Robot Motion Planning

James Bruce
Computer Science Department
Carnegie Mellon University

April 7, 2004

Agent Planning

An agent is a situated entity which can choose and execute actions within in an environment. A robot is a physical agent.

In general planning requires the following:

- A model of the agent's environment or world state
- A model of how its actions affect the world state

This allows us to predict the outcome of a sequence of actions for a given world state.

Planning is the reverse: The search for a sequence of actions to achieve a desired outcome.

Symbolic vs. Motion Planning

Symbolic Planning

- Environment state is composed of symbols (Ex: predicates)
- Operators are both actions and the world model

Motion Planning

- Environment state is made up of real-valued dimensions
- World model describes how real valued actions affect the agent's position

Robot Motion Planning

A mobile robot needs to *navigate*. Navigation is carrying out locomotion primitives to move between points while avoiding obstacles).

A fixed-base robot moves within a workspace, limited by the extents of its joints, to carry out actions. It must avoid other objects in its workspace as well as itself.

In other words, robots form and execute sound plans.

This can be achieved using motion planning, if we can create an *environment model* and an *action model*.

Environment Models for Robots

- Easy for fixed-base robots in structured environment
- Mobile robots often used in less structured environments
- Traditional navigation techniques avoided explicit modelling
- Explicit modelling is now practical
  - probabilistic localization and map building (SLAM)
  - distributed/networked sensors

Environment Models for Robots

Complicating factors:

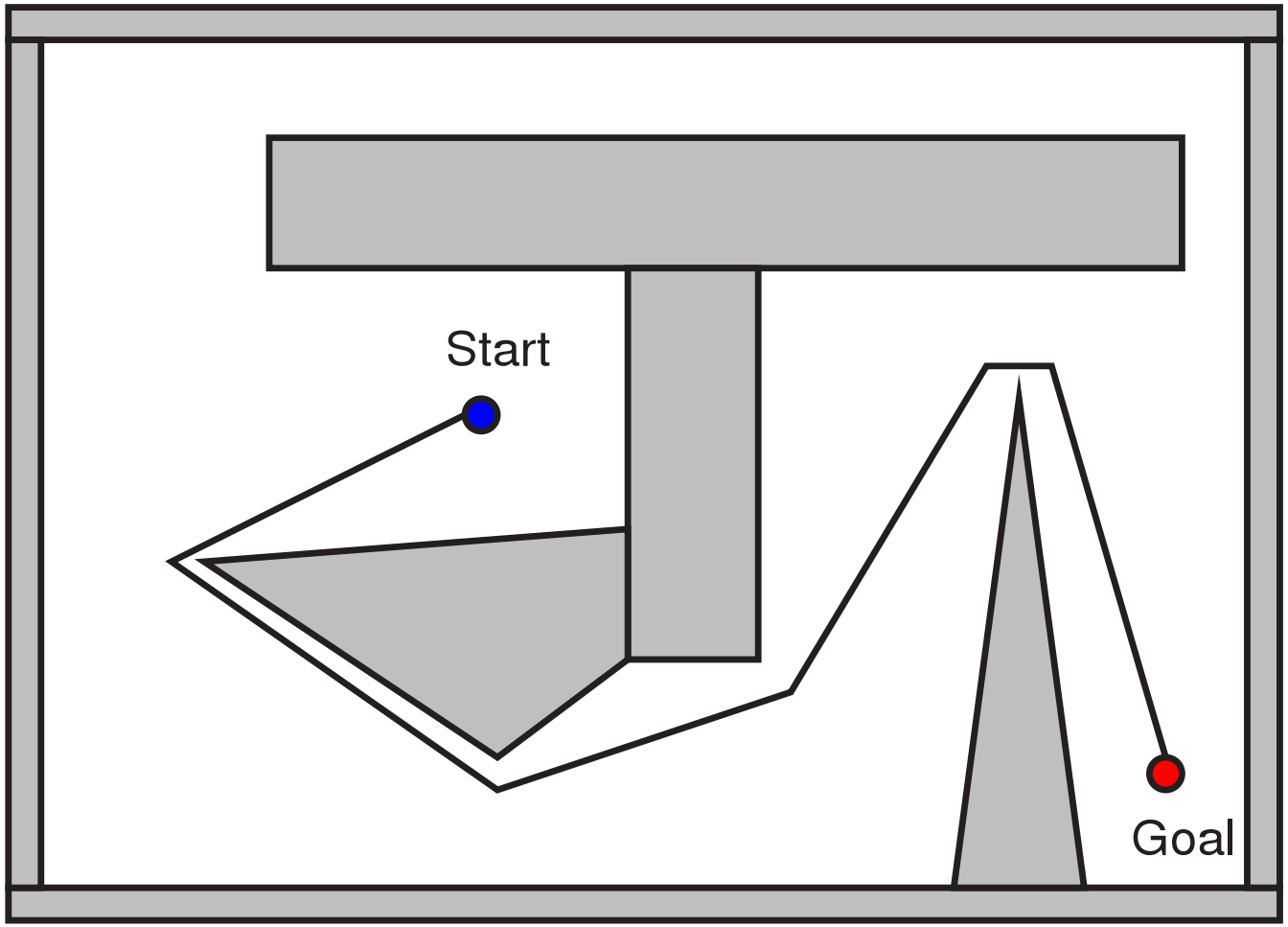
- workspace dimensionality
- number of obstacles and obstacle geometry
- complexity of robot states (configuration space)
- error or uncertainty from environment sensors

Action Models for Robots

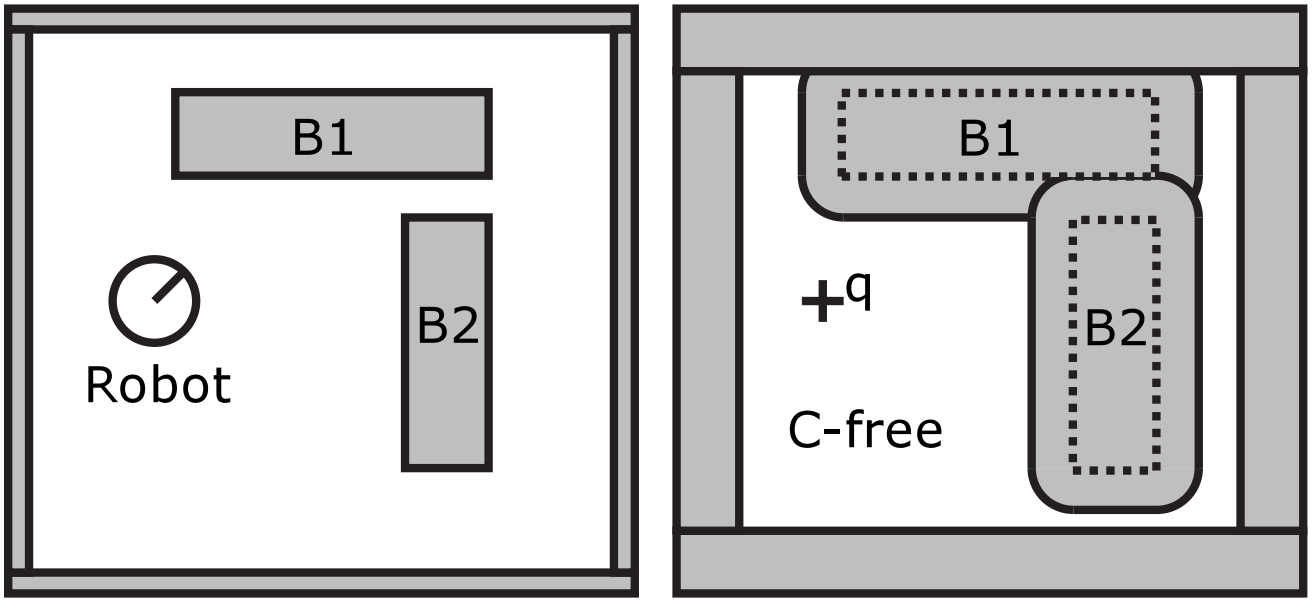
This is conceptually easy, but has many complications:

- Constraints on robot actions
  - non-holonomic motion (Ex: car-like robots)
  - bounded velocity
  - bounded acceleration
- Dynamics at high speeds
- Error or uncertainty in actions

Problem



Definitions



Workspace (W) Configuration Space (C-space)

A vector q including all dof is called a configuration. The set of all q values within range defines a configuration space C.

The Low-Level Planning Problem

Algorithm Input:

- position sensors, localization
- detected obstacles, map
- local/global world models
- goal specification

Algorithm Output:

- Simple path can be represented just by waypoints
- In general, a sequence of actions to execute

Planning Algorithm Criteria

- Sound: Any plan returned is valid
- Complete: If a plan exists, it is found
- Optimal: The plan returned is optimal w.r.t some metric

- Efficient World Updates: Can change world without recomputing everything
- Efficient Query Updates: Can change query without replanning from scratch
- Good dof Scalability: Scales well with increasing C-space dimensions
- Support for holonomic or kinematic constraints

Overview of Approaches

Options to apply discrete domain search in continuous domains:

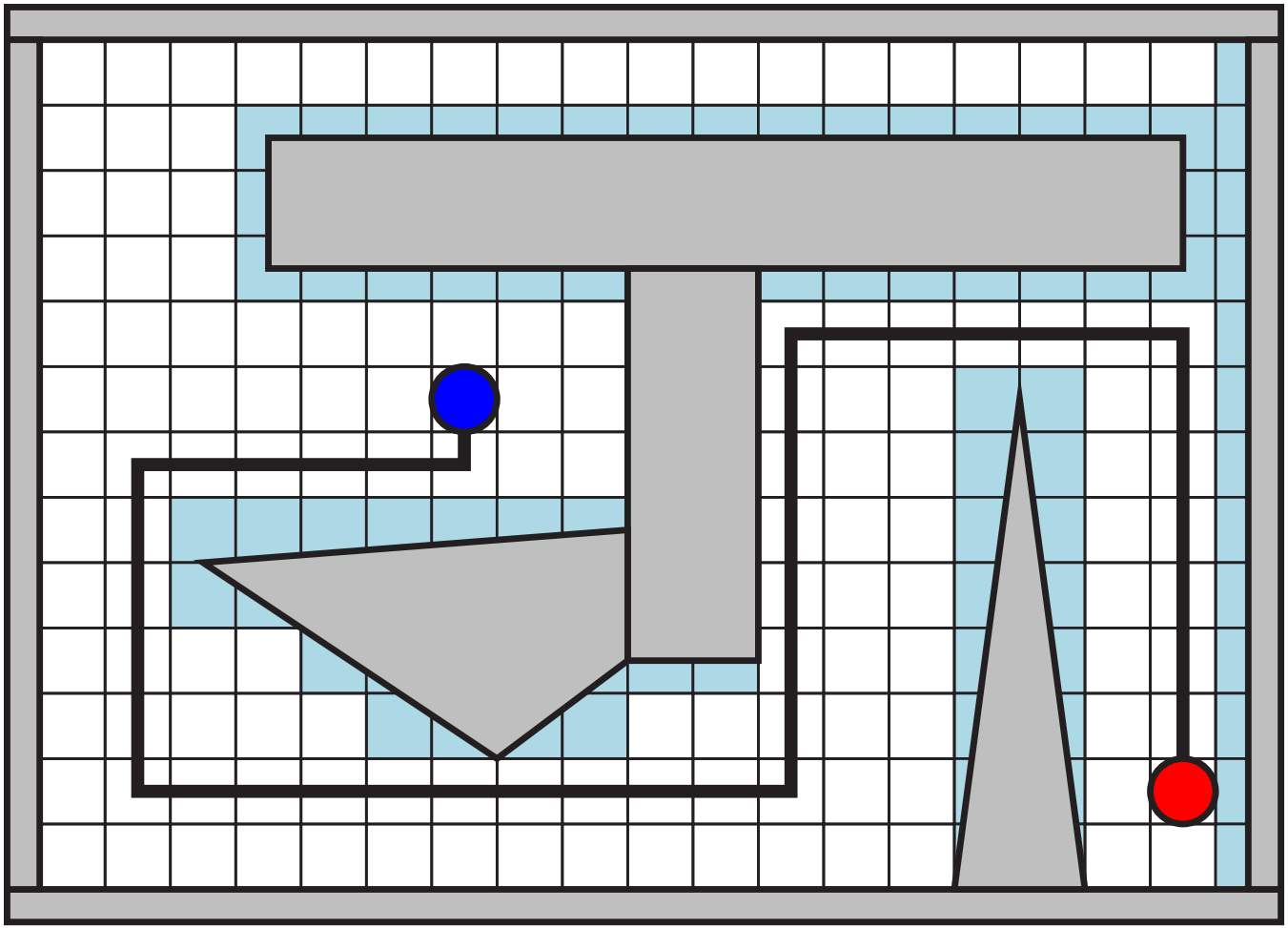
- Tile environment and actions
  - pros: simplest to implement
  - cons: steep speed/resolution tradeoff, discrete motion
- Analyze environment to find optimal decomposition, actions
  - pros: tend to be fast
  - cons: little d.o.f. scalability, numerical/noise problems
- Sample-based Road Map Methods
  - pros: tend to be fast
  - cons: require preprocessing (highly dynamic domains?)

Grid A*, Navigation Functions

- A* is a classical search method, often applied on grids

- Navigation Functions
  - a global local minima-free potential to the goal (i.e. a universal plan)
  - sometimes free space in grid is skeletonized to avoid obstacles more
  - typically use A* or Dijkstra's to fill in the grid or skeleton

Grid Example



Improving on A* Efficiency

Can we make A* deal more efficiently with world updates?

D* (Stentz 1995)

- A* can be modified to propagate cost changes
- maintains optimality of A*
- makes minimal change to universal plan
- a fairly involved implementation however

Other decompositions

Other approximate decompositions:

- Trapezoidal decomposition (2D)
- Quadtree decomposition (2D)
- Octree decomposition (3D)

Exact (but non-optimal) decomposition:

- Vornoi diagram

Optimal Decomposition

With assumptions about the geometry (2D, polygonal obstacles) we can come up with an optimal decomposition.

By optimal we mean it is guaranteed to contain the shortest semi-free path.

Most well known approach is called the Visibility Graph Method.

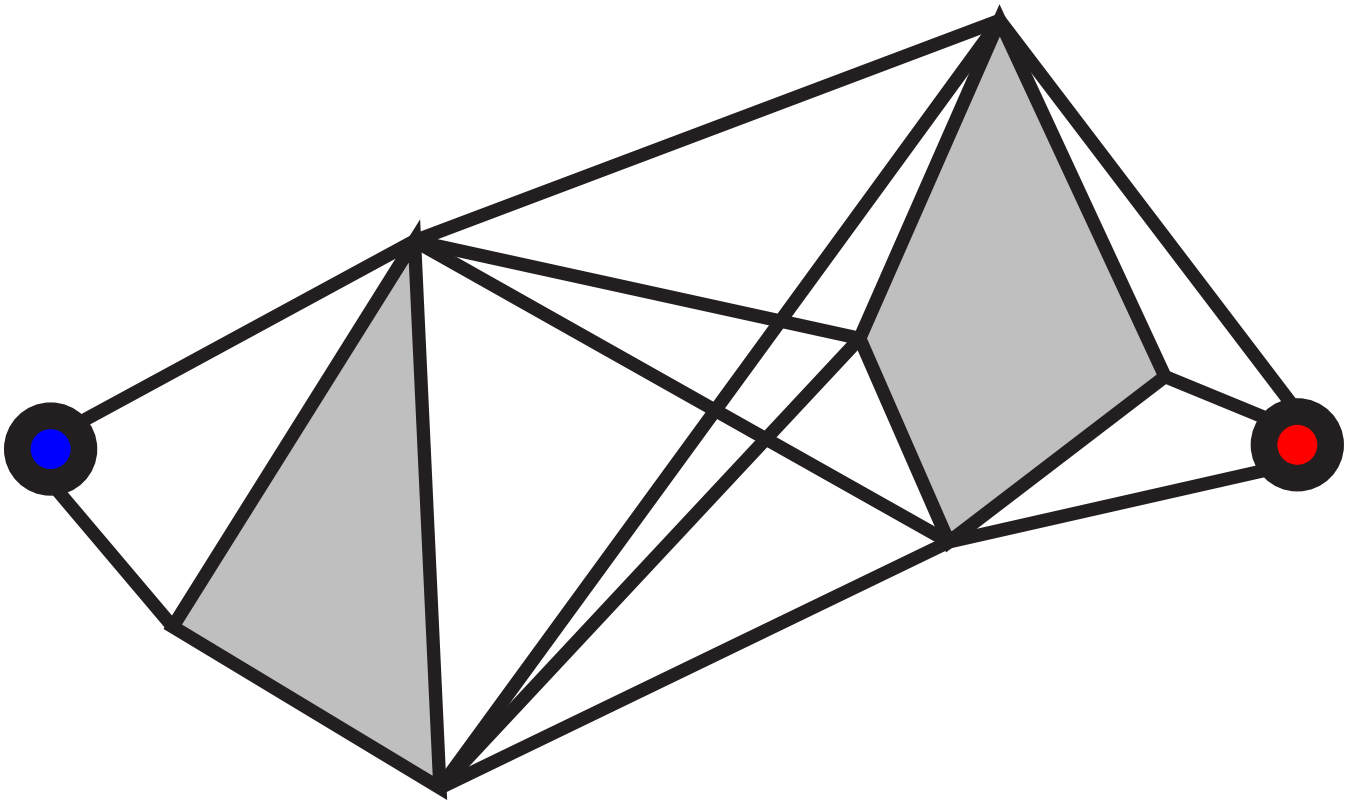
Visibility Graph Method

Key Insight: either go to the left or right of every line segment.

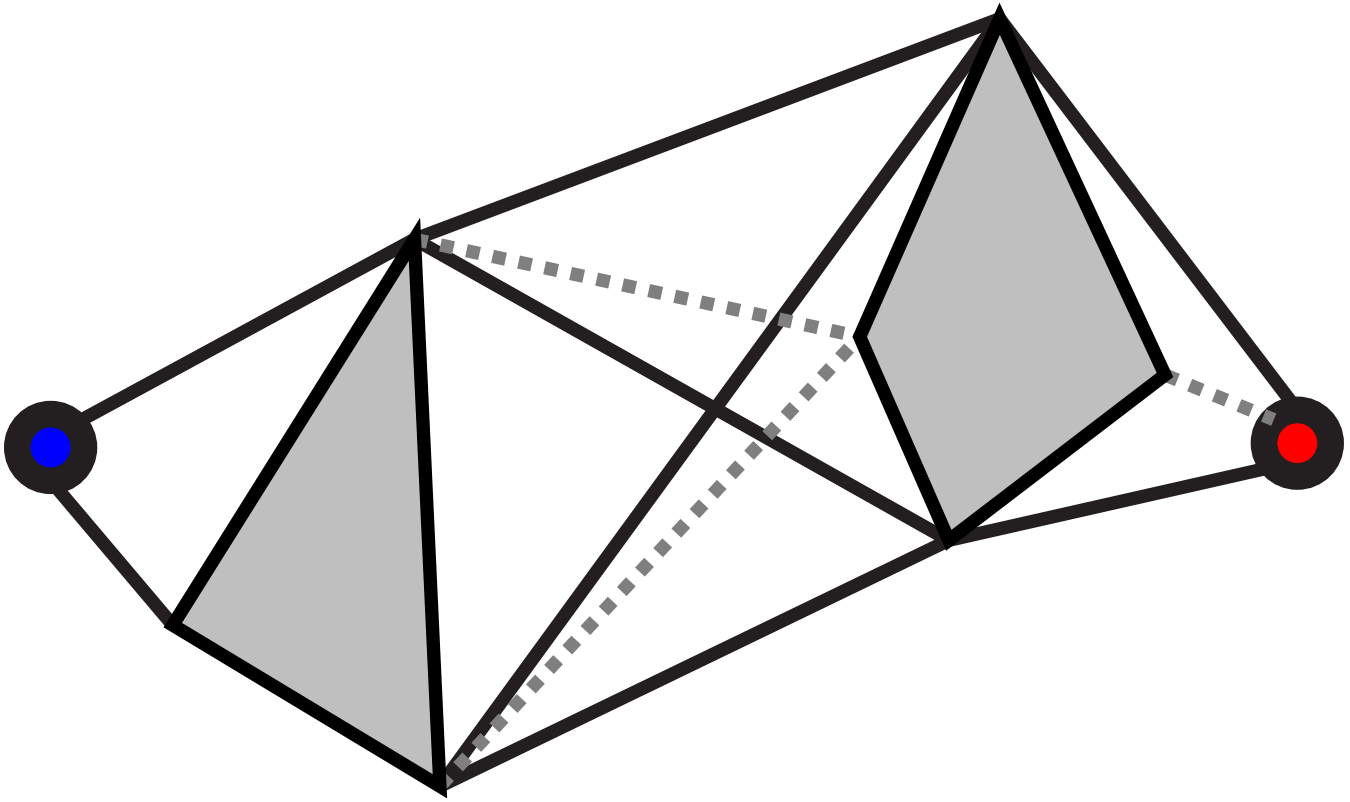
Algorithm:

- For every pair of endpoints, add it to a graph if the line between them does not hit another obstacle.
- Endpoints include every polygon corner, and the initial and goal configurations.

Visibility Graph



Reduced Visibility Graph



Optimization: Between two obstacles, only tangents matter

RRT

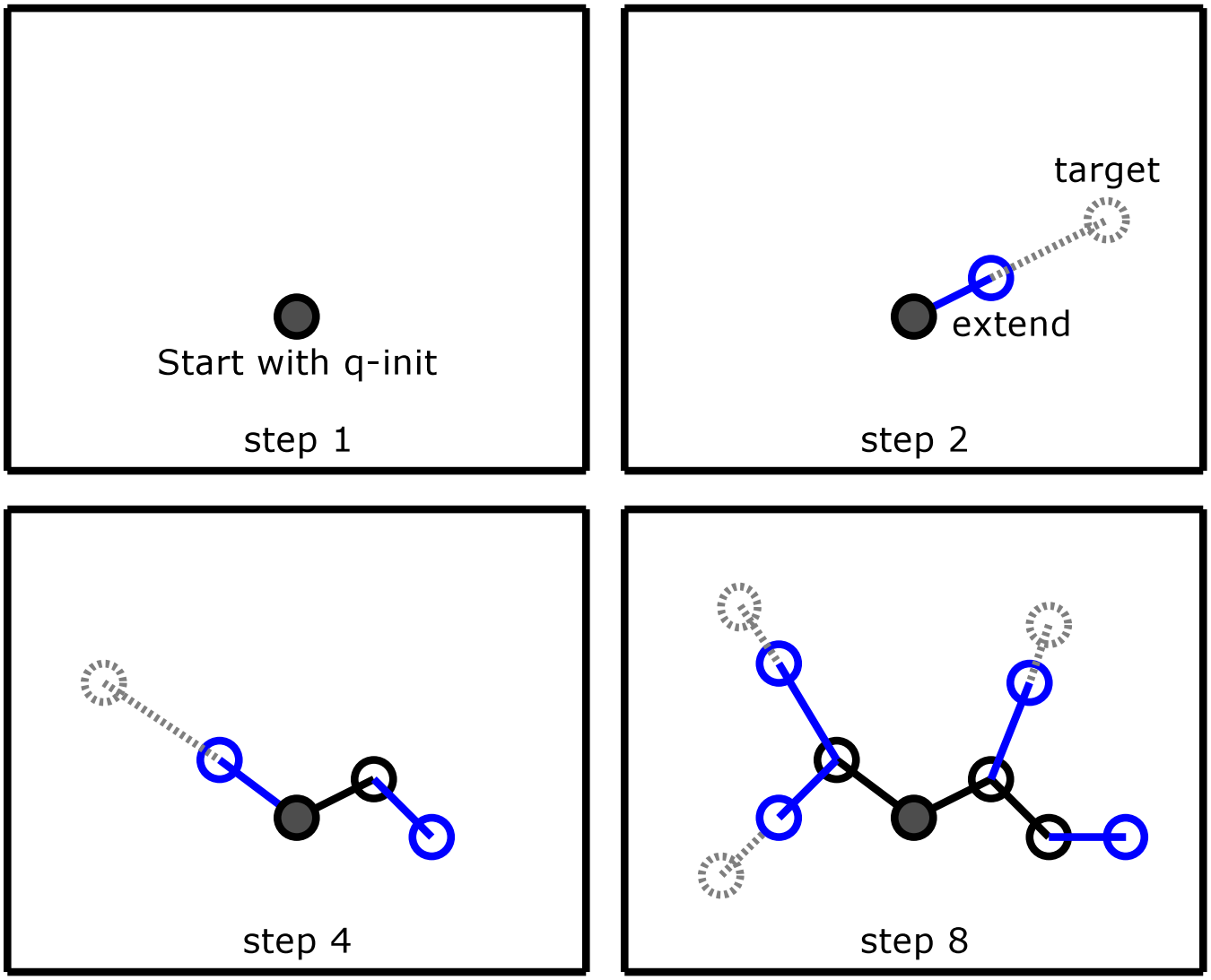
Rapidly Exploring Random Tree (LaValle 1998)

- Explores continuous spaces efficiently
- Can form the basis for a probabilistically complete path planner

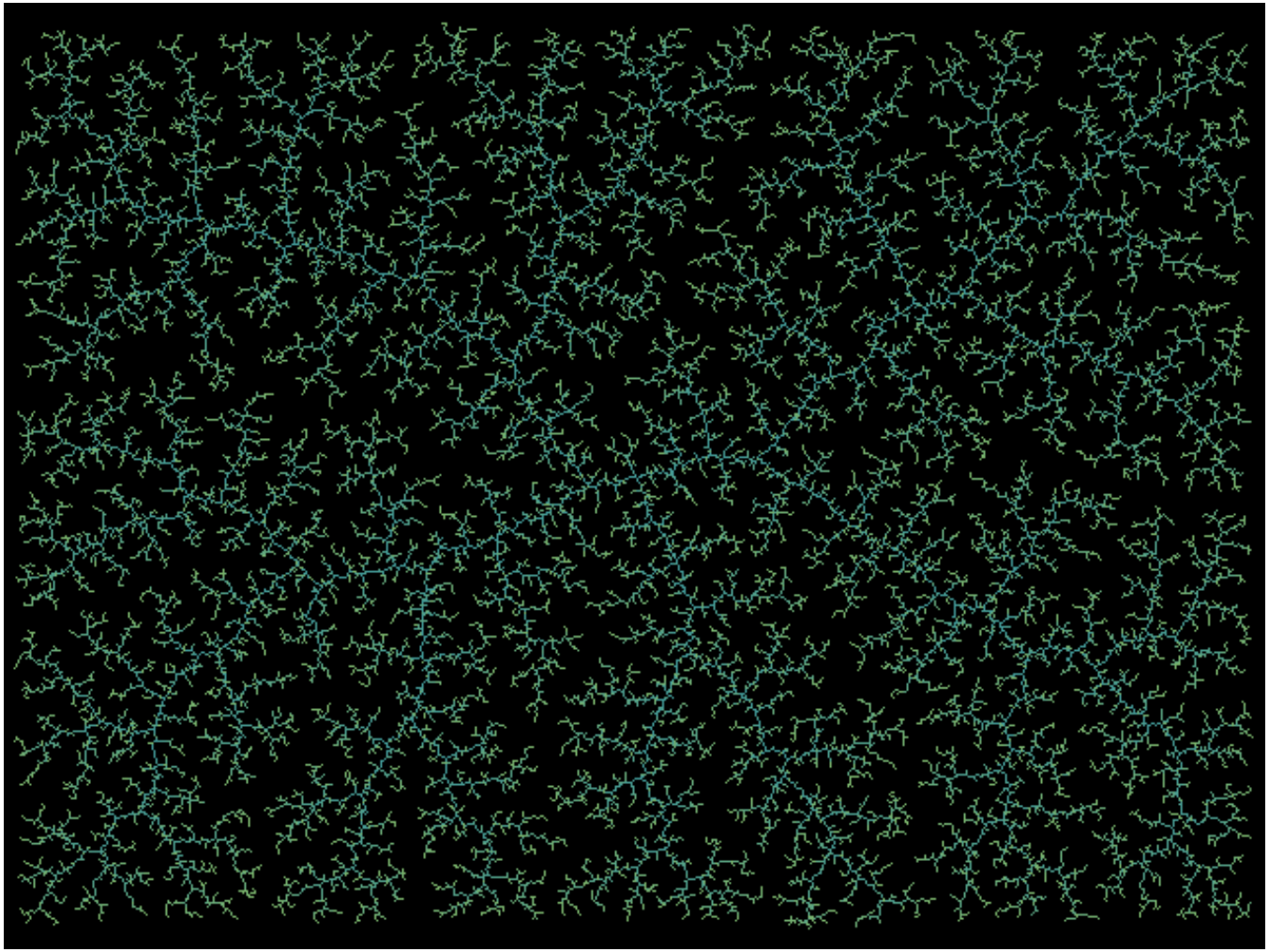
Basic RRT Algorithm:

(1) Initially, start with the initial configuration as root of tree
(2) Pick a random state in the configuration space
(3) Find the closest node in the tree
(4) Extend that node toward the state if possible
(5) Goto (2)

Basic RRT Example



RRT Demo

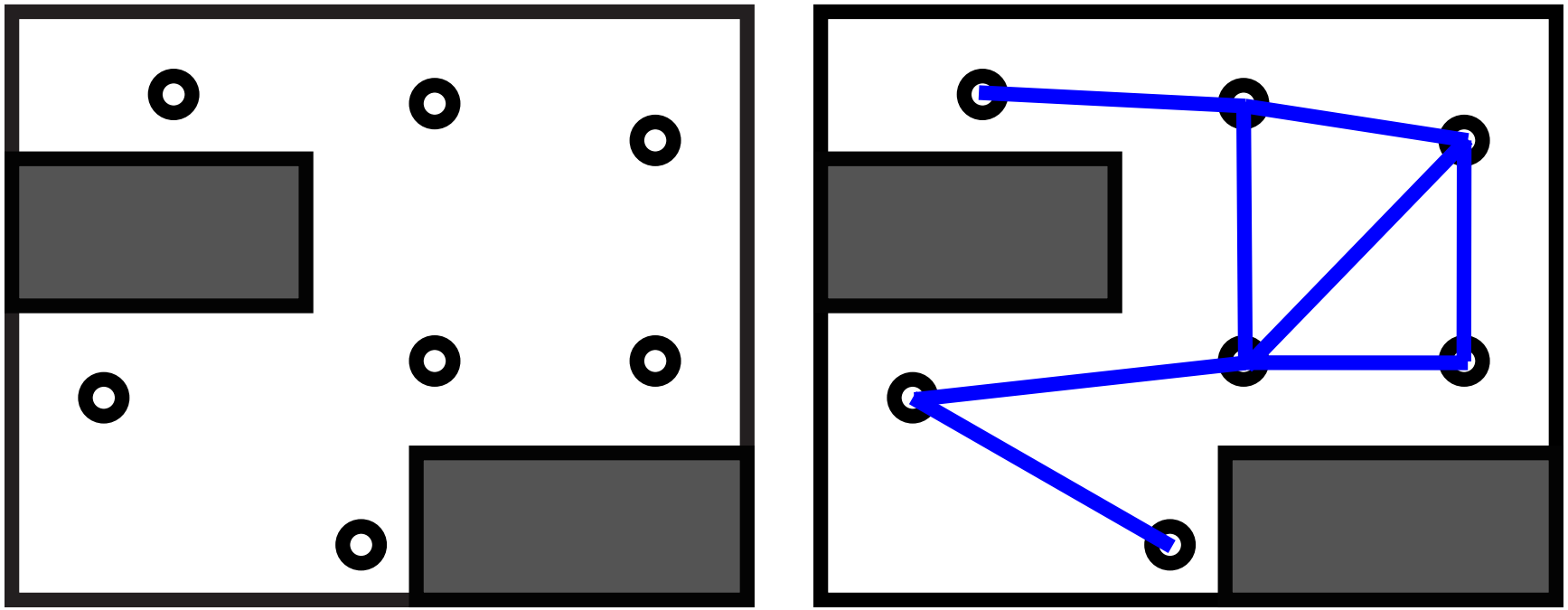


PRM

- Probabilistic Roadmap (Kavraki et. al. 1996)
- Separate planning into two stages
- Learning Phase
  - find random sample of free configurations (vertices)
  - attempt to connect pairs of nearby vertices with a local planner
  - if a valid plan is found, add an edge to the graph
- Query Phase
  - find local connections to graph from initial and goal positions
  - search over roadmap graph using A* to find a plan

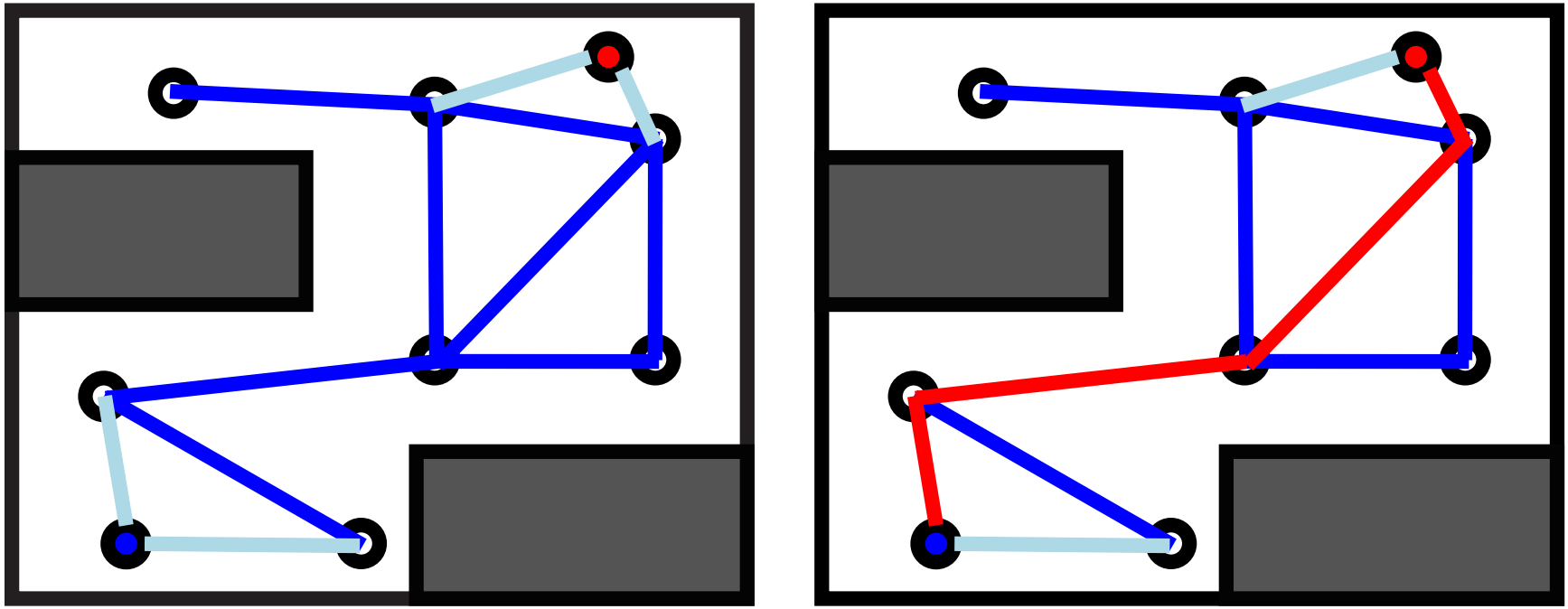
Basic PRM

Learning Phase:



Basic PRM

Query Phase:



Enhancing PRM

Enhancements:

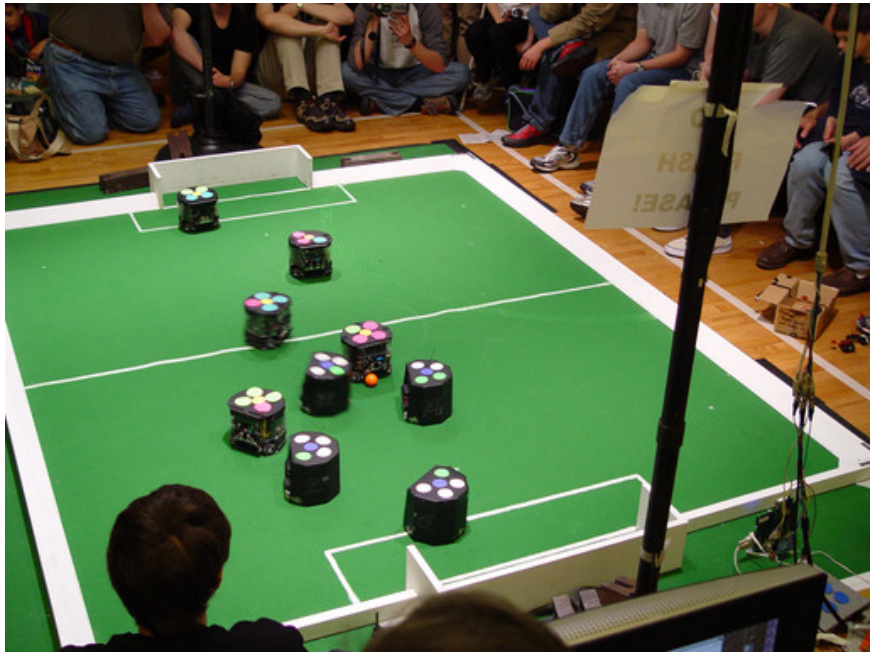
- ObsPRM/Gaussian PRM: sample more nodes near free space boundary
- LazyPRM: don't do collision checks until query stage
  - assume connections are free by default
- PRT: build roadmap from "fat nodes" each of which is an RRT tree

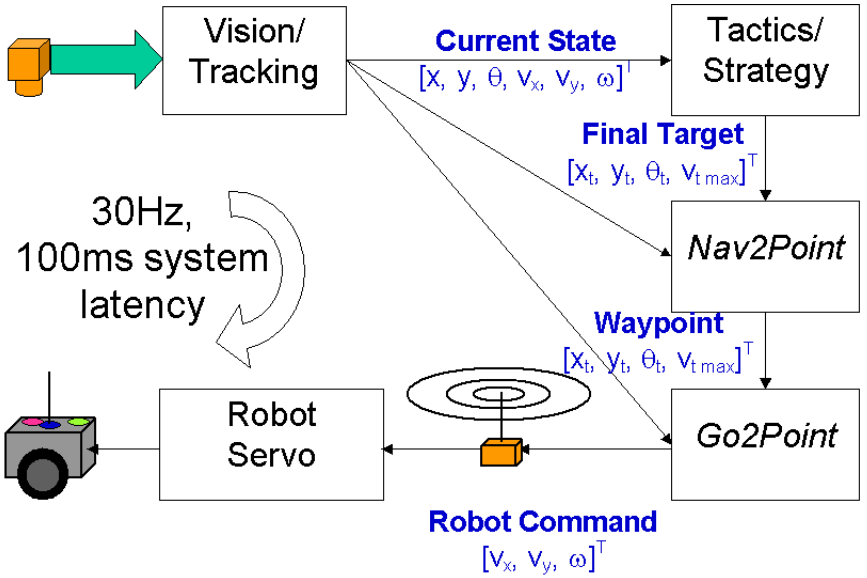
Algorithm Summary

Table 1: Comparison of related planning algorithms

Approach	Complete	Optimal	Efficient World Updates	Efficient Query Updates	Good dof Scalability	Non-Holonomic
Grid A^*	res	grid	no	no	no	no
Grid D^*	res	grid	yes	yes	no	no
Nav Func	res	grid	no	no	no	no
Vis Graph	yes	yes	no	no	no!	no
RRT	prob	no	semi	semi	yes	yes
PRM	prob,res	graph	no	yes	yes	semi
Lazy PRM	prob	graph	yes	yes	yes	no
PRT	prob	no	no	yes	yes	no

RoboCup F180 Platform





RoboCup F180:

- 5 on 5 soccer, overhead vision, centralized control
- implemented realtime RRT-based planner for navigation

ERRT: RRT + Waypoint Cache

Plain RRT planner has little bias toward plan quality, and replanning is naive (all previous information is thrown out before replanning).

Waypoints:

- Previously successful plans can guide new search
- Biases can be encoded by modifying the target point distribution

Waypoint Cache:

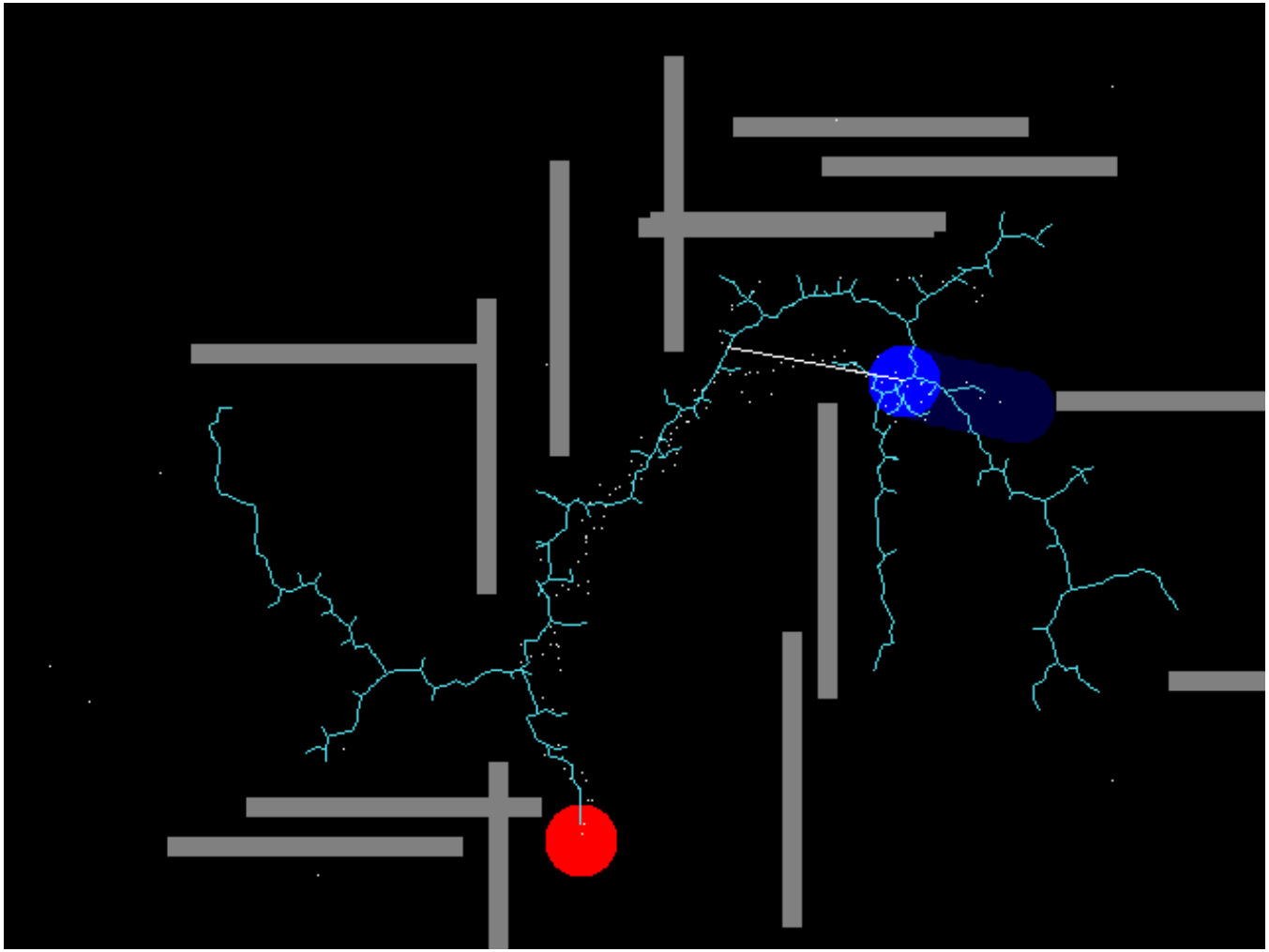
- When a plan is found, store nodes into a bin with random replacement
- During target point selection, choose a random item from waypoint cache with probability q

ERRT Algorithm

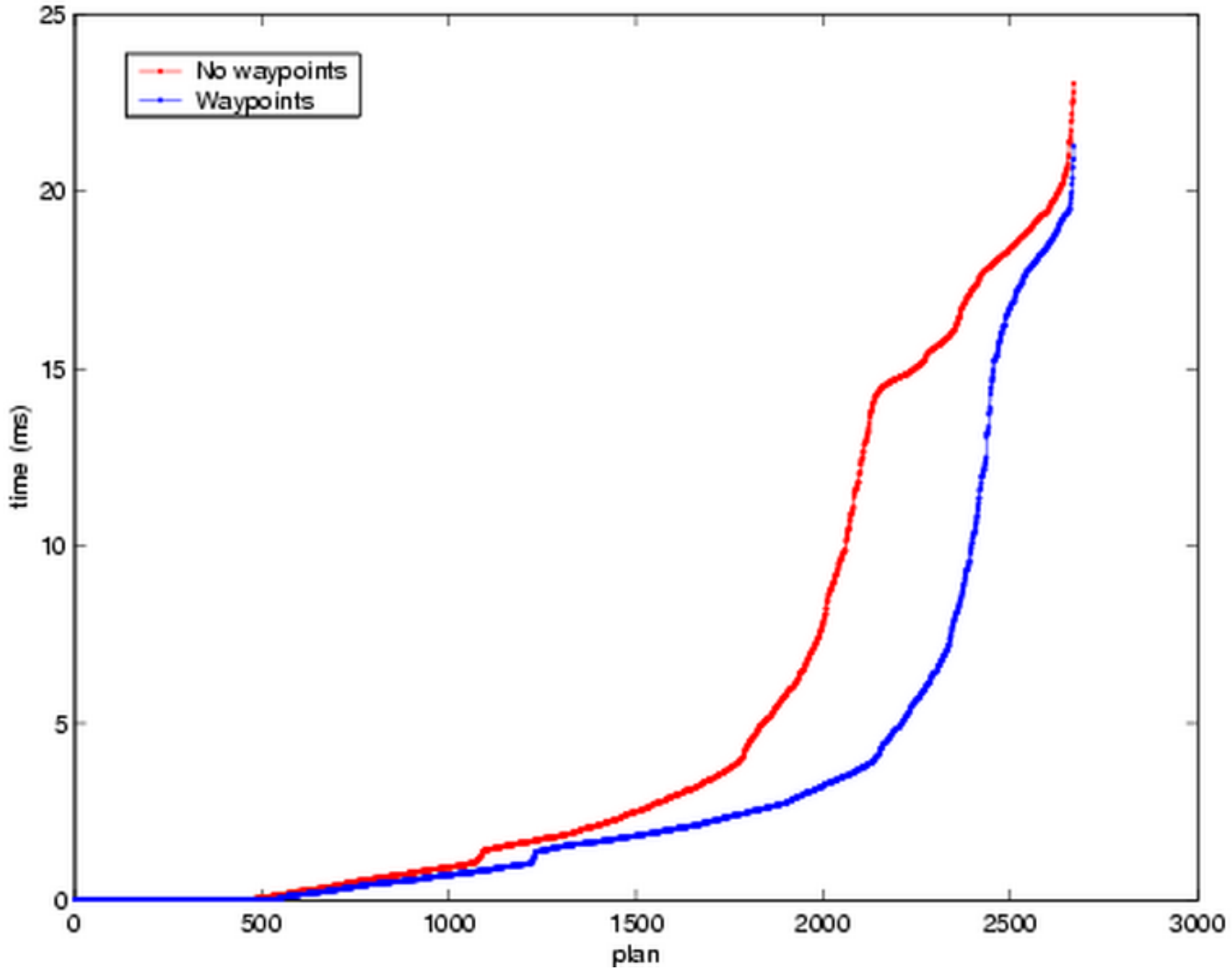
RRT-GoalBias Algorithm:

(1) Initially, start with the initial configuration as root of tree
(2) Pick a target state
  - goal configuration with probability p
  - random state from waypoint cache with probability q
  - random configuration with probability $1 - p - q$
(3) Find the closest node in the tree
(4) Extend that node toward the state if possible
(5) Goto (2)

ERRT



Waypoints [Results]



*RRT Limitations

Limitations of current RRT variants:

- Cannot represent costs very well
- Hard to characterize which path is found
- Fast, but we still throw out a lot of information

PRM Revisited

- PRMs are nice because they build a representation of free space
- This can be used later for more efficient queries than ERRT
- But we have to be able to deal with a changing world
- For efficiency we can rely on the world not changing too quickly

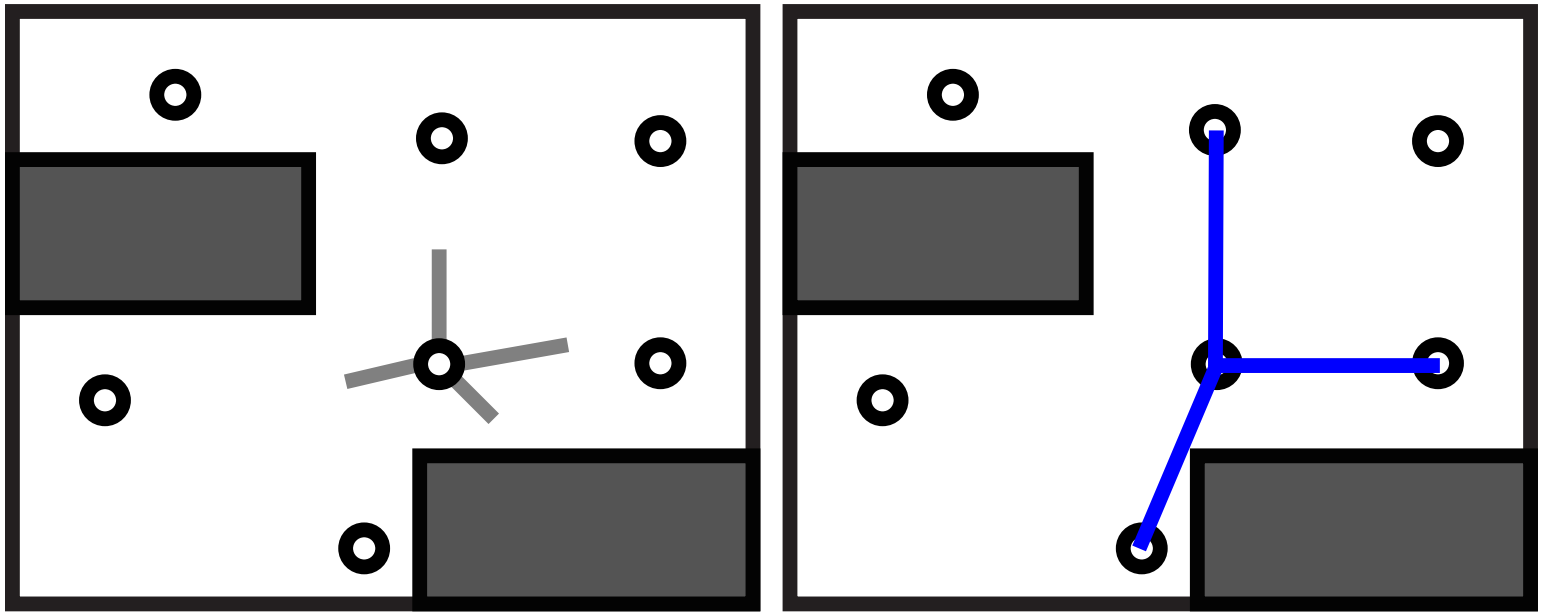
PRM's Critical Questions

- How to sample for free locations to create nodes
- Which nodes to try to connect to each other
- Simplest PRM is uniform, and fully connected
- This has n^2 complexity for n nodes, however

Choices for parameters:

- Sampling is uniform, but will not converge that way
- Connection: choose nearby positions based on random distribution
- Find nearest node to randomly chosen one and try to connect to it
- This is related to sampling and extension in RRT

PRM Connection Generation



PRM in Dynamic Environments

- Need to be able to modify existing roadmap
- First, remove any nodes or edges that have become invalid
- Also remove some fraction of existing states randomly
- Add new states, and try to connect to graph as described

Replacing Nodes:

- Each node has a replacement probability
  - new nodes start with a value based on a prior distribution
  - over time the probability is increased (aging)
  - set probability to a low value if node is used in a plan
- Using non-uniform replacement we can achieve non-uniform roadmaps

Dynamic PRM: Demo

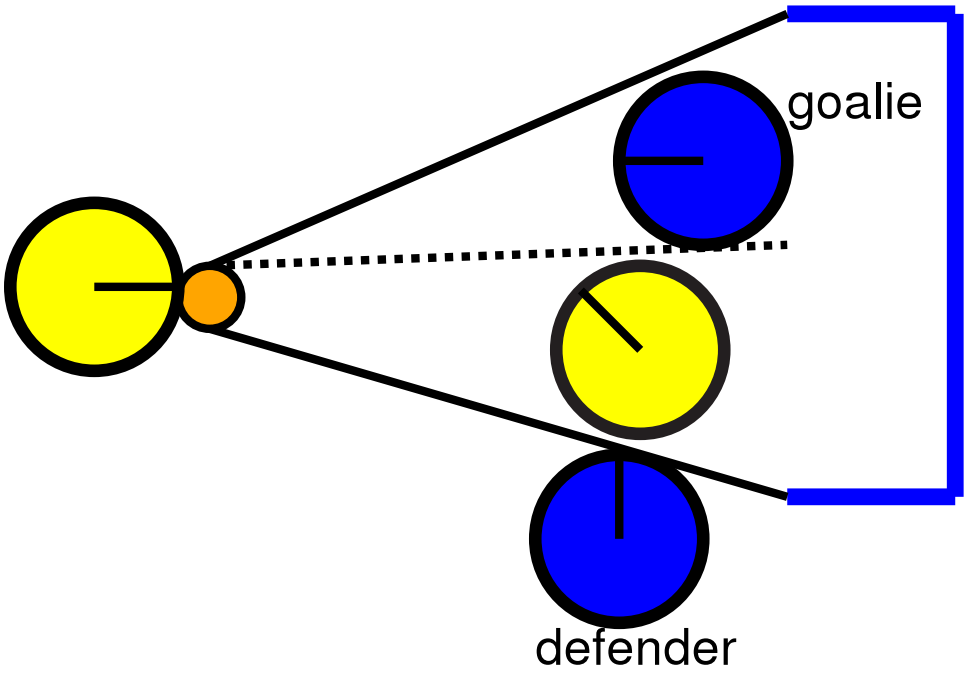
Extended Algorithm Summary

Table 2: Comparison of related planning algorithms

Approach	Complete	Optimal	Efficient World Updates	Efficient Query Updates	Good dof Scalability	Non-Holonomic
Grid A^*	res	grid	no	no	no	no
Grid D^*	res	grid	yes	yes	no	no
Nav Func	res	grid	no	no	no	no
Vis Graph	yes	yes	no	no	no!	no
RRT	prob	no	semi	semi	yes	yes
PRM	prob,res	graph	no	yes	yes	semi
Lazy PRM	prob	graph	yes	yes	yes	no
PRT	prob	no	no	yes	yes	no
ERRT	prob	no	semi	semi	?	no
Dynamic PRM	prob	graph	yes	yes	?	no

Goals as Objective Functions

- Represent goals as objective function rather than a point
- What if the goal point cannot be reached
- Can represent goals that are arbitrary sets of configurations
- Method for ranking alternatives



Goals as Objective Functions Demo

Open Areas of Research

My take on interesting active aread of research in motion planning:

- more accurate modelling of robot dynamics
  - acceleration and velocity bounds are either pessimistic or inaccurate
  - action models containing uncertainty
  - faster planning for complex robots on a terrain
- online learning of optimized planner parameters
  - modern planners have far too many parameters requiring user adjustment
- biped step planning
  - balance and foot placement constaints are complex
  - must be realtime or you will fall over