Trajectory and Robotics Planning

Jiří Vokřínek
A4M33PAH - 2.4.2012

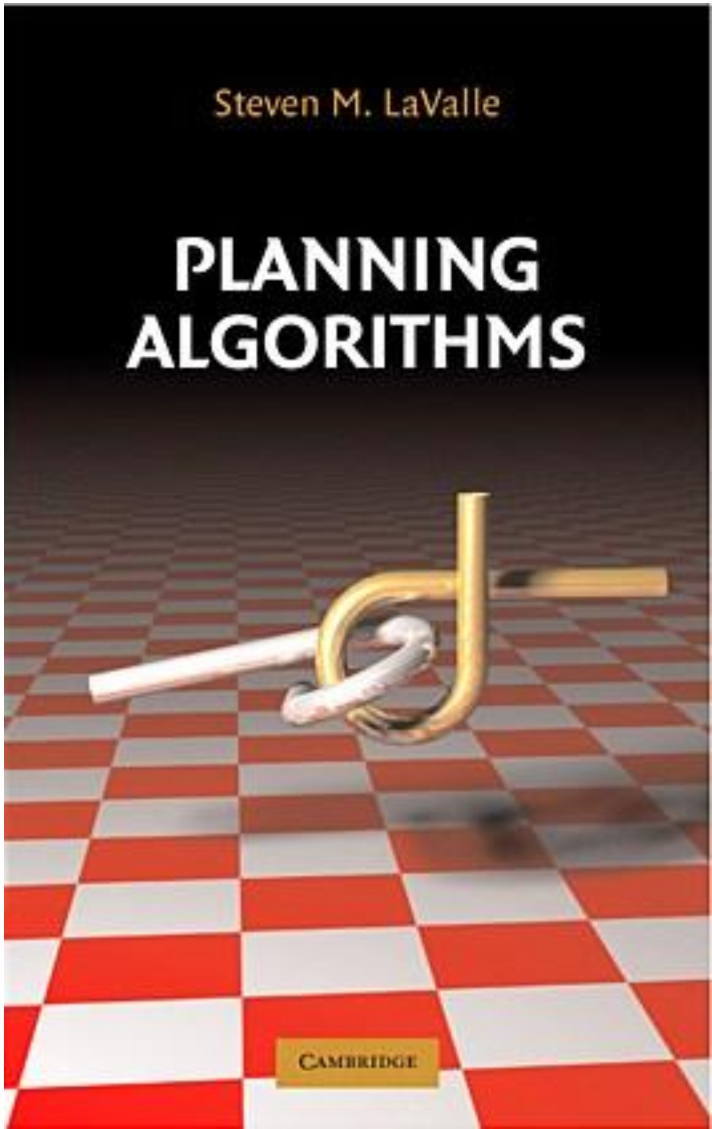
Literature

Steven M. LaValle.

Planning Algorithms.

Cambridge University Press, 2006.

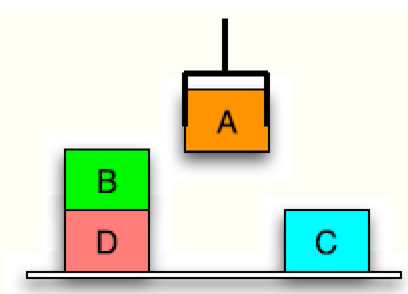
Available online:
http://planning.cs.uiuc.edu/

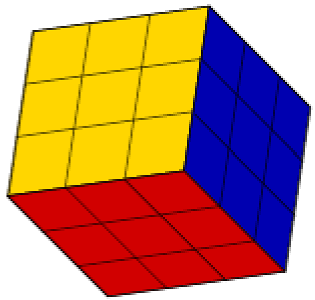


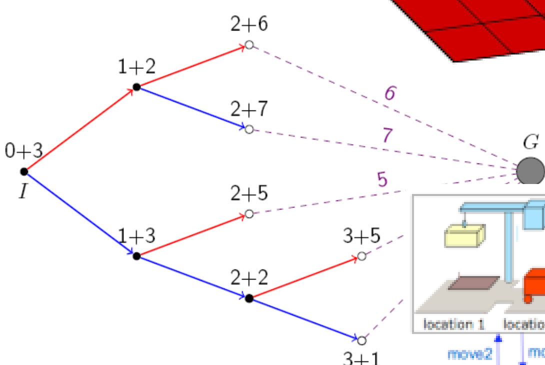
Robotic Planning Problems

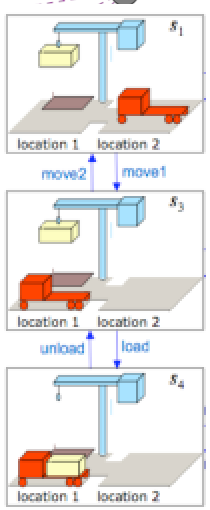
- Localization
- Mapping and Navigation
- Collision detection/avoidance
  - Obstacles
  - Other robots
- Motion planning
  - Roadmap, visibility graphs
  - Cell decomposition
  - Potential fields
- Coverage planning

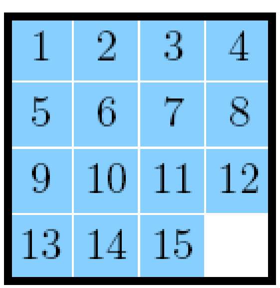
Robotic Planning Problems



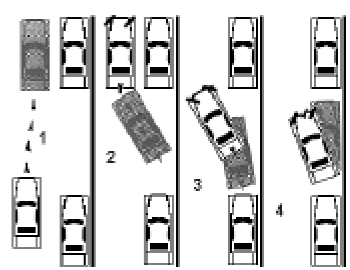


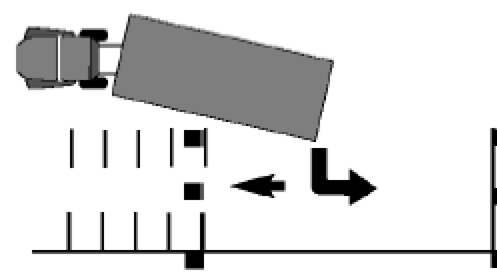






VS





Robotic Planning Problems

- Goal:
  - Compute motion strategies (geometric time-parameterized paths/trajectories)
  - Move to the specific position
  - Build a map of the region
  - Find a target, explore an area
  - Assemble/disassemble parts

Robotic Planning Problems

- Problem: compute a collision-free path for a moving object among static obstacles
- Input:
  - Geometry of a moving object and obstacles
  - Kinematics of the robot (degrees of freedom)
  - Initial and goal robot configurations (positions & orientations)
- Output: continuous sequence of collision-free robot configurations connecting the initial and goal configurations

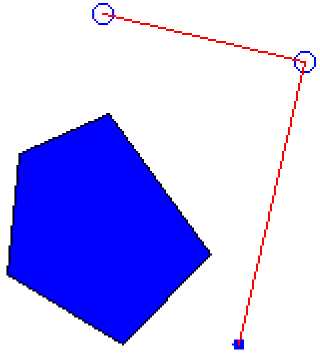
Robotic Planning Problems

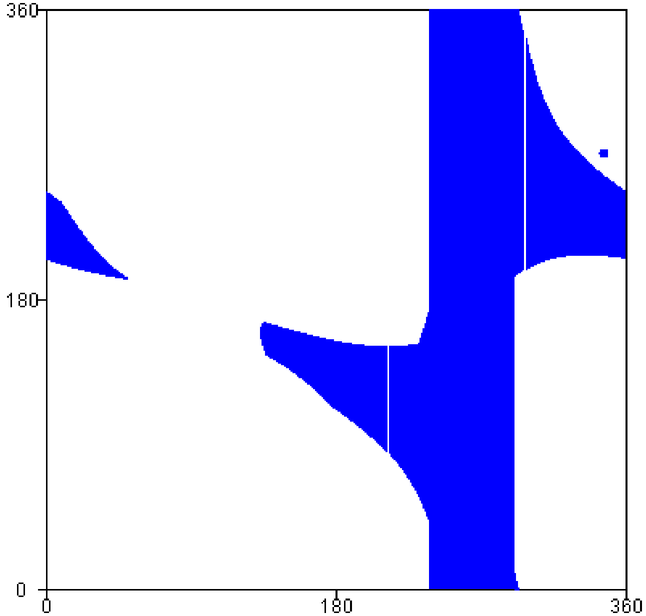
- Configuration space
- Sampling-based motion planning
- Combinatorial motion planning

- Several variants of the path planning problem have been proven to be PSPACE-hard.
- A complete algorithm may take exponential time (complete algorithm finds a path if one exists and reports no path exists otherwise).

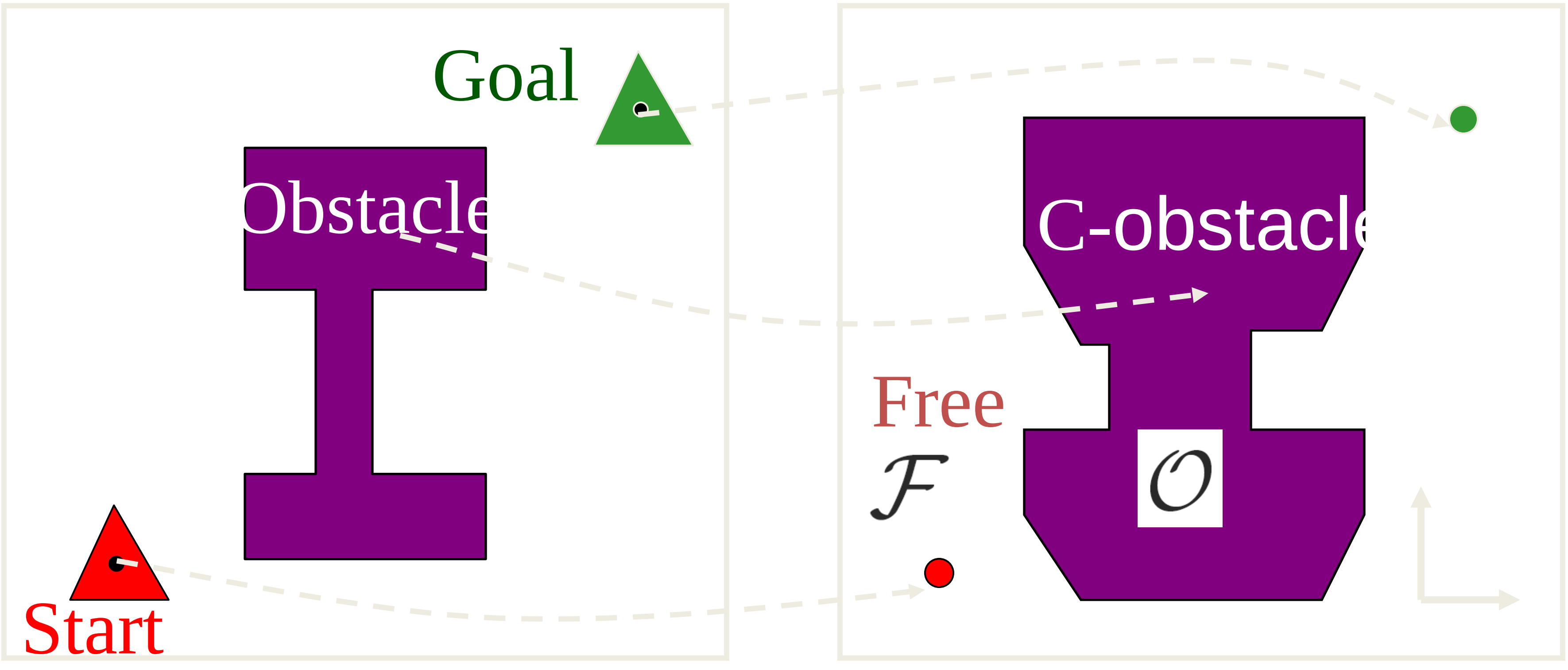
Configuration Space

- Number of degrees of freedom (dimension of configuration space)
- Geometric complexity
- http://ford.ieor.berkeley.edu/cspace/





Configuration Space 2D Translation



Problem Formulation for Point Robot

- Input:
  - Robot represented as a point in the plane
  - Obstacles represented as polygons
  - Initial and goal positions
- Output:
  - A collision-free path between the initial and goal positions

Problem Formulation for Point Robot

continuous representation

(configuration space formulation)

↓

discretization

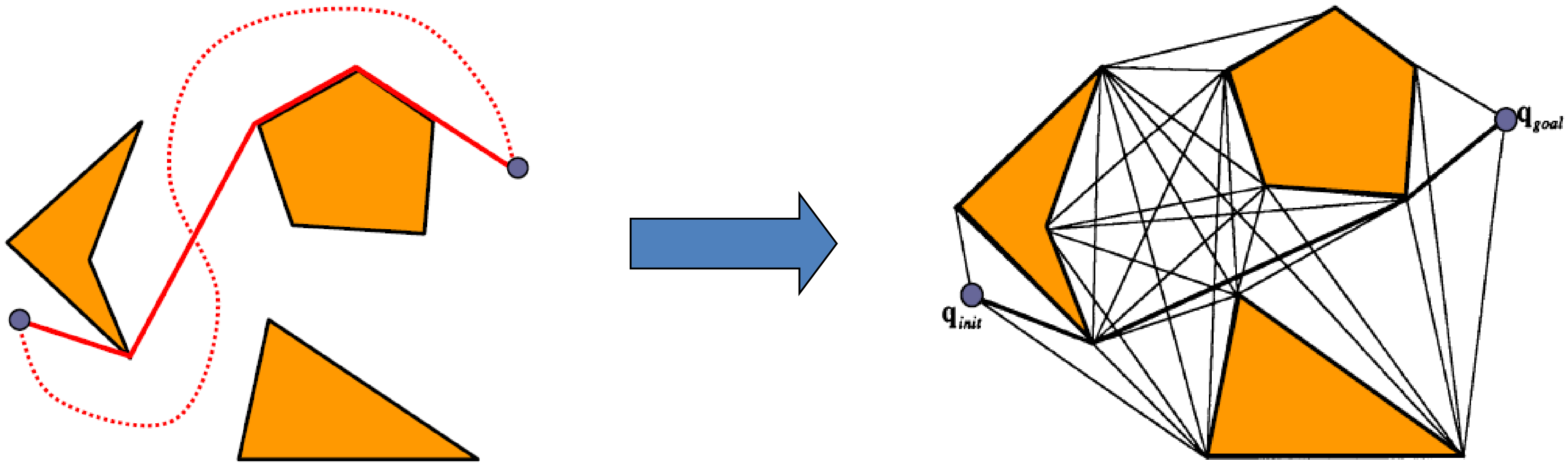
(random sampling, processing critical geometric events)

↓

graph searching

(breadth-first, best-first, A*)

Visibility Graph Method



- Nodes: start, goal, obstacle vertex
- Edges: complete graph without edges that intersect the obstacles
- $O(n^3)$ time, $O(n^2)$ space (naive algorithm)

A Simple Algorithm for Building Visibility Graphs

Input: q_{init}, q_{goal}, polygonal obstacles
Output: visibility graph G

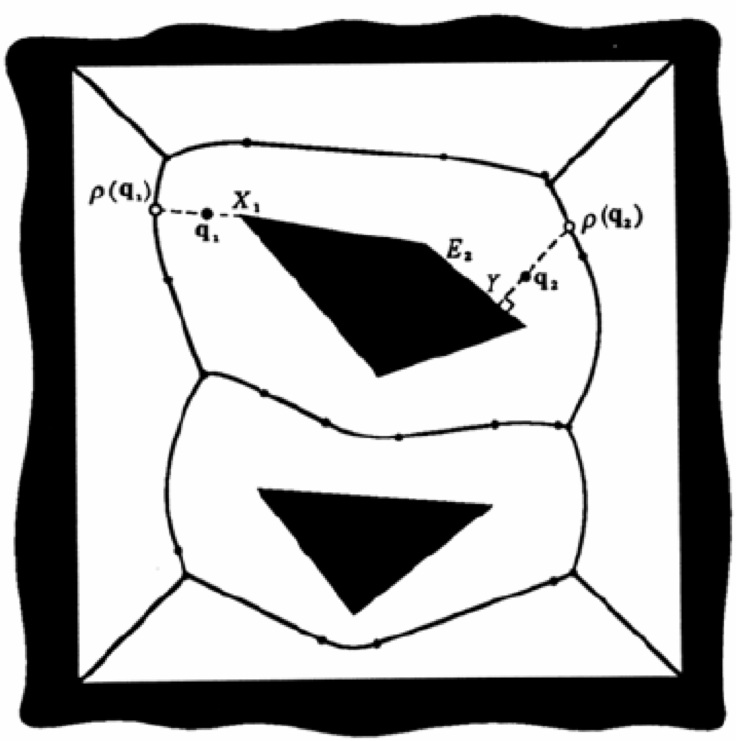
```
1: for every pair of nodes u,v
2:   if segment(u,v) is an obstacle edge then
3:     insert edge(u,v) into G;
4:   else
5:     for every obstacle edge e
6:       if segment(u,v) intersects e
7:         go to (1);
8:     insert edge(u,v) into G.
```

Path Planning Approaches

- Roadmap – connectivity graph of the free space
- Cell decomposition – free space represented as a cell-grid
- Potential field – potential function over the free space that has a global minimum at the goal

Roadmap Methods

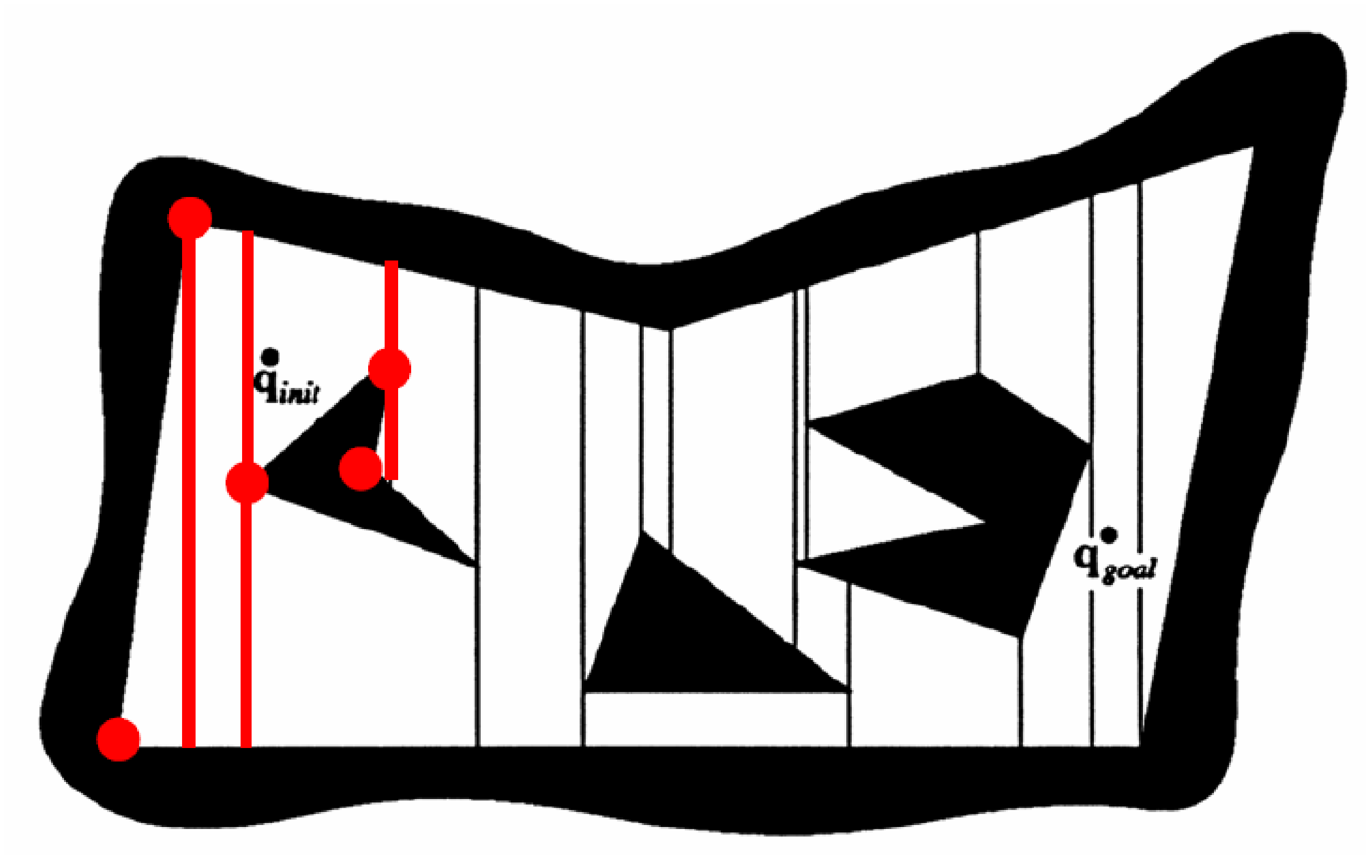
- Visibility graph
- Voronoi diagram - maximizes clearance
  - Generates a very safe roadmap which avoids obstacles as much as possible
- Silhouette
- Probabilistic roadmaps



Cell Decomposition Methods

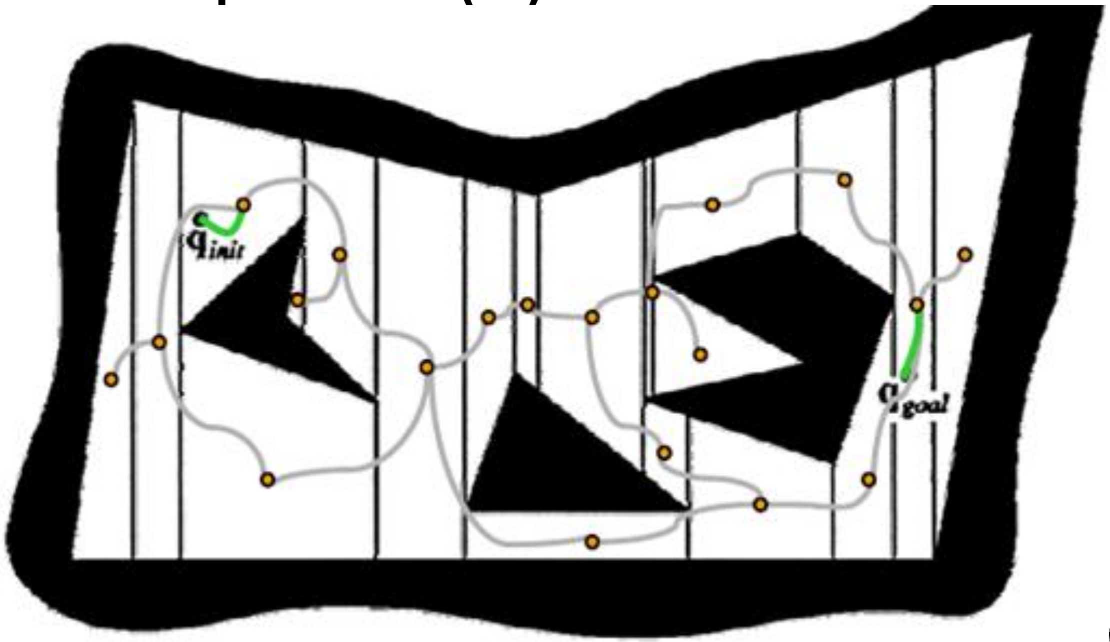
- Exact cell decomposition (trapezoids, triangles)
- Approximate cell decomposition (rectangles, squares)
  - Hierarchical space decomposition
  - Quad trees, octant trees, etc.

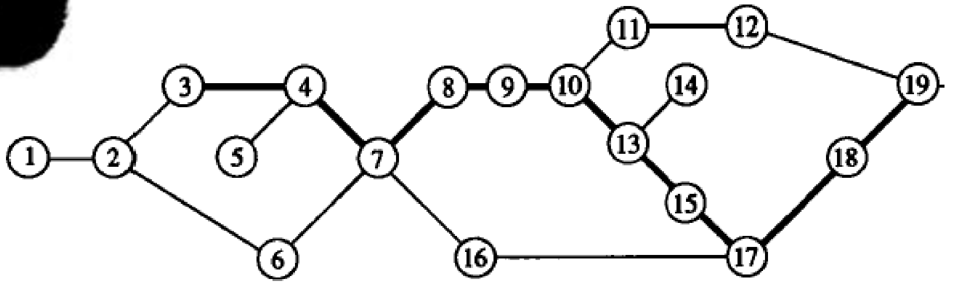
Trapezoidal Decomposition



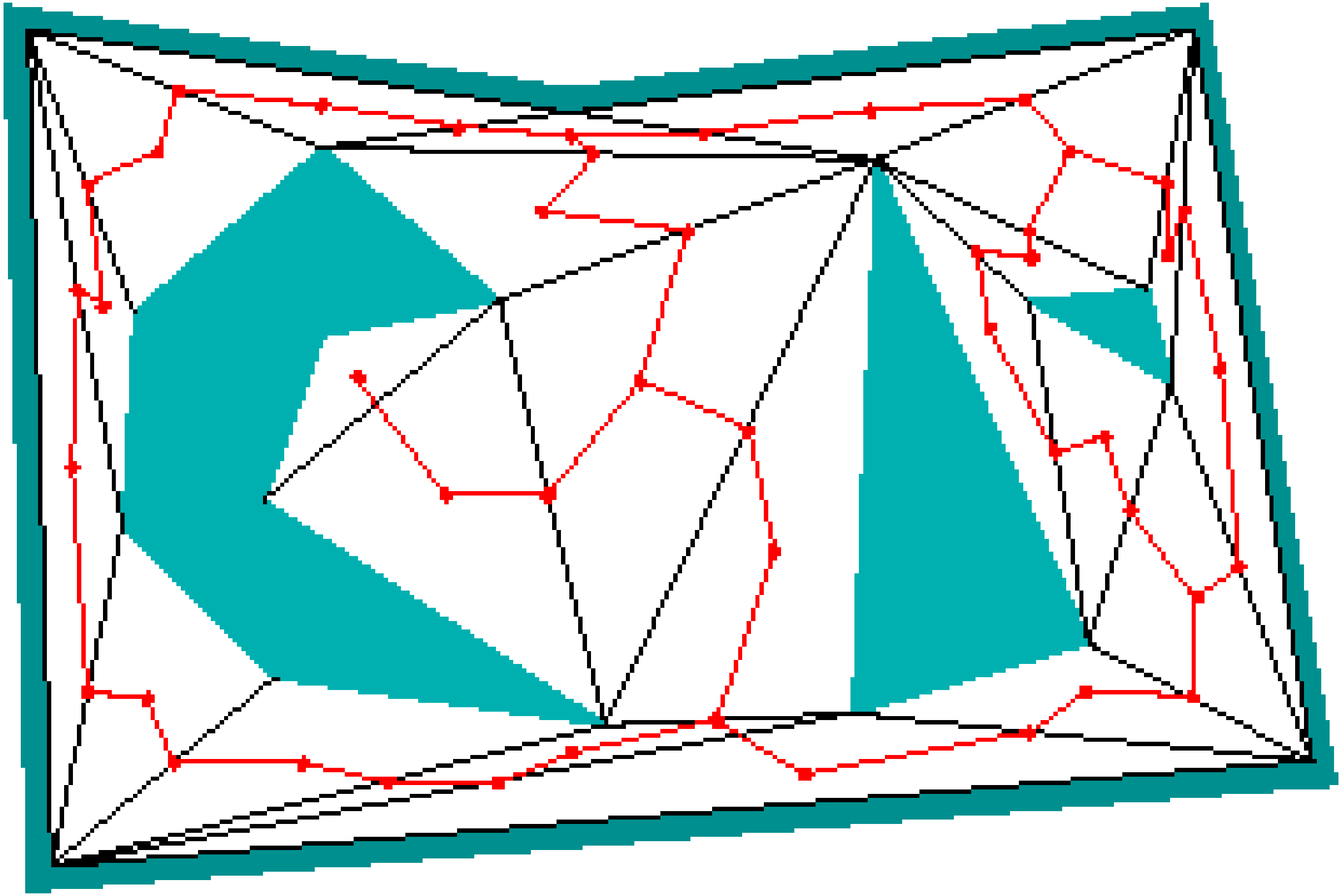
Trapezoidal Decomposition

- Running time O(n log n) by planar sweep
- Space O(n)

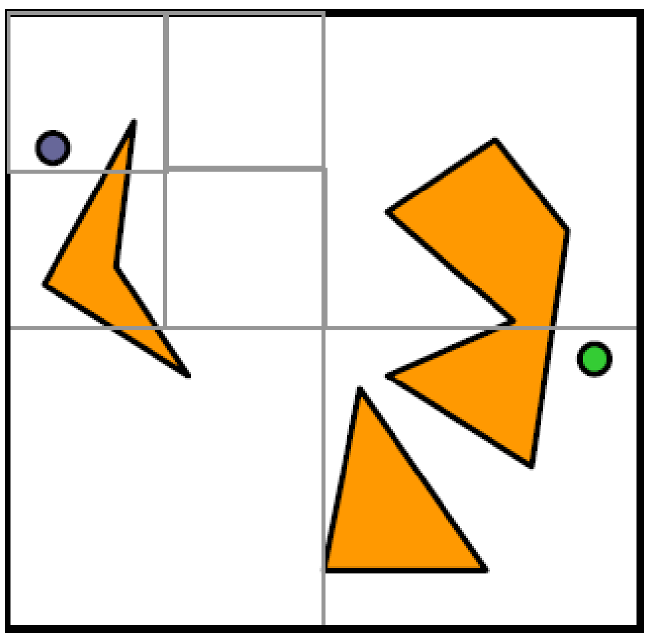


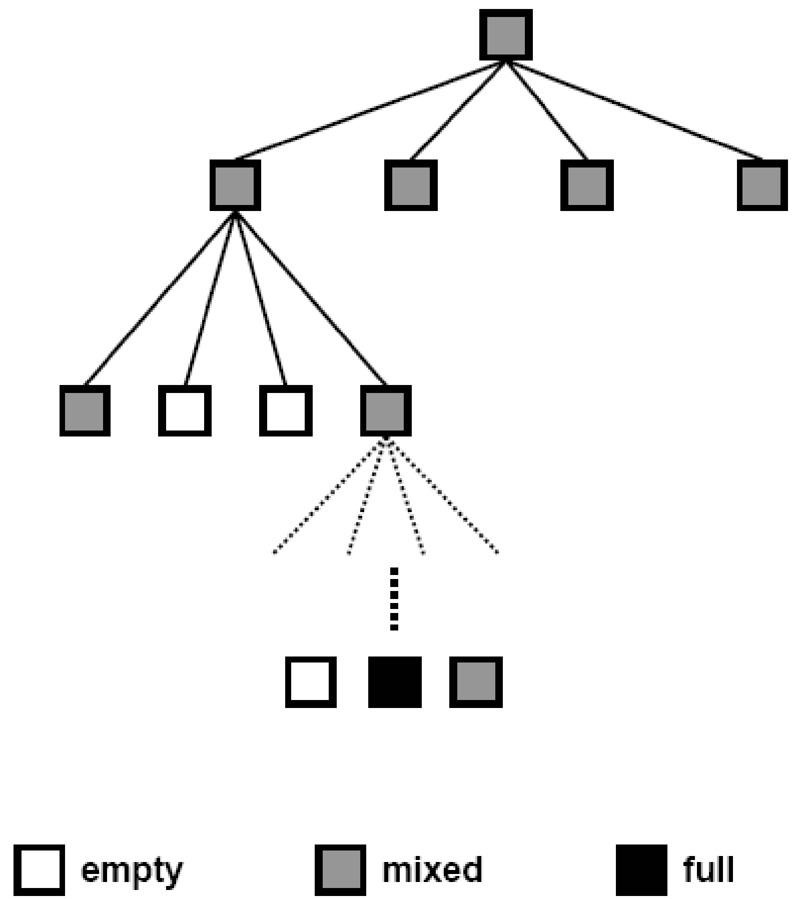


Triangular Decomposition

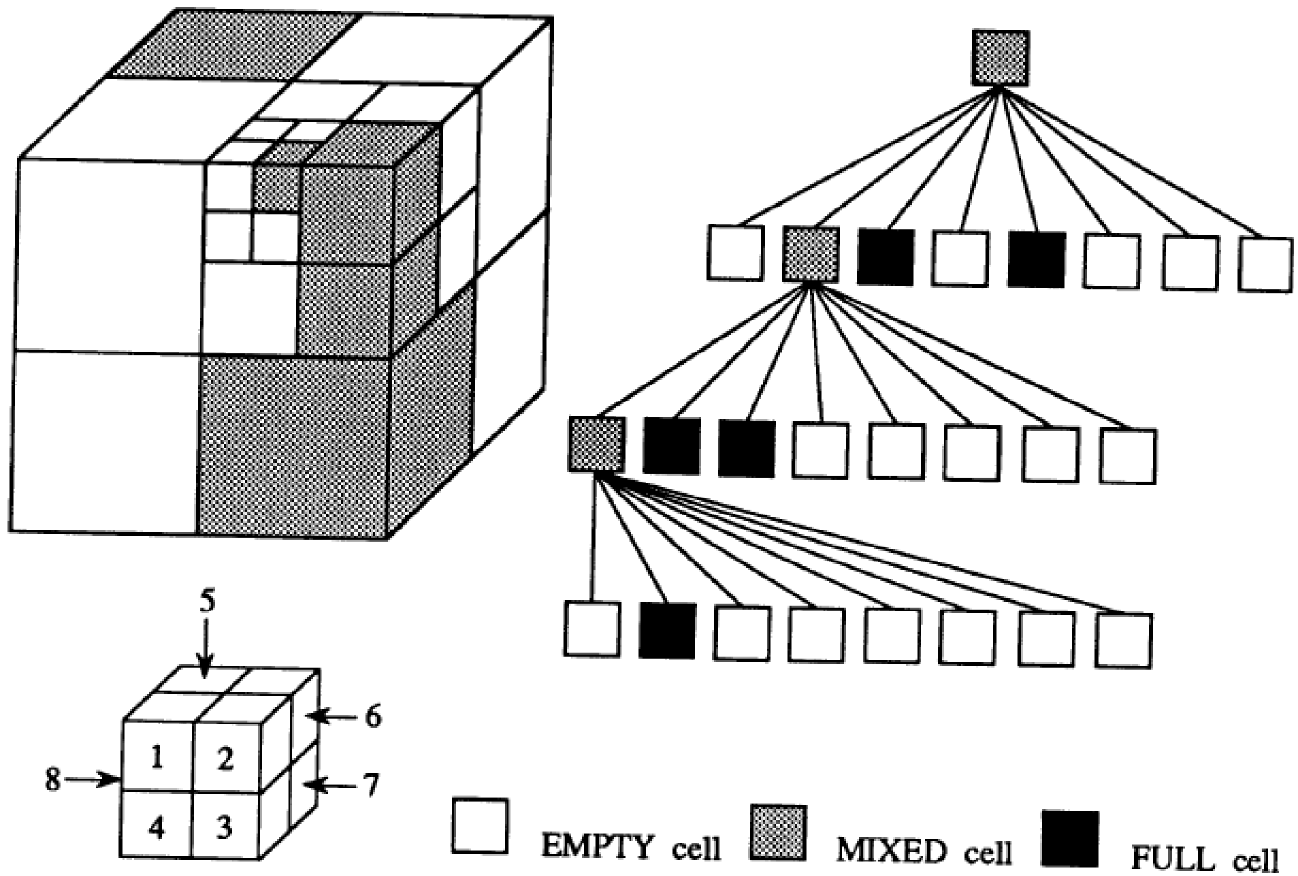


Quad Tree





Octant Tree



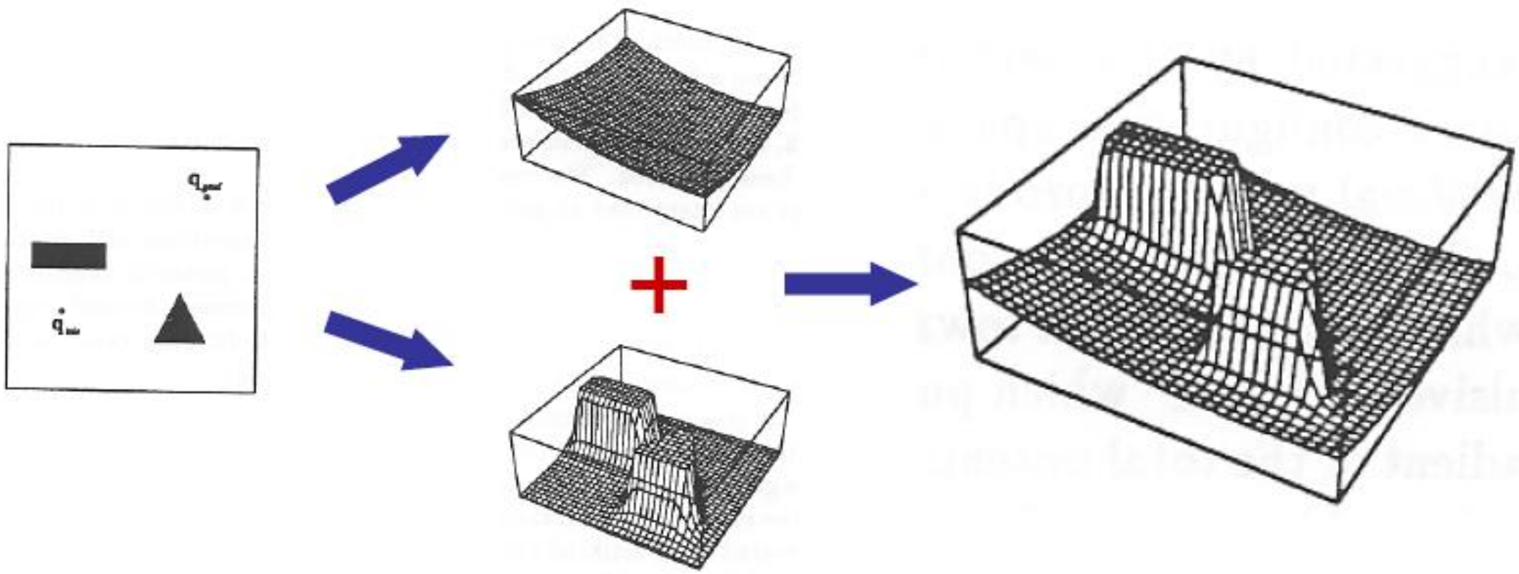
Algorithm Outline

1. Decompose the free space F into cells.
2. Search for a sequence of **mixed** or **free** cells that connect the initial and goal positions.
3. Further decompose the mixed.
4. Repeat (2) and (3) until a sequence of **free** cells is found.

Potential Field

- Scalar function over the free space
- Robot applies a force proportional to the negated gradient of the potential field
- A navigation function is an ideal potential field that
  - has global minimum at the goal
  - has no local minima
  - grows to infinity near obstacles
  - is smooth

Potential Field



Algorithm Outline

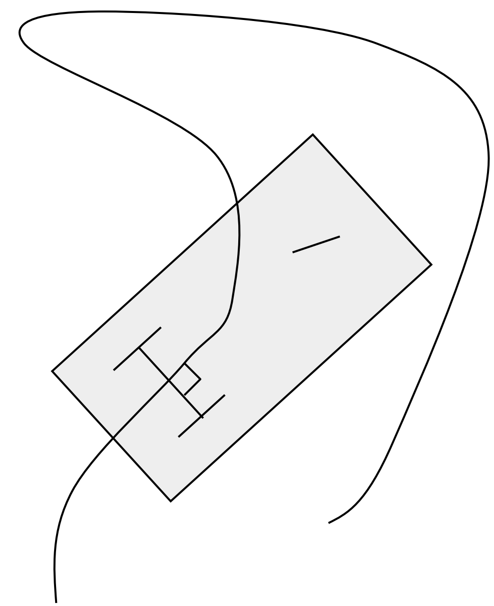
- Place a regular grid G over the configuration space
- Compute the potential field over G
- Search G using a best-first algorithm with potential field as the heuristic function

Extension of Robotic Problem

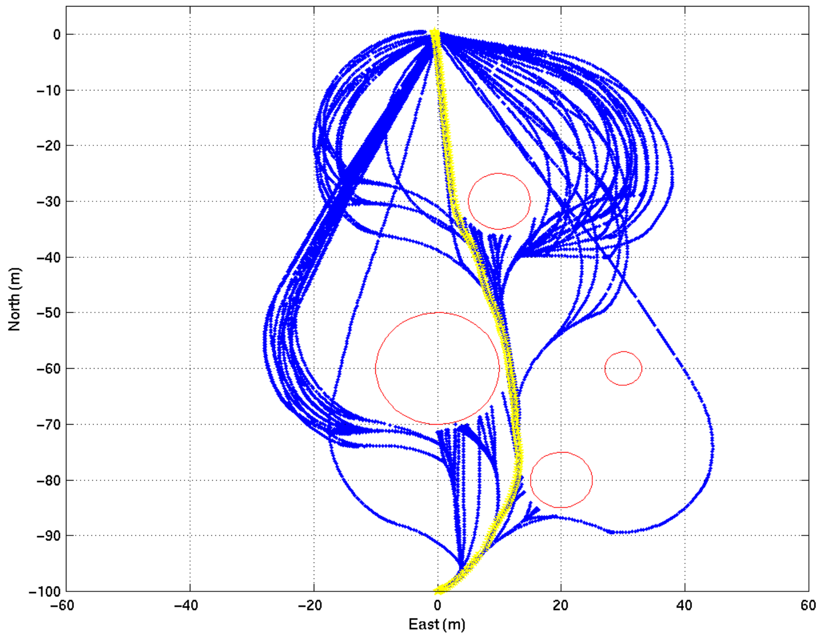
- More complex robots
  - Multiple robots
  - Movable objects, moving obstacles
  - Nonholonomic & dynamic constraints
  - Physical models and deformable objects
  - Sensorless motions (exploiting task mechanics)
  - Uncertainty in control and/or sensing
  - Optimal motion planning
  - Integration of planning and control

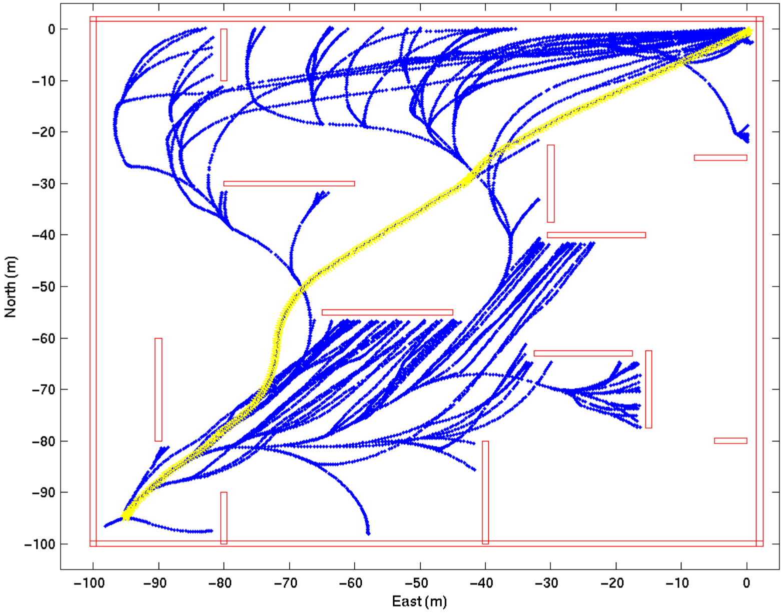
Integrating Dynamics

- Point robot trajectory vs. system dynamics
- Controlling problem
- Feasible trajectories reduce configuration space
- Trajectory primitives
  - Maneuvers
- Motion planning incl. system dynamics



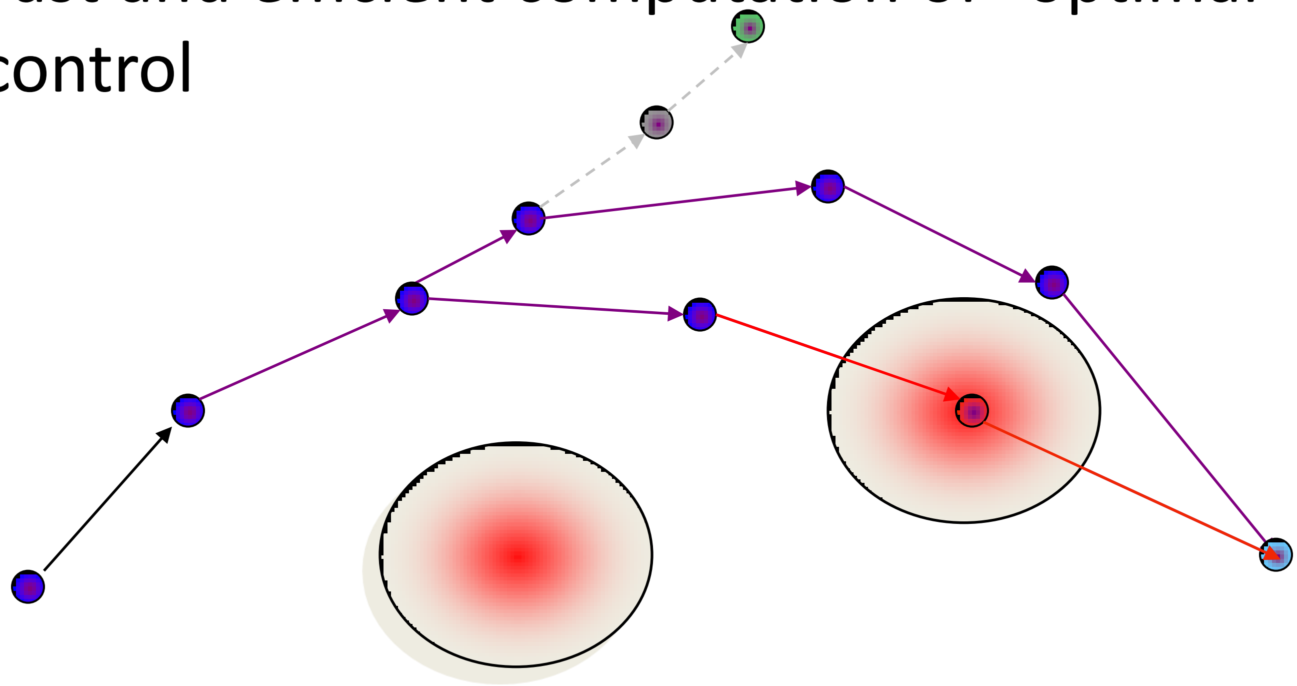
Maneuver Tree





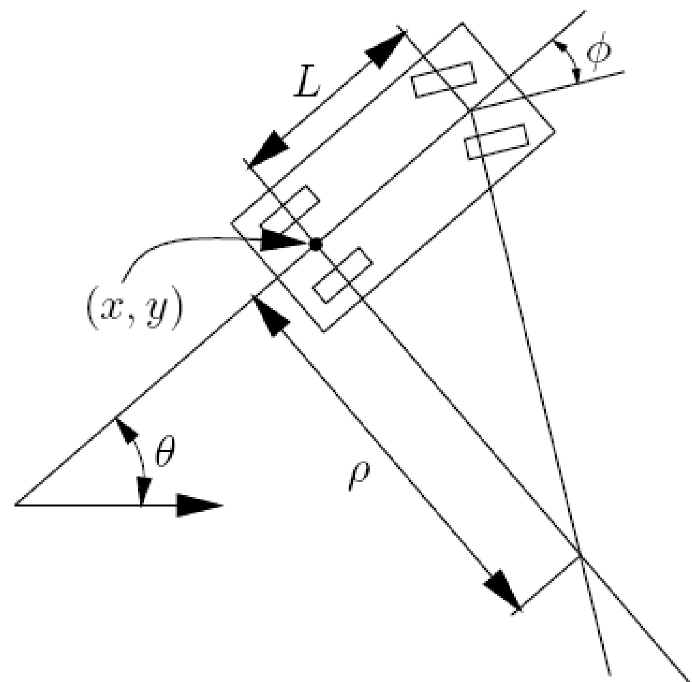
Rapidly-exploring Random Trees

- Optimal cost function in the free workspace case provides:
  - Pseudo-metric on the hybrid space
  - Fast and efficient computation of “optimal” control

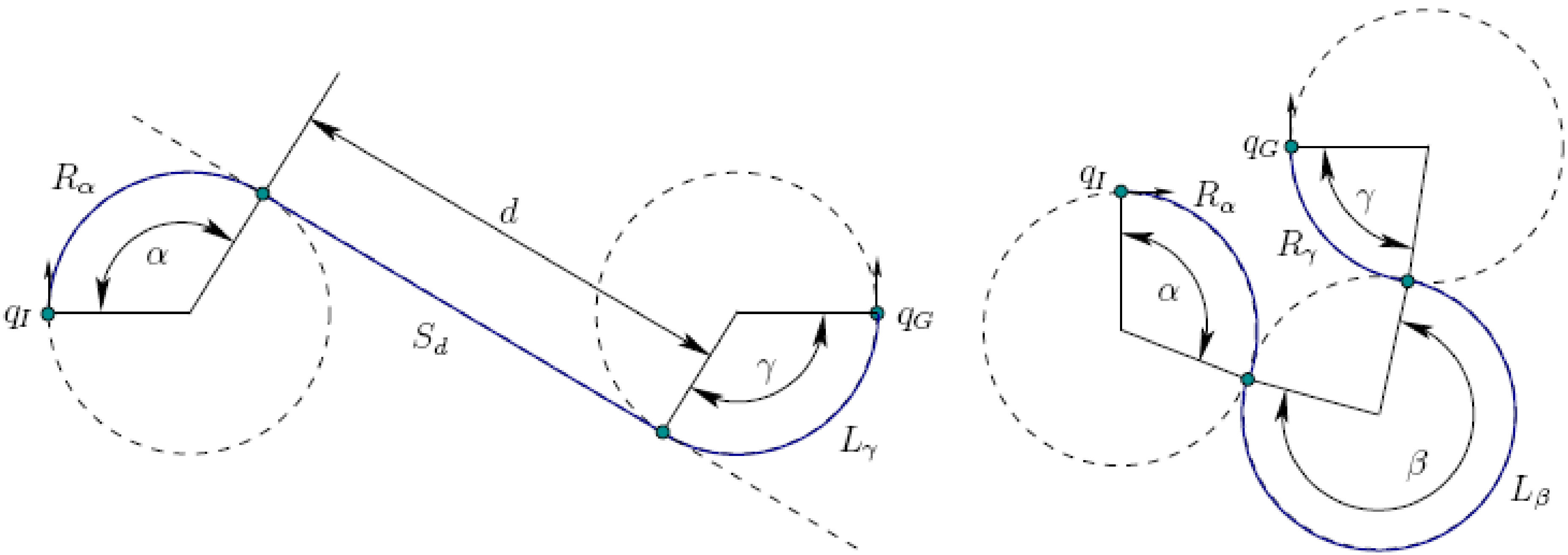


Maneuvers

- Dubins curves
  - Optimal path for wheeled vehicles
  - Consist of three primitives
  - No reverse direction allowed
- Reeds-Shepp Curves
  - Reverse direction allowed



Dubins Curves



$R_\alpha S_d L_\gamma$ $R_\alpha L_\beta R_\gamma$

Motion Planning

Point Robot

- Space Discretization
- Roadmap Construction
- Path Search
- Trajectory Smoothing
- Regulation to Waypoint

Rigid Body

- Space Discretization
- Maneuver Tree Construction
- Path Search
- Regulation to Maneuver

Hybrid Approach

- Space Discretization
- Roadmap Construction
- Path Search
- Trajectory Smoothing using Maneuvers
- Regulation to Maneuver

Adaptive Path Planner