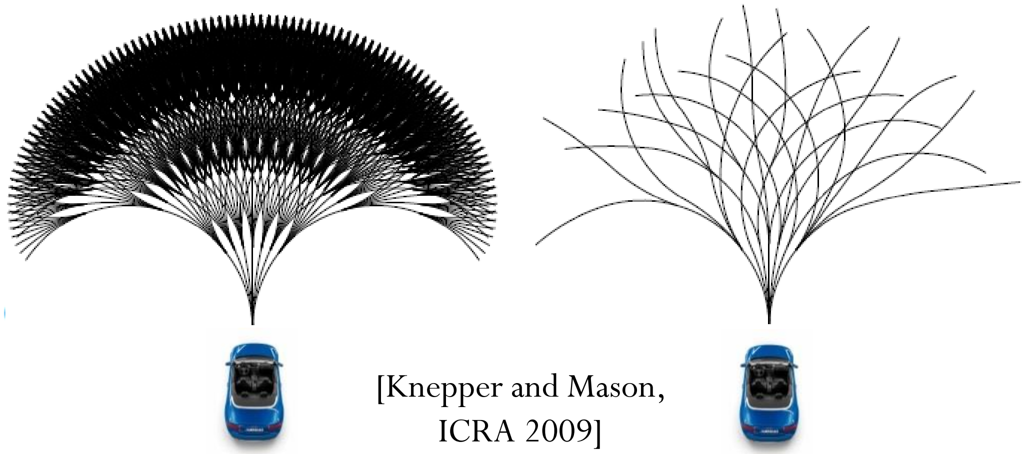


Fig. 5. Car maneuvering in a cluttered workspace.

# Sequencing of Primitives

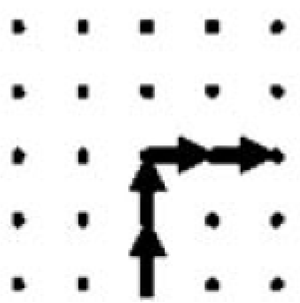
- Choice of set of primitives affects
  - Completeness
  - Optimality
  - Speed
- Seeks to build good (small) sets of primitives



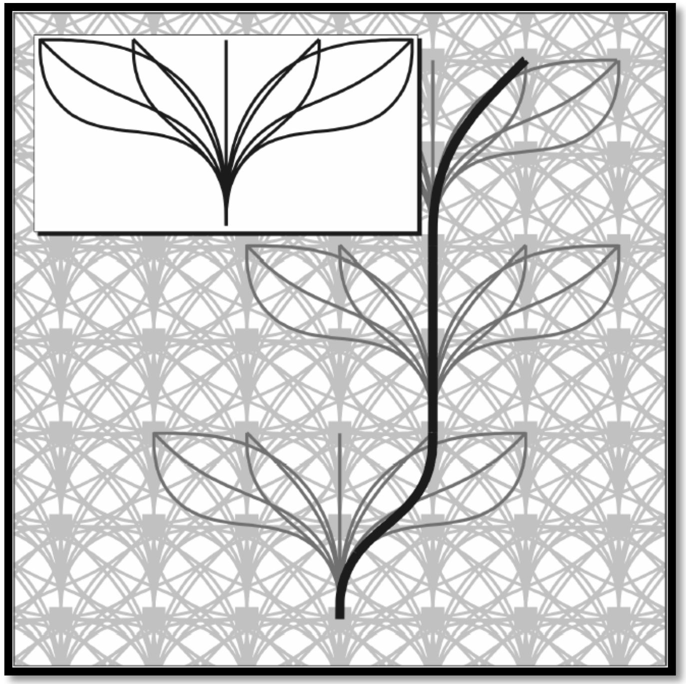
[Knepper and Mason, ICRA 2009]

# 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

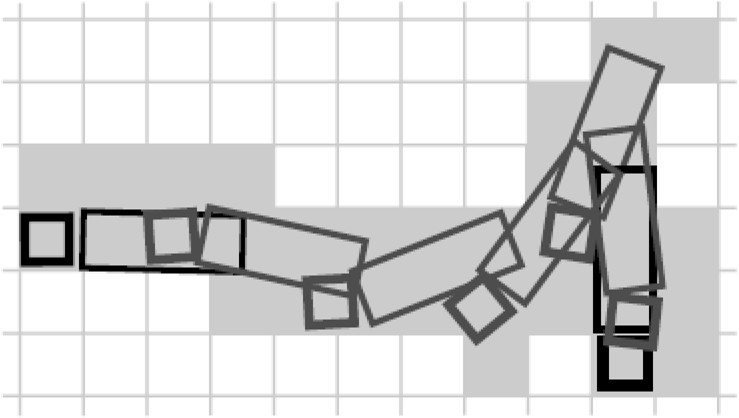


Traditional lattice yields discontinuous motion



# State Lattice

- Impose continuity constraints at graph vertices
- Search state lattice like any graph (i.e. A*)
- Pre-compute swept volume of robot for each primitive for faster collision check



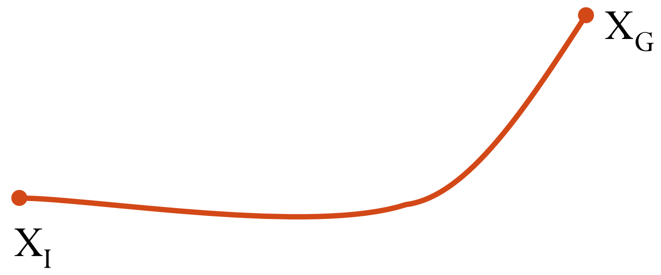
Pivtoraiko et al. 2009

# Sampling-Based Planning

- 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** – position and velocity
  - 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

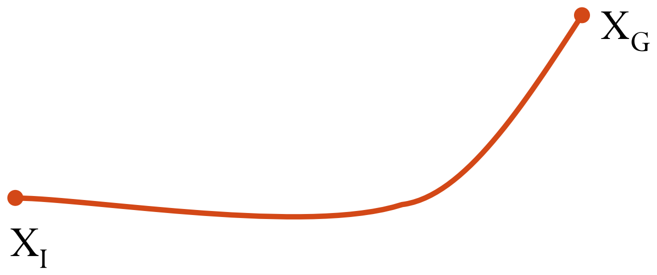
# PRM-style Non-Holonomic Planning

- Same as regular PRM
  - Sampling, graph building, and query strategies
- Problem
  - Local planner needs to reach an **EXACT** state (i.e. a given node) while obeying non-holonomic constraints



# PRM-style Non-Holonomic Planning

- 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



# RRT-style Non-Holonomic Planning

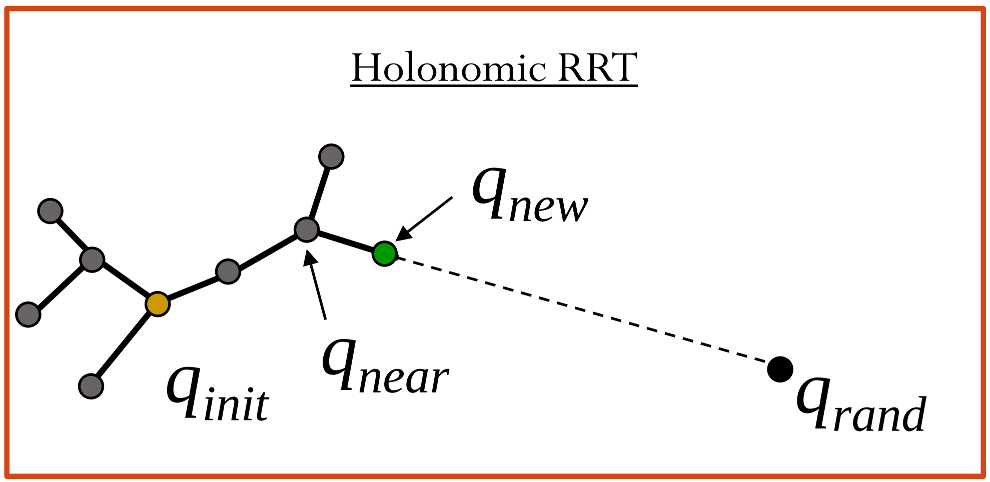
- RRT was originally proposed as a method for non-holonomic planning
- Sampling and tree building is the same as regular RRT
- Problem?
  - Not all straight lines are valid, can't extend toward nodes
  - Use **motion primitives** to get as close to target node as possible

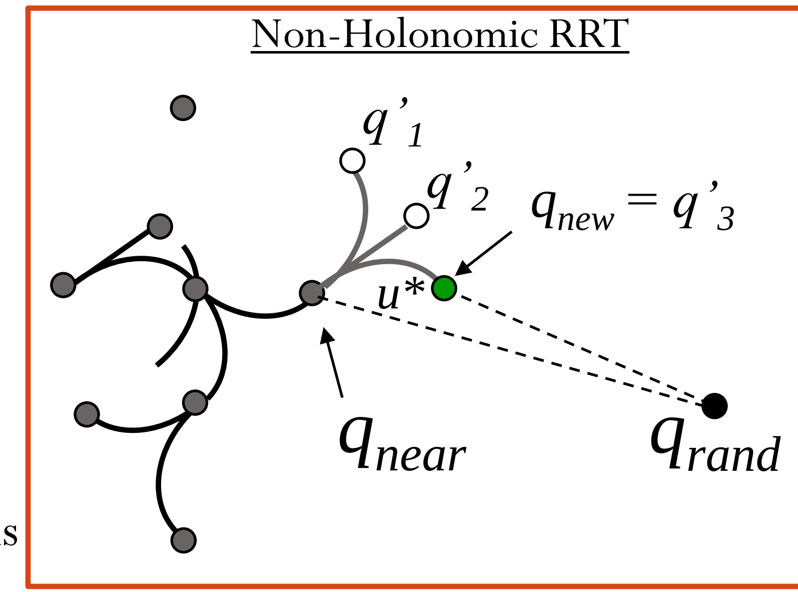
# RRTs for Non-Holonomic Systems

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

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

$$\text{chose } u_* = \arg\min(d(q_{rand}, q'))$$

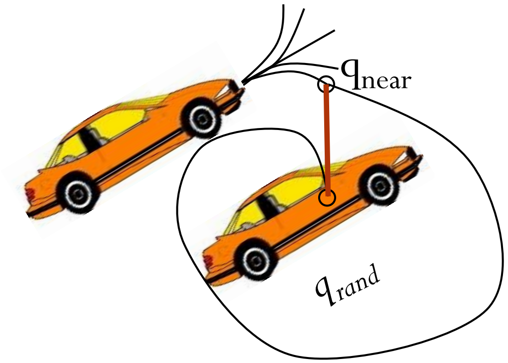



- You probably won't reach $q_{rand}$ by doing this
  - Key point: No problem, you're still exploring!

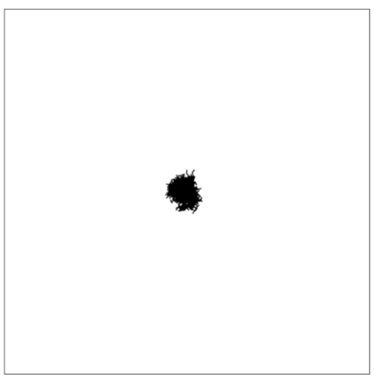
# RRTs and Distance Metrics

- Hard to define *d*, the distance metric
  - Mixing velocity, position, rotation ,etc.

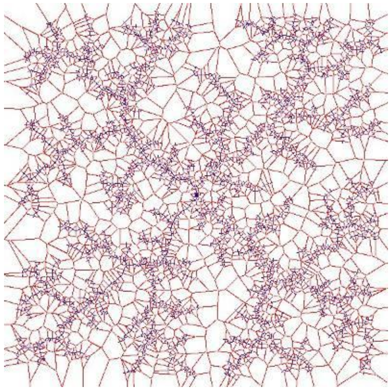



Configurations are close according to Euclidian metric, but actual distance is large

- How do you pick a good $q_{near}$?

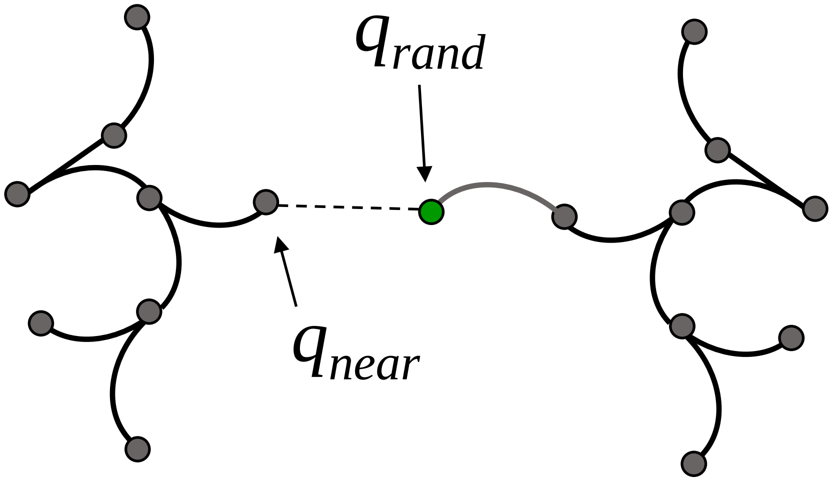


Random Node Choice
(bad distance metric)



Voronoi Bias
(good distance metric)

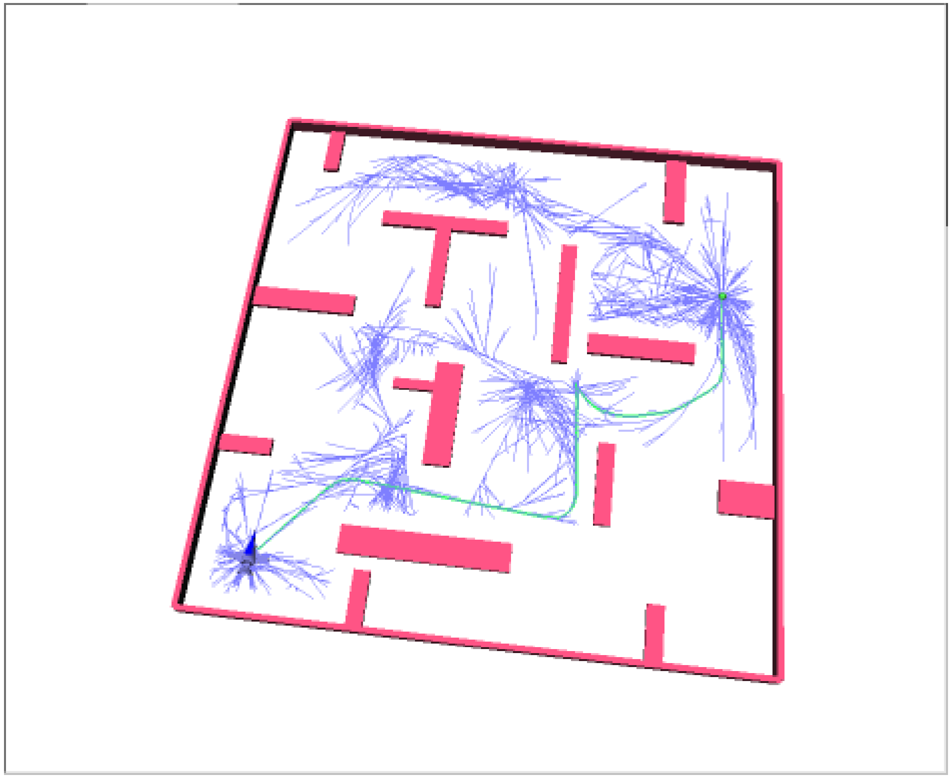
# BiDirectional Non-Holonomic RRT




- How do we bridge these two points?

# Non-holonomic Smoothing

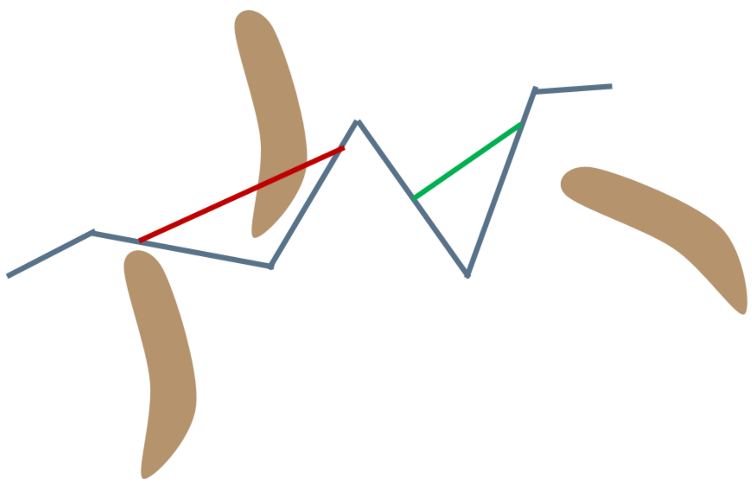
- Similar to holonomic case, paths produced can be highly suboptimal



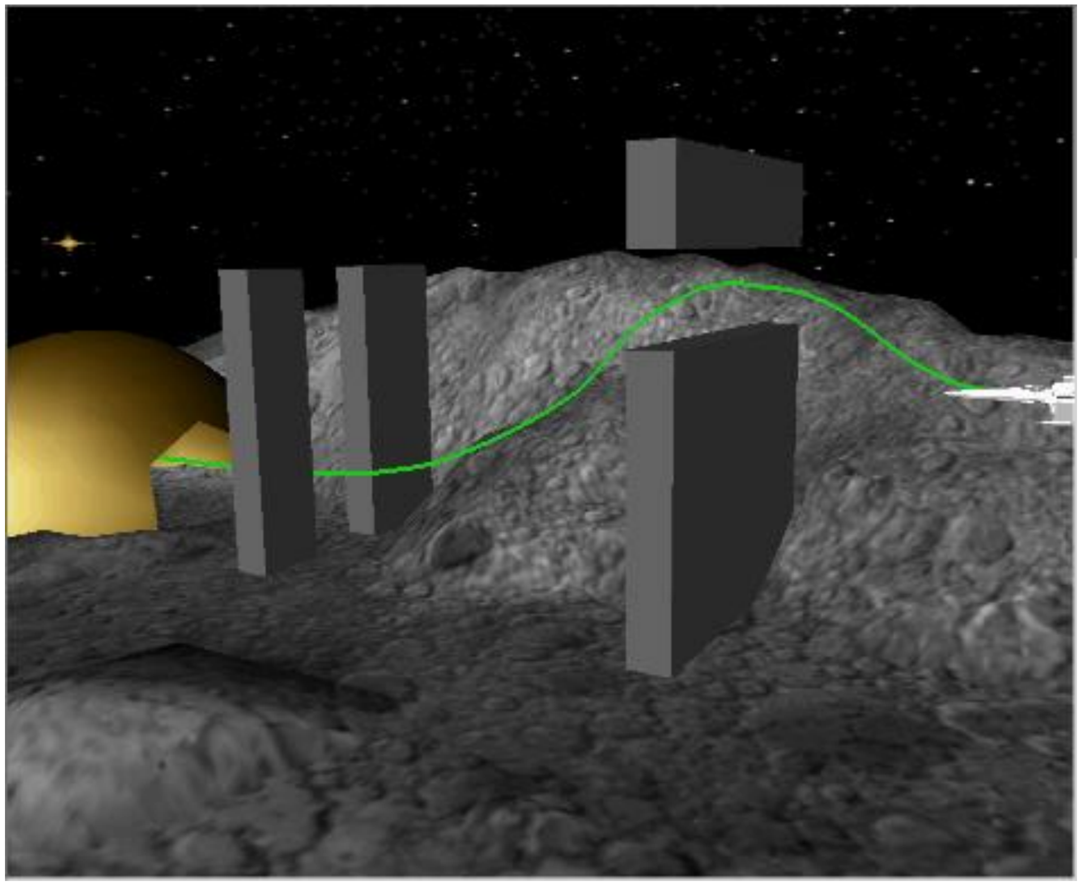
Hovercraft with 2 Thrusters in 2D

# 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
  - Search for sequence of primitives to get to a goal state
  - 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

- Start reading papers from class website
  - Bring questions to class
- Make sure to read Presentation Guidelines
- Make sure to look at Presentation Grading Sheet
- Make sure to look at Review Guidelines