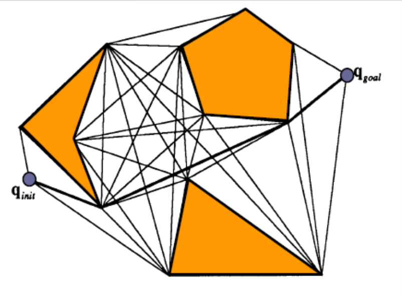
Part 1 - Motion Planning

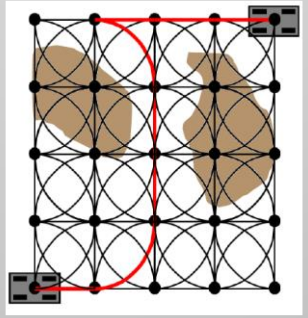
Motion Planning Algorithms

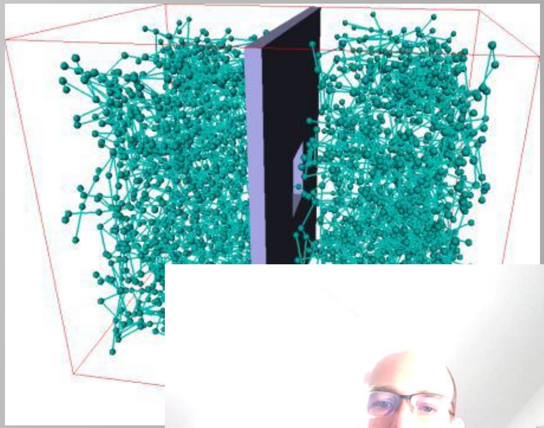
Overview

- Roadmap methods
  - Visibility graphs
  - Voronoi diagrams
- Decomposition methods
  - Approximate
  - Exact
- Potential fields
- Sampling-based Planning
  - PRM
  - RRT
- Applications to Computer Animation
- Dynamic environments
- Crowd Simulation



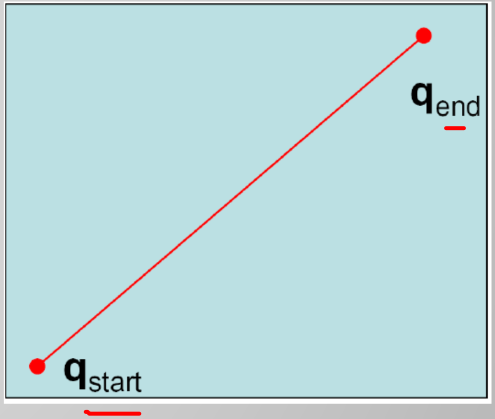
- Deterministic, heuristic or probabilistic
- Planning algorithms evaluation criteria:
  - Completeness
  - Optimality
  - Speed
  - Generality





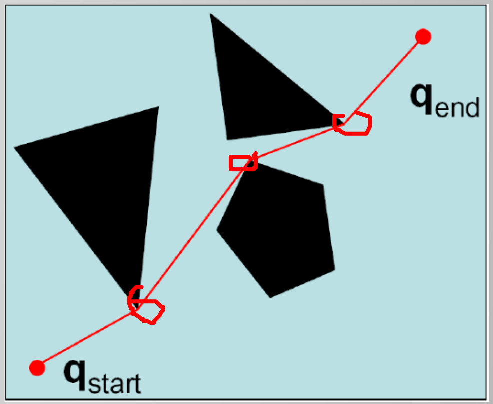
VISIBILITY GRAPHS

IN THE ABSENCE OF OBSTACLES, THE BEST PATH IS THE STRAIGHT LINE BETWEEN Q_{START} AND Q_{END}



Visibility graphs

Assuming polygonal obstacles:
it looks like the shortest path is a sequence of straight lines joining the vertices of the obstacles

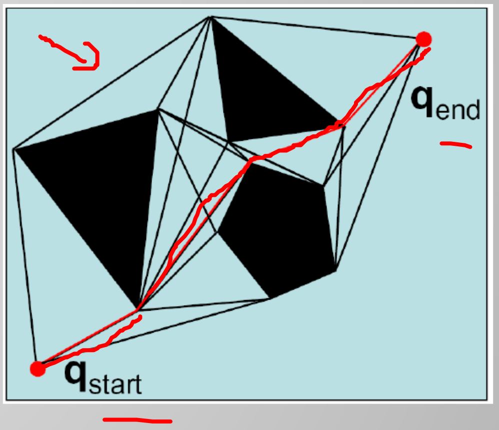


VISIBILITY GRAPHS

VISIBILITY GRAPH G
= SET OF UNBLOCKED LINES BETWEEN THE VERTICES OF THE OBSTACLES + Q_{START} AND Q_{GOAL}

A NODE P IS LINKED TO A NODE P’ IF P’ VISIBLE FROM P

SOLUTION = SHORTEST PATH IN THE VISIBILITY GRAPH



Visibility graphs

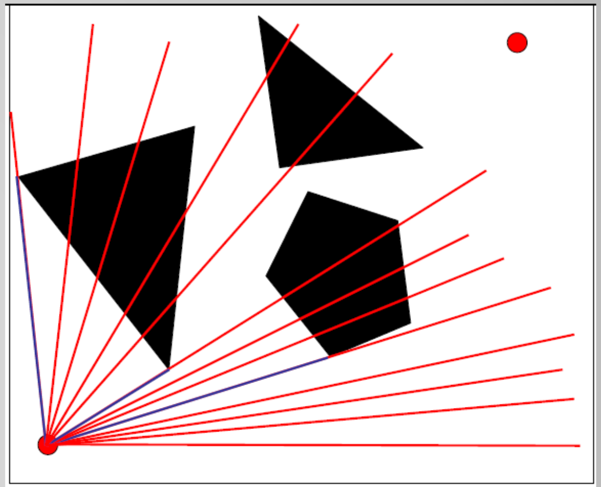
Construction: sweep algorithm

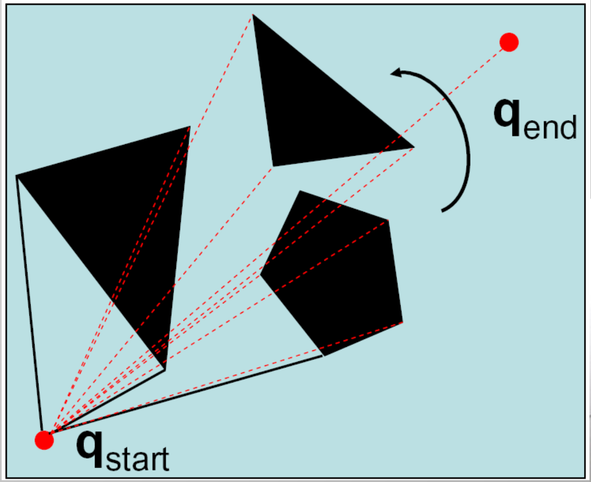
Sweep a line originating at each vertex

Record those lines that end at visible vertices

Complexity

- N = total number of vertices of the obstacle polygons
- Naïve: $O(N^3)$
- Sweep: $O(N^2 \log N)$



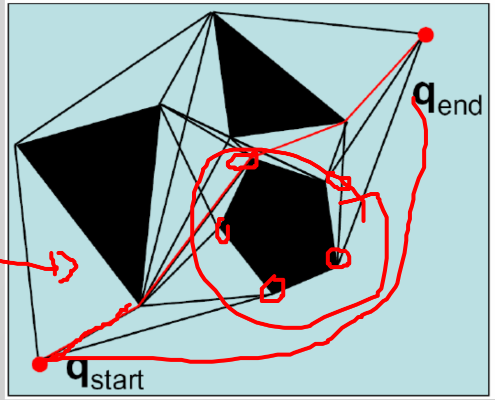


Visibility graphs

Shortest path but:

- Tries to stay as close as possible to obstacles
- Any execution error will lead to a collision
- Complicated in >> 2 dimensions

We may not care about strict optimality so long as we find a safe path. Staying away from obstacles is more important than finding the shortest path



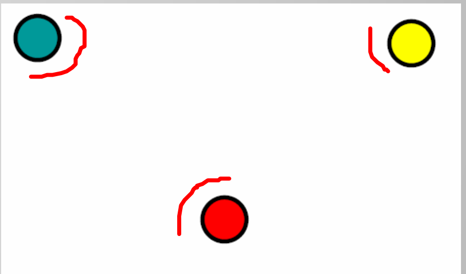
Need to define other types of "roadmaps"

Lozano-Pérez, Tomás; Wesley, Michael A. (1979), "An algorithm for planning collision-free paths amon obstacles", Communications of the ACM, ***22*** *(10): 560–570, doi:10.1145/359156.359164*

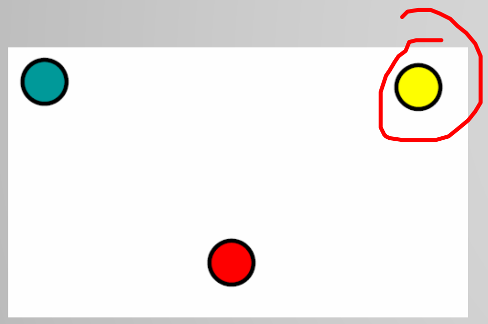
Voronoi Diagram

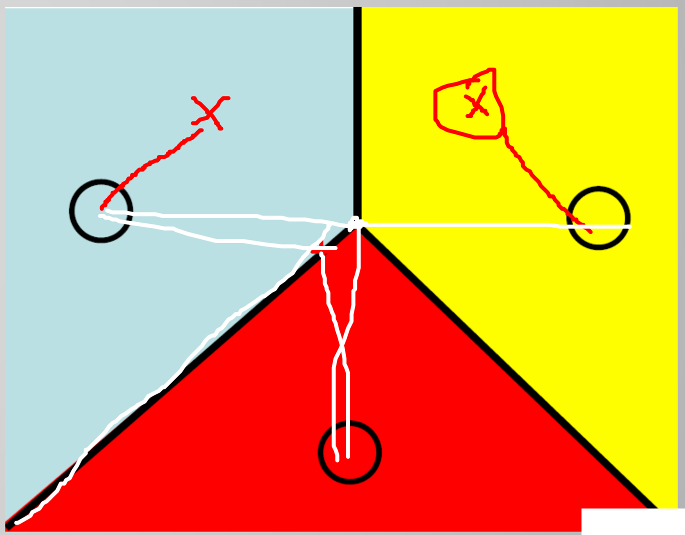
Given a set of data points in the plane:

- Color the entire plane such that the color of any point in the plane is the same as the color of its nearest



VORONOI DIAGRAM



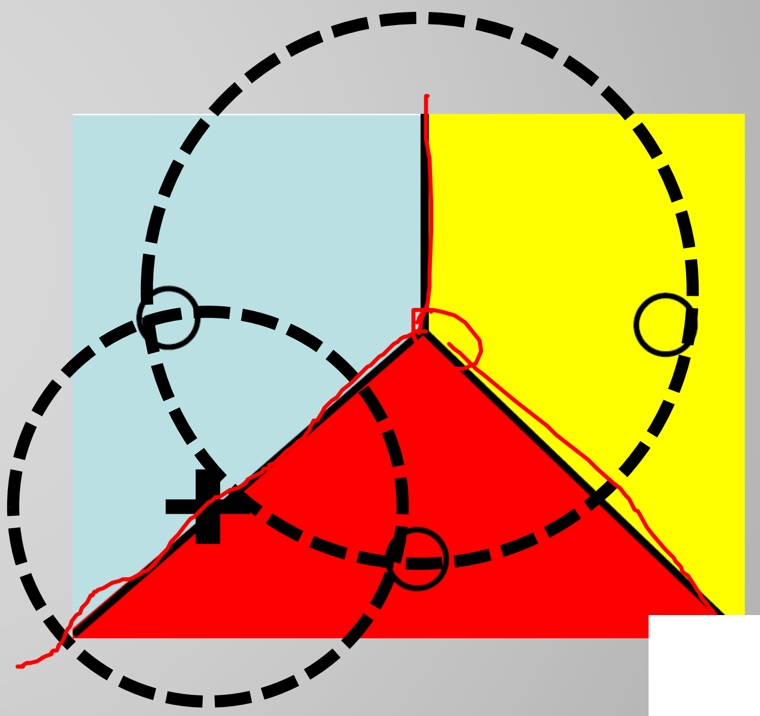


Voronoi Diagram

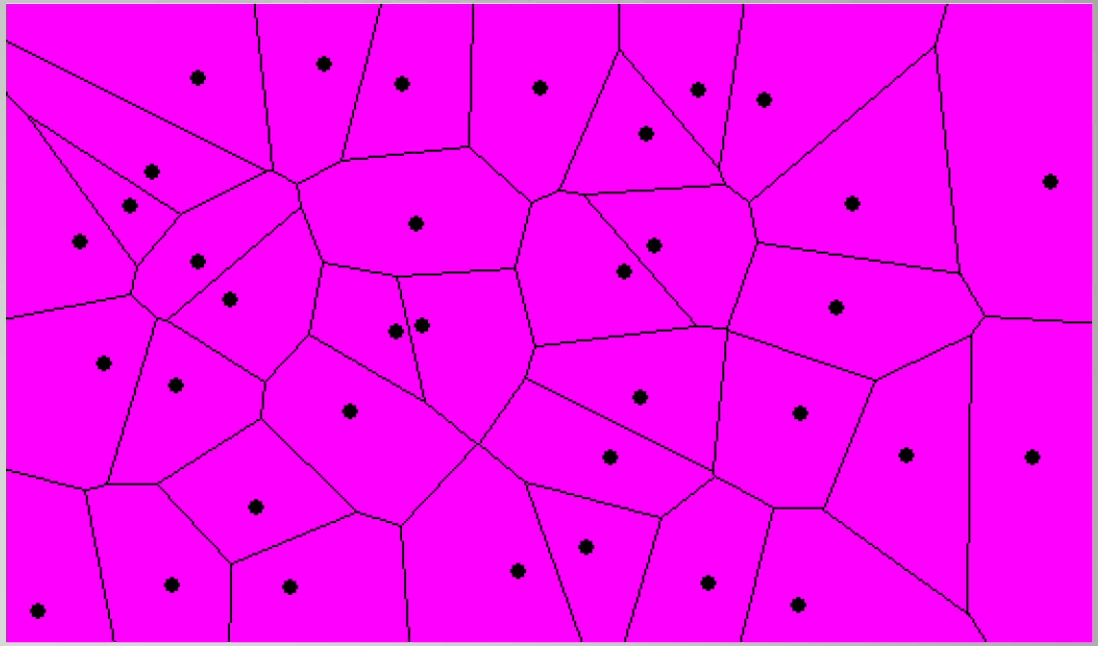
Voronoi diagram:
The set of line segments separating the regions corresponding to different colors

Line segment = points equidistant from 2 data points

Vertices = points equidistant from > 2 data points



VORONOI

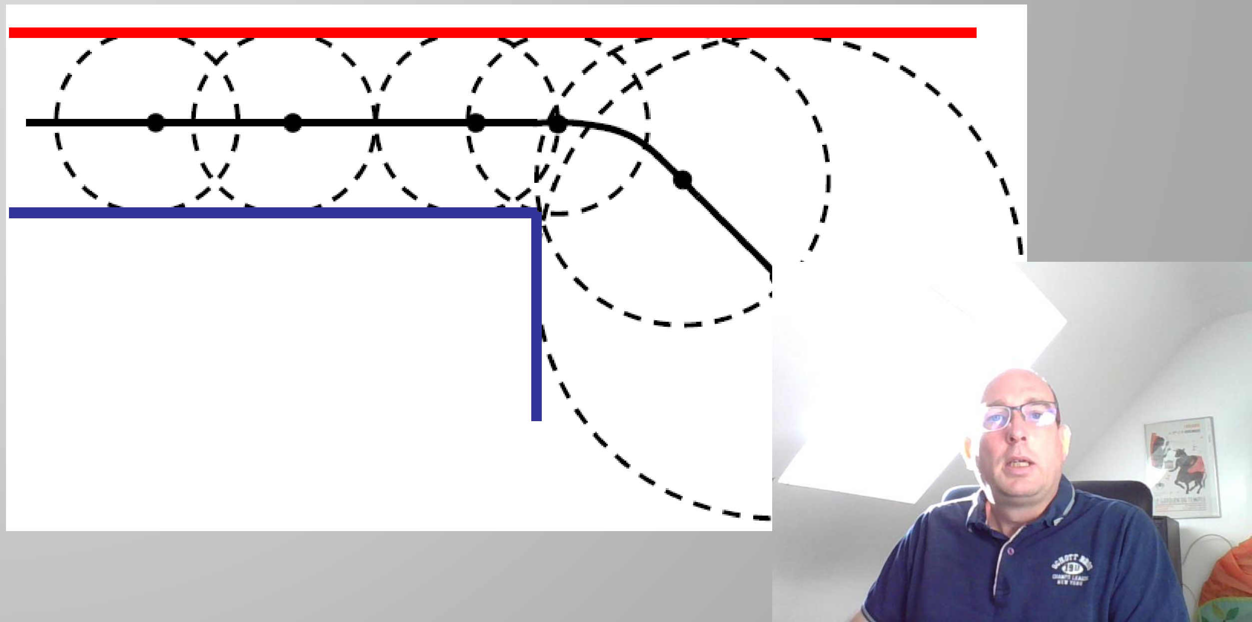


COMPLEXITY (IN THE PLANE):

- O(*N* log *N*) time
- O(*N*) space

BEYOND POINTS:

- Edges are combinations of straight line segments and segments of quadratic curves
- Straight edges: Points equidistant from 2 lines
- Curved edges: Points equidistant from one corner and one line



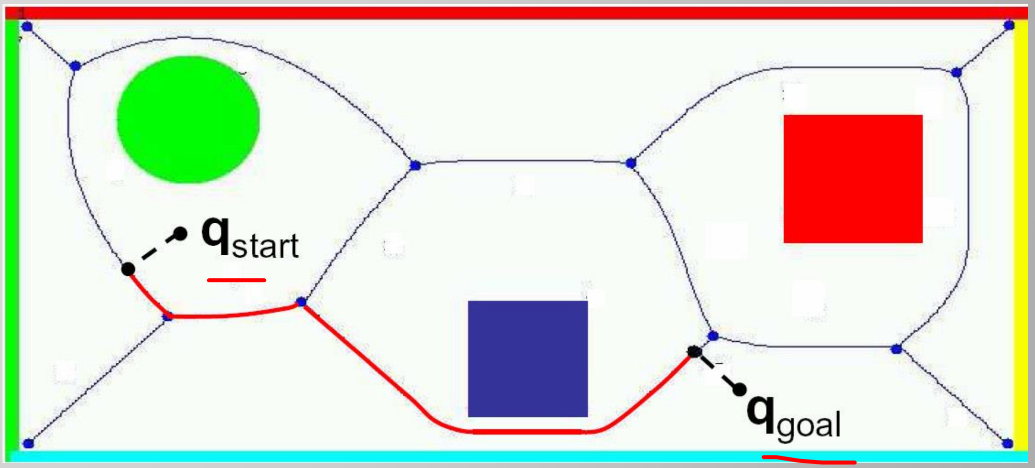
Voronoi Diagram

Key property:

The points on the edges of the Voronoi diagram are the *furthest* from the obstacles

Idea:

Construct a path between q_{START} and q_{GOAL} by following edges on the Voronoi diagram (Use the Voronoi diagram as a roadmap graph instead of the visibility graph)



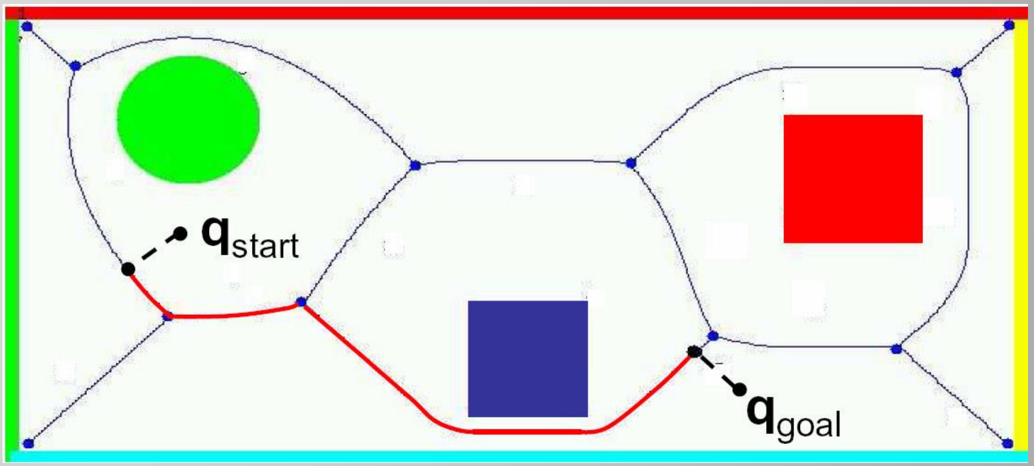
Voronoi Diagram

Difficult to compute in higher dimensions or nonpolygonal worlds

- Approximate algorithms exist
- Use of Voronoi is not necessarily the best

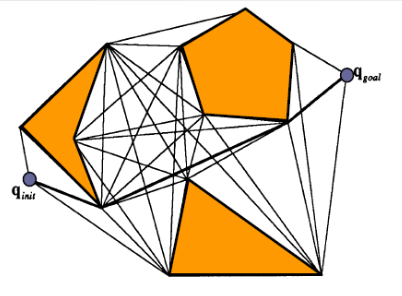
Heuristic ("stay away from obstacles") can lead to paths that are much too conservative

Can be unstable: small changes in obstacle configuration can lead to large changes in the diagram

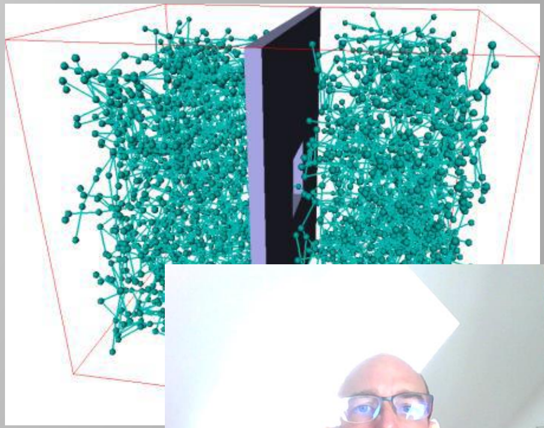


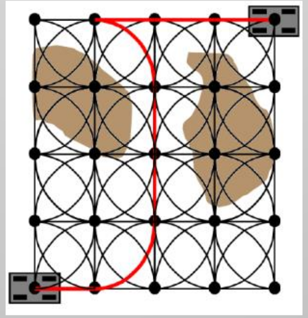
Overview

- Roadmap methods
  - Visibility graphs
  - Voronoi diagrams
- Decomposition methods
  - Approximate
  - Exact
- Potential fields
- Sampling-based Planning
  - PRM
  - RRT
- Applications to Computer Animation
- Dynamic environments
- Crowd Simulation



- Deterministic, heuristic or probabilistic
- Planning algorithms evaluation criteria:
  - Completeness
  - Optimality
  - Speed
  - Generality



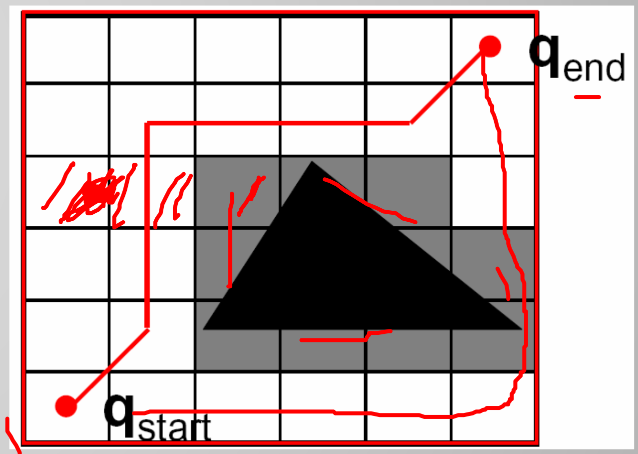


APPROXIMATE CELL DECOMPOSITION

DEFINE A DISCRETE GRID IN C_{SPACE}

- Mark any cell of the grid that intersects C_{obs} as blocked

FIND PATH THROUGH REMAINING CELLS BY USING (FOR EXAMPLE) A* (E.G., USE EUCLIDEAN DISTANCE AS HEURISTIC)



CANNOT BE *COMPLETE* AS DESCRIBED SO FAR. WHY?

NOTE:

- deterministic approaches with exact decomposition are complete
- deterministic approaches with approximate decomposition are representation-complete

Approximate cell decomposition

Cannot find a path in this case even though one exists

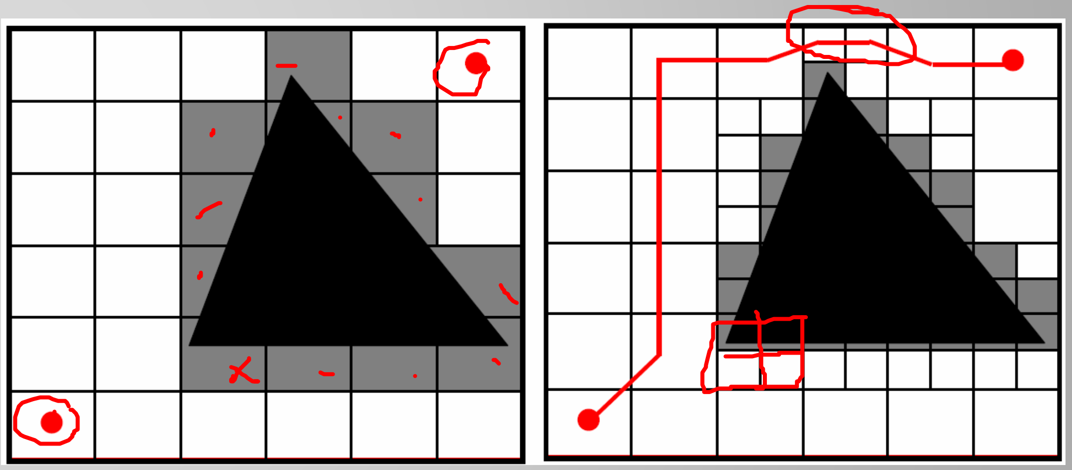
Solution: distinguish between

- Cells that are entirely contained in Cobs (*FULL*) and
- Cells that partially intersect C_{obs} (*MIXED*)

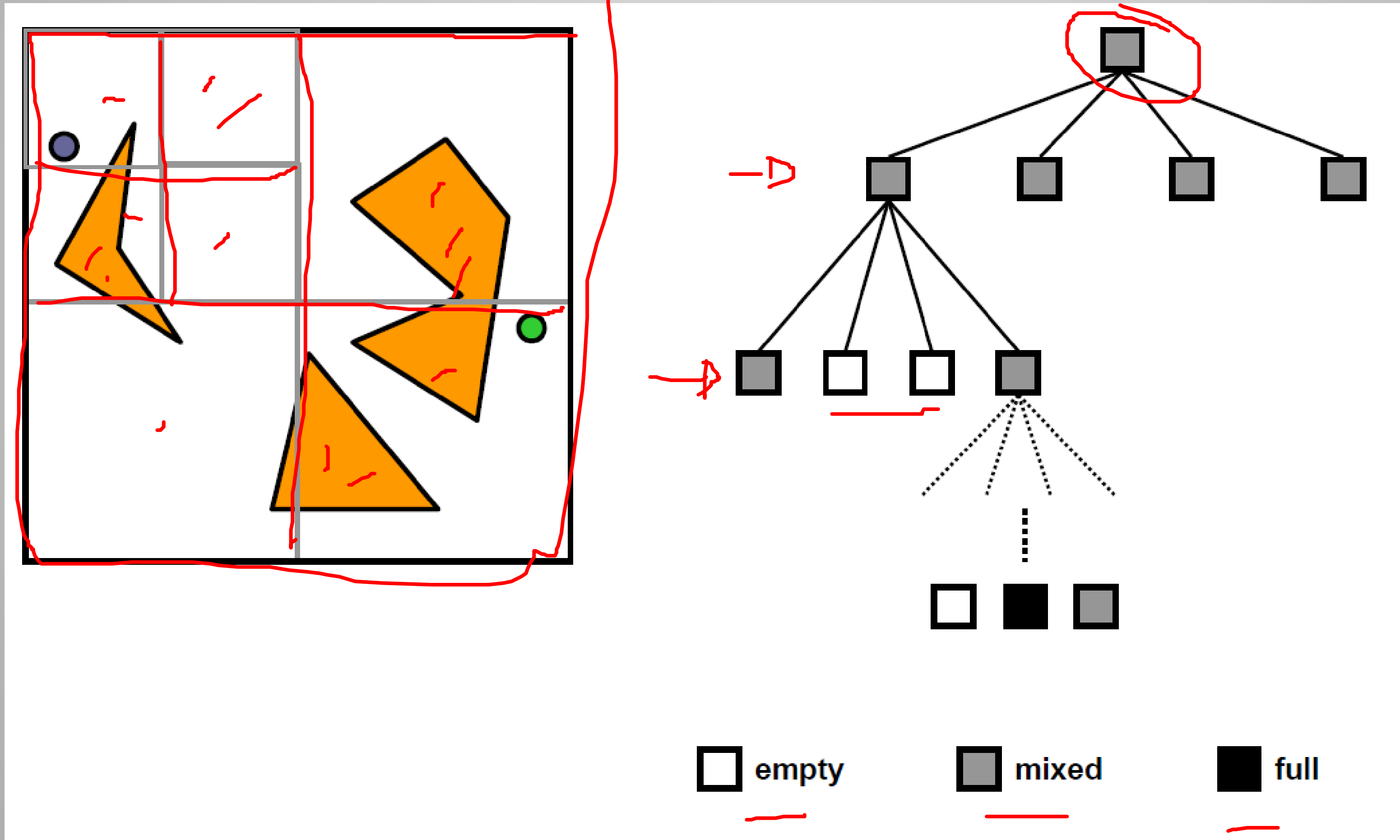
Try to find a path using the current set of cells

If no path found:

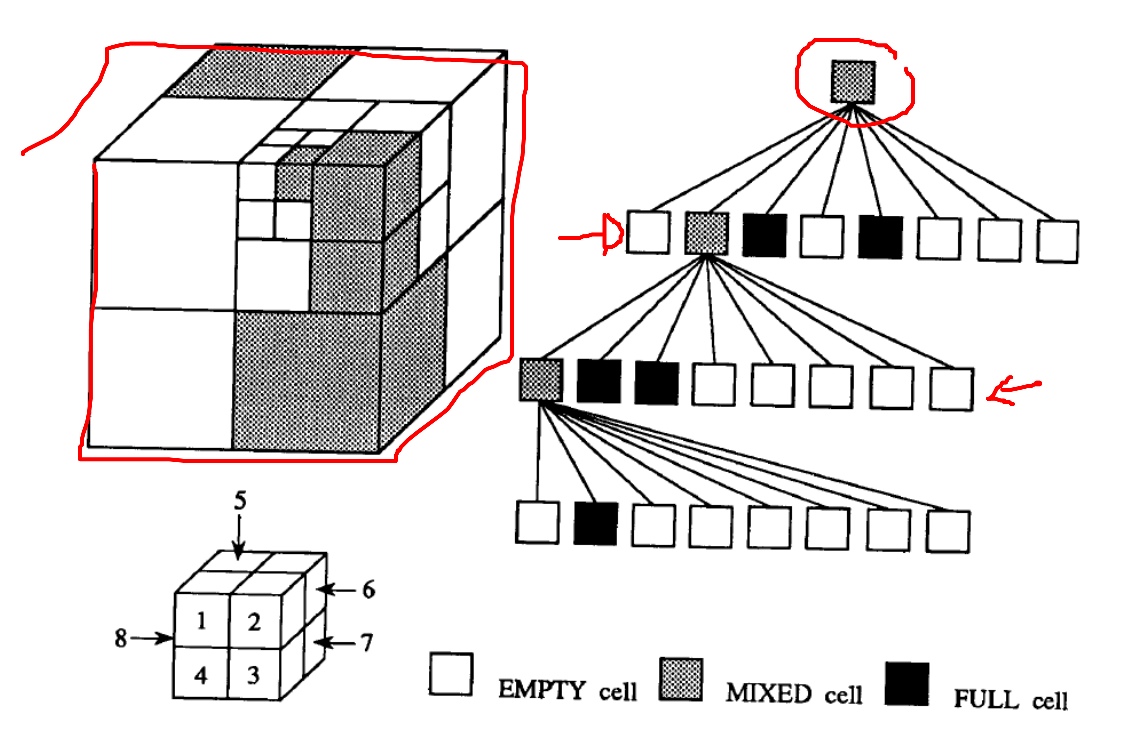
- Subdivide the *MIXED* cells and try again with the new set of cells



QUADTREE DECOMPOSITION



Octree decomposition



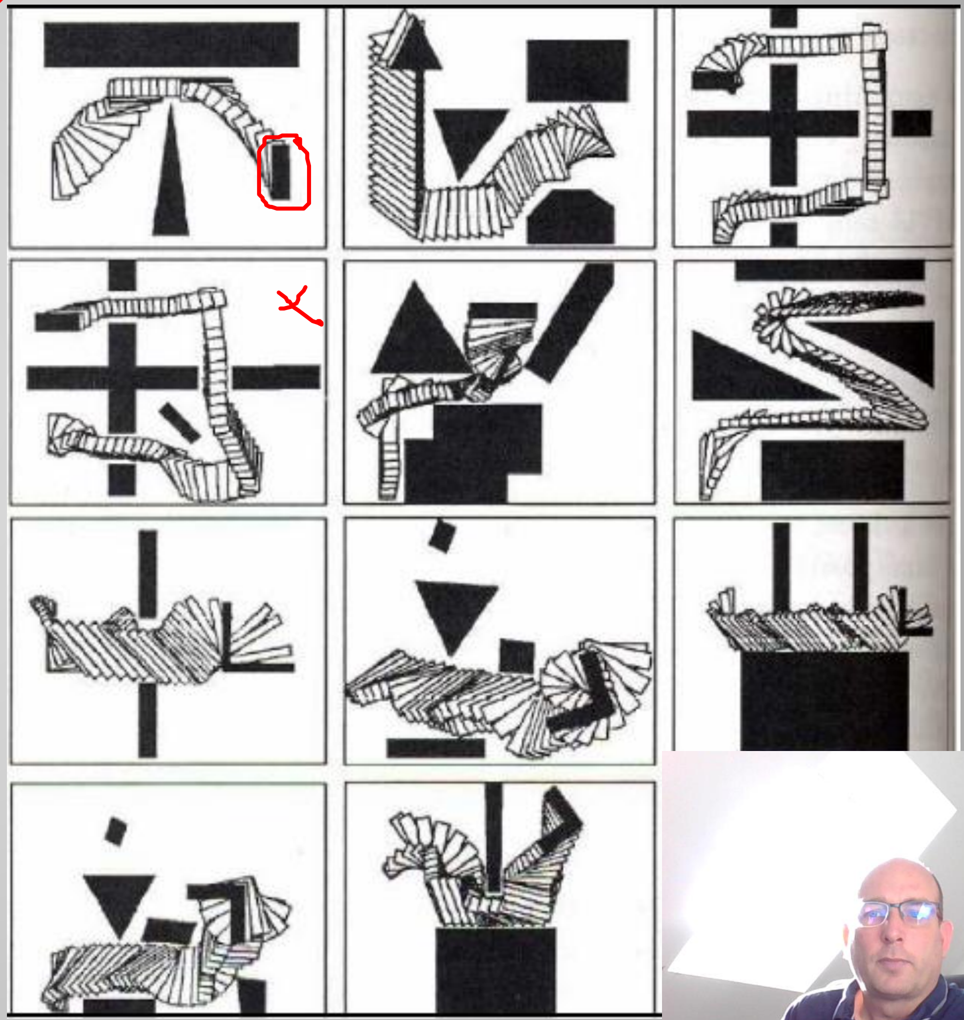
Approximate cell decomposition

Good:

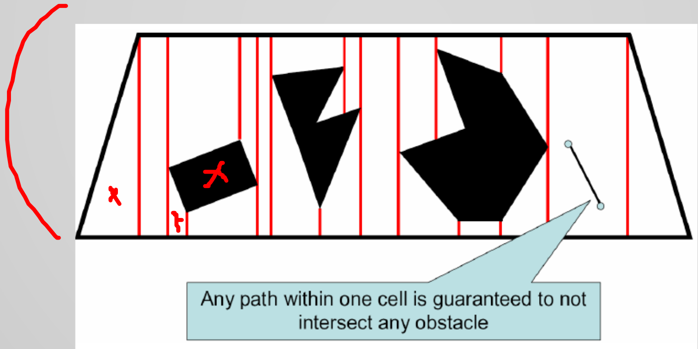
- Limited assumptions on obstacle configuration
- Approach used in practice
- Find obvious solutions quickly

Bad:

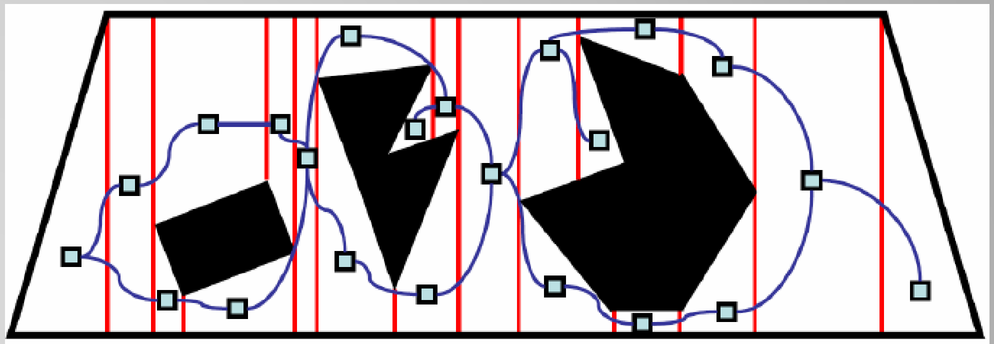
- No clear notion of optimality ("best" path)
- Trade-off completeness/computation
- Still difficult to use in high dimensions



EXACT CELL DECOMPOSITION

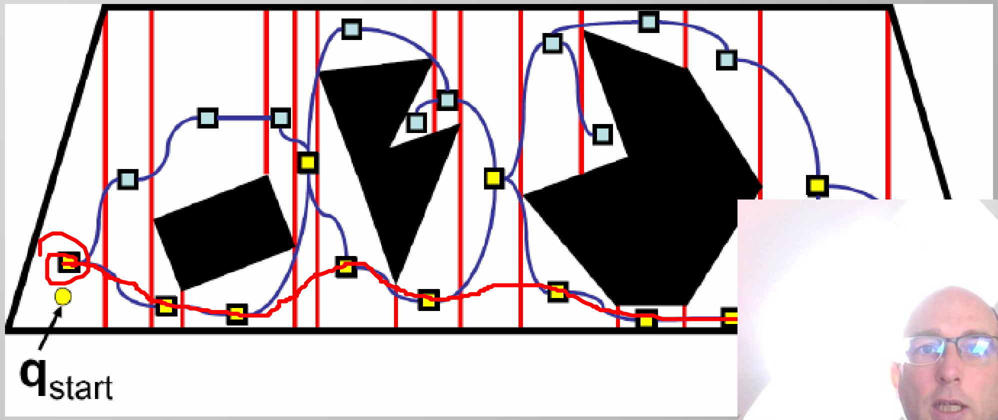


Exact cell decomposition



The graph of cells defines a roadmap

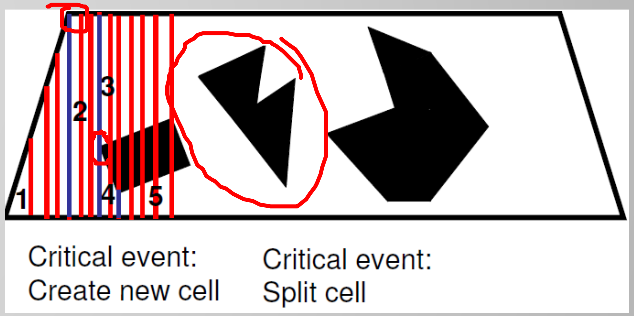
And can be used to find a path between any two configuration



Exact cell decomposition

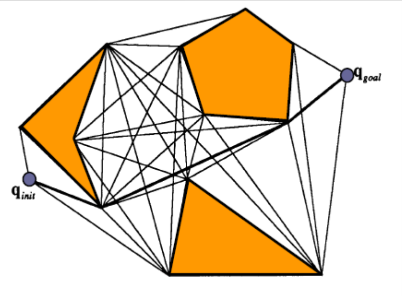
Plane Sweep algorithm

- Initialize current list of cells to empty
- Order the vertices of Cobs along the *x* direction
- For every vertex:
  - Construct the plane at the corresponding *x* location
  - Depending on the type of event:
- Split a current cell into 2 new cells OR
- Merge two of the current cells
  - Create a new cell
- Complexity (in 2-D):
  - Time: O(N log N)
  - Space: O(N)

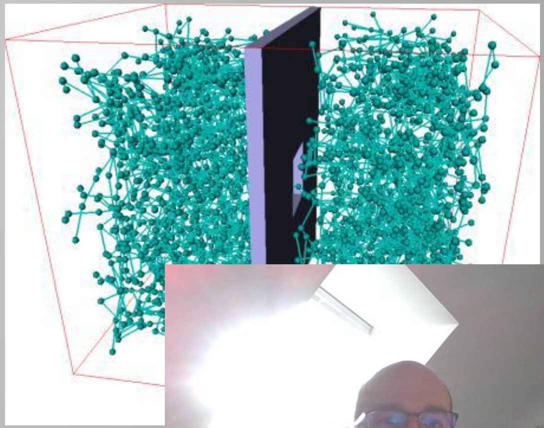


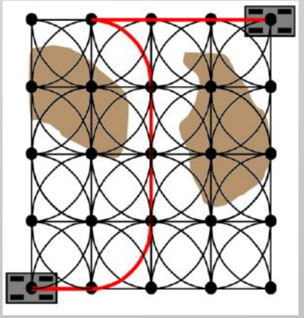
Overview

- Roadmap methods
  - Visibility graphs
  - Voronoi diagrams
- Decomposition methods
  - Approximate
  - Exact
- Potential fields
- Sampling-based Planning
  - PRM
  - RRT
- Applications to Computer Animation
- Dynamic environments
- Crowd Simulation



- Deterministic, heuristic or probabilistic
- Planning algorithms evaluation criteria:
  - Completeness
  - Optimality
  - Speed
  - Generality

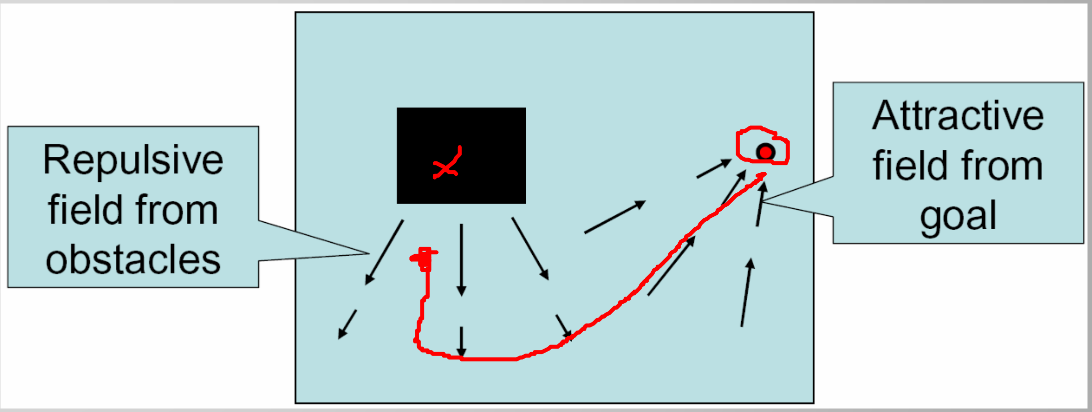




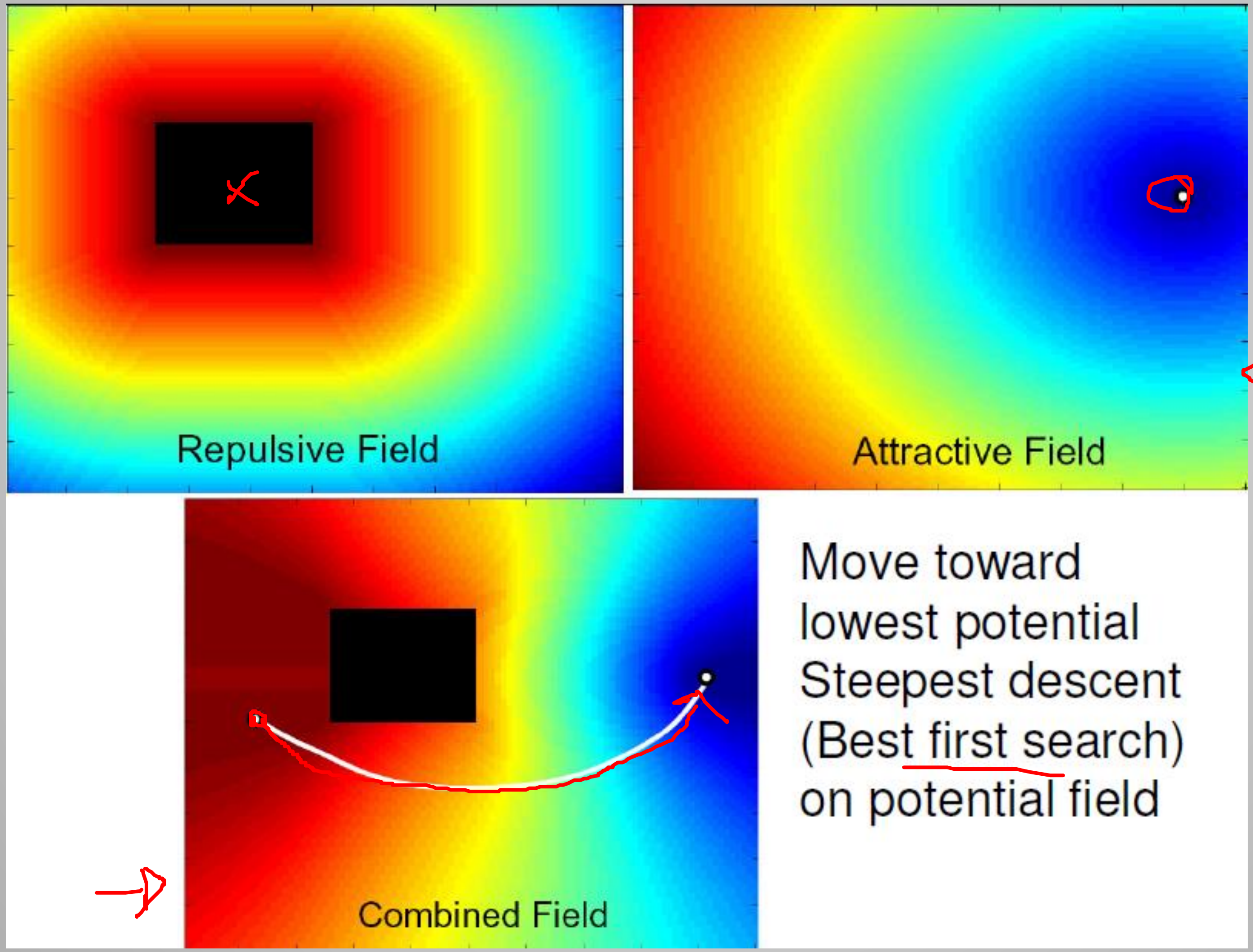
Potential fields

- Stay away from obstacles: Imagine that the obstacles are made of a material that generate a *repulsive* field

- Move closer to the goal: Imagine that the goal location is a particle that generates an *attractive* field



Potential fields



Move toward lowest potential
Steepest descent
(Best first search)
on potential field

Potential fields

$$U_g(\mathbf{q}) = d^2(\mathbf{q}, \mathbf{q}_{goal})$$

Distance to goal state

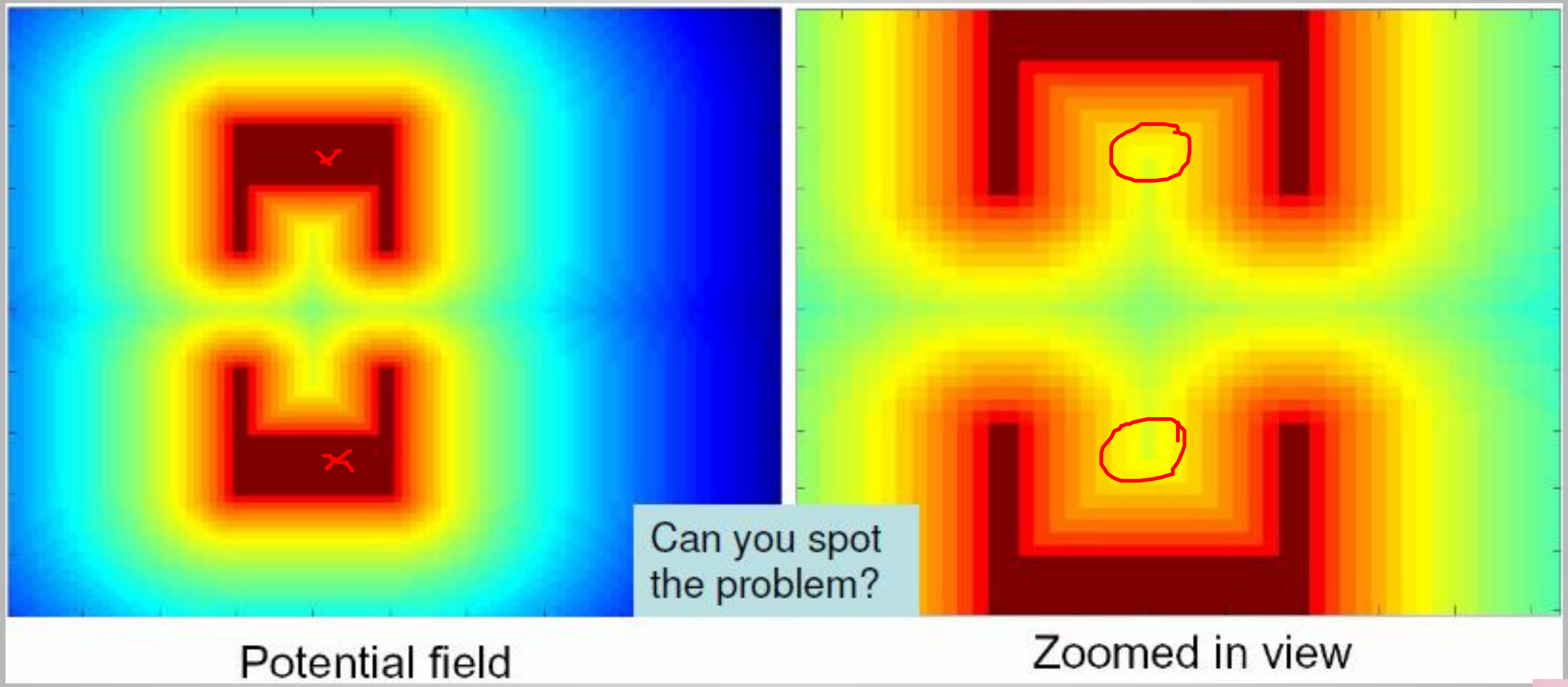
$$U_o(\mathbf{q}) = \frac{1}{d^2(\mathbf{q}, Obstacles)}$$

Distance to nearest obstacle point. Note: Can be computed efficiently by using the *distance transform*

$$U(\mathbf{q}) = U_g(\mathbf{q}) + \lambda U_o(\mathbf{q})$$

λ controls how far we stay from the obstacles

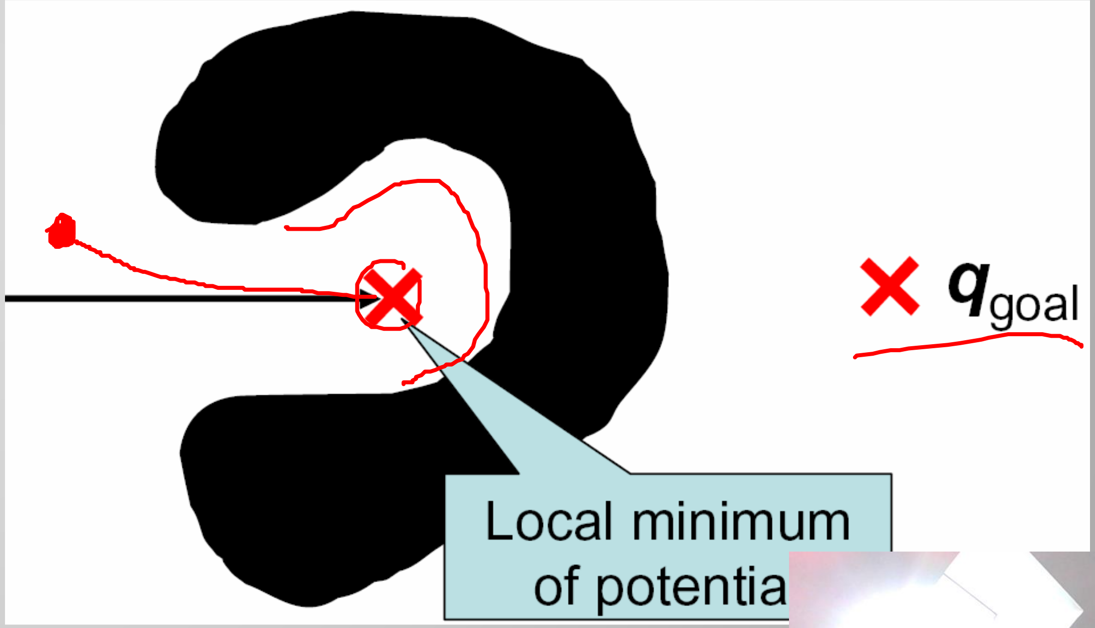
Potential fields



Potential fields in general exhibit local minima

Special case: Navigation function

- $U(\boldsymbol{q}_{goal}) = 0$
- For any $\boldsymbol{q}$ different from $\boldsymbol{q}_{goal}$, there exists a neighbor $\boldsymbol{q}'$ such that $U(\boldsymbol{q}') < U(\boldsymbol{q})$



GETTING OUT OF LOCAL MINIMA

Repeat

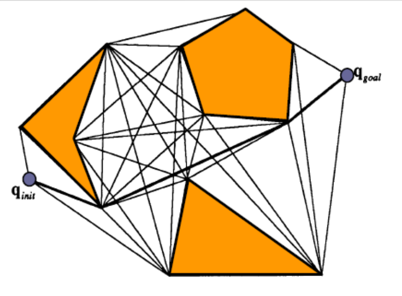
- If $U(\boldsymbol{q}) = 0$ return Success
- If too many iterations return Failure
- Else:
  - Find neighbor $\boldsymbol{q}_n$ of $\boldsymbol{q}$ with smallest $U(\boldsymbol{q}_n)$
  - If $U(\boldsymbol{q}_n) < U(\boldsymbol{q})$ OR $\boldsymbol{q}_n$ has not yet been visited
- Move to $\boldsymbol{q}_n$ ($\boldsymbol{q} \leftarrow \boldsymbol{q}_n$)
- Remember $\boldsymbol{q}_n$

Repeat

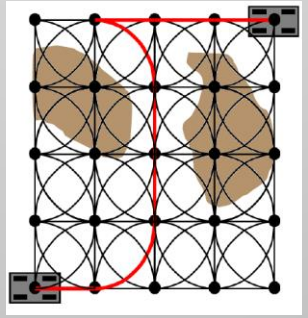
- If $U(\boldsymbol{q}) = 0$ return Success
- If too many iterations return Failure
- Else:
  - Find neighbor $\boldsymbol{q}_n$ of $\boldsymbol{q}$ with smallest $U(\boldsymbol{q}_n)$
  - If $U(\boldsymbol{q}_n) < U(\boldsymbol{q})$
    - Move to $\boldsymbol{q}$n ($\boldsymbol{q} \leftarrow \boldsymbol{q}_n$)
  - Else
    - Take a random walk for T steps starting at $\boldsymbol{q}_n$
    - Set $\boldsymbol{q}$ to the configuration rea
      of the random walk

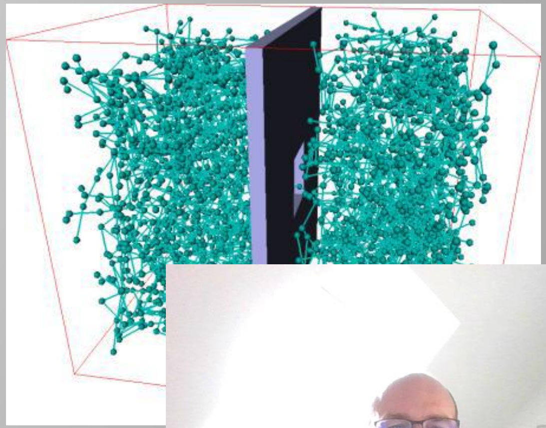
Overview

- Roadmap methods
  - Visibility graphs
  - Voronoi diagrams
- Decomposition methods
  - Approximate
  - Exact
- Potential fields
- Sampling-based Planning
  - PRM
  - RRT
- Applications to Computer Animation
- Dynamic environments
- Crowd Simulation



- Deterministic, heuristic or probabilistic
- Planning algorithms evaluation criteria:
  - Completeness
  - Optimality
  - Speed
  - Generality





Planning in High-Dimensional Spaces

IDEAL: A complete planner

- Guarantees to find a solution in finite time if exists
- Indicates the non existence of a solution in a finite time
- Note: completeness means the computation of 1 path, among others.

PROBLEM: P-SPACE complexity [Reif '79]

Solutions:

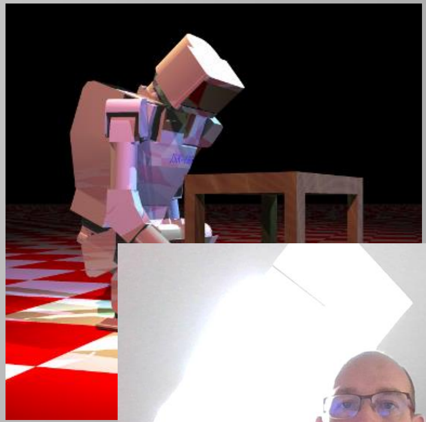
- Lower the dimension of the search space
- Limit the number of possible solutions (e.g., bound the search space)
- Sacrifice optimality
- Sacrifice completeness

Key idea: Instead of searching the whole configuration space, randomly explore solutions and capture them

Facilitates a « probing » instead of exhaustive exploration

Drawbacks?

- Completeness and optimality are lost
- Classical trade-off however



Probabilistic Roadmaps Method (PRM)

Relies on 3 elements:

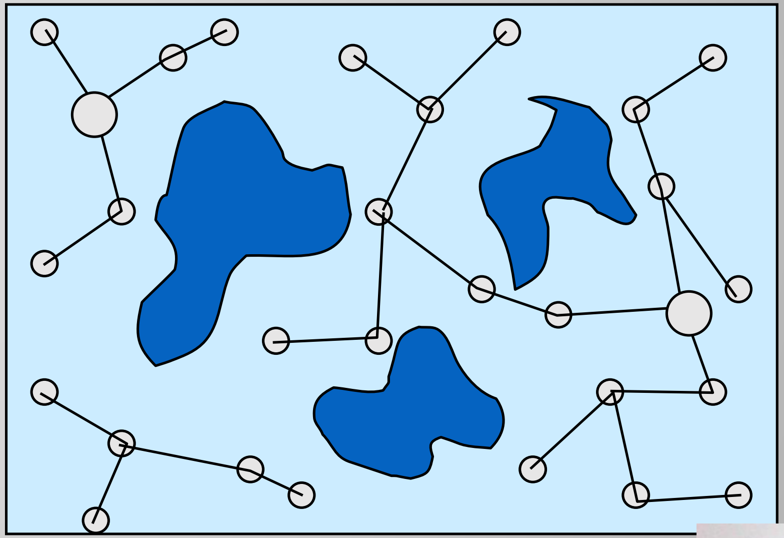
- Collision checker
- Local Method
- Sampler

2 major steps:

- Exploration Phase
- Query Phase

Key Idea: explore randomly C-space and capture C-free topology into a roadmap

Complete in infinite time: probabilistically complete



Local method

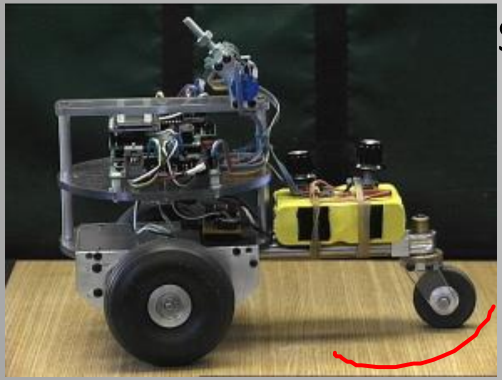
In charge of kinematics constraints

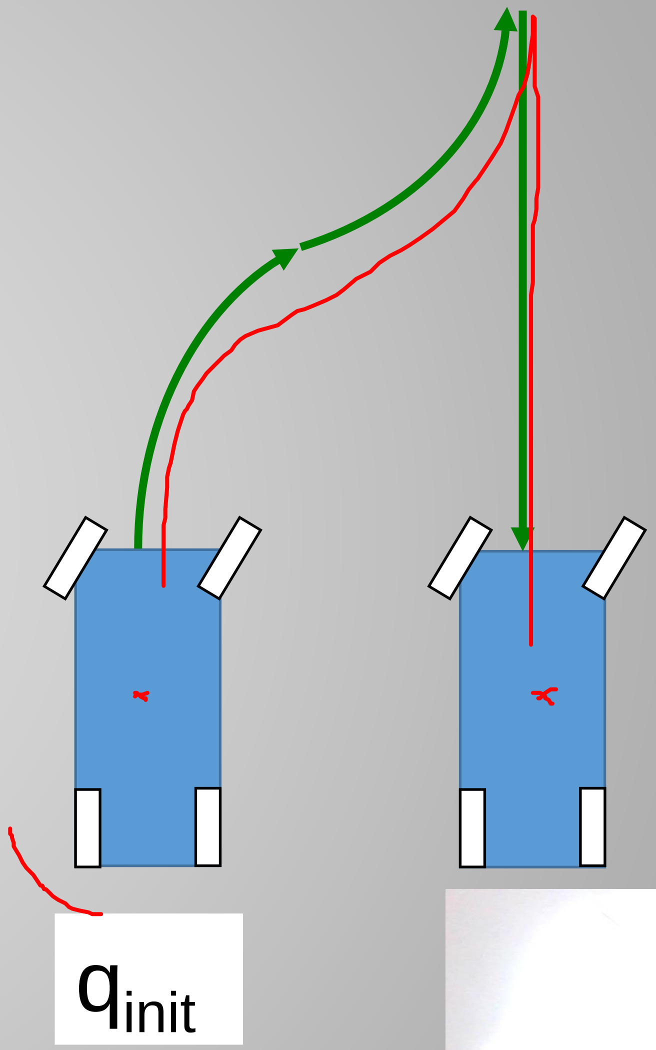
Given q_{init} and q_{goal}, how to compute a local path joining them?

Not taking into account obstacles

Examples:

- Free: linear
- Shepp





PREPROCESSING: LEARNING PHASE

ITERATIVE ALGORITHM

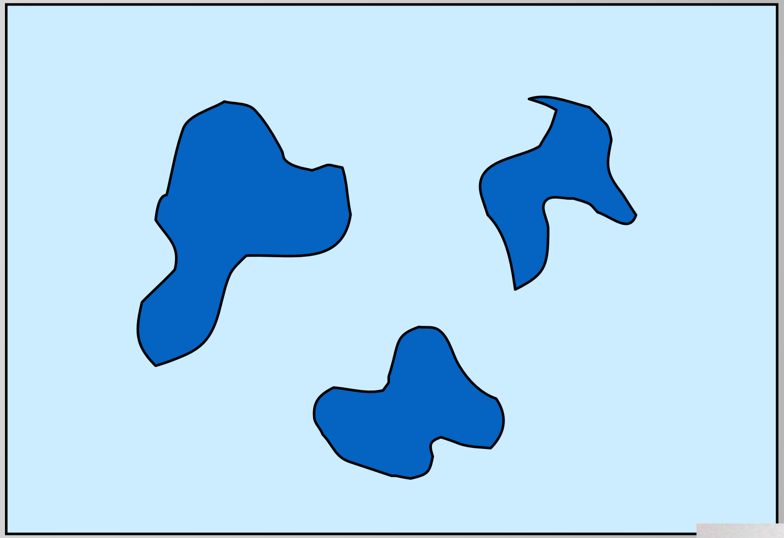
1. COMPUTE RANDOM CONFIGURATION

- Collision checker

2. CONNECT CONFIGURATION

- Collision checker
- Local method

3. GOTO 1



PREPROCESSING: LEARNING PHASE

ITERATIVE ALGORITHM

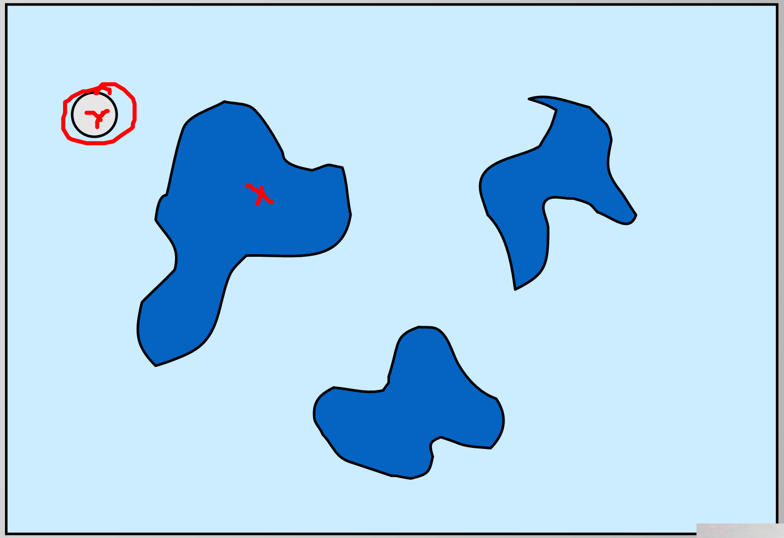
1. COMPUTE RANDOM CONFIGURATION

- Collision checker

2. CONNECT CONFIGURATION

- Collision checker
- Local method

3. GOTO 1



Preprocessing: learning phase

Iterative algorithm

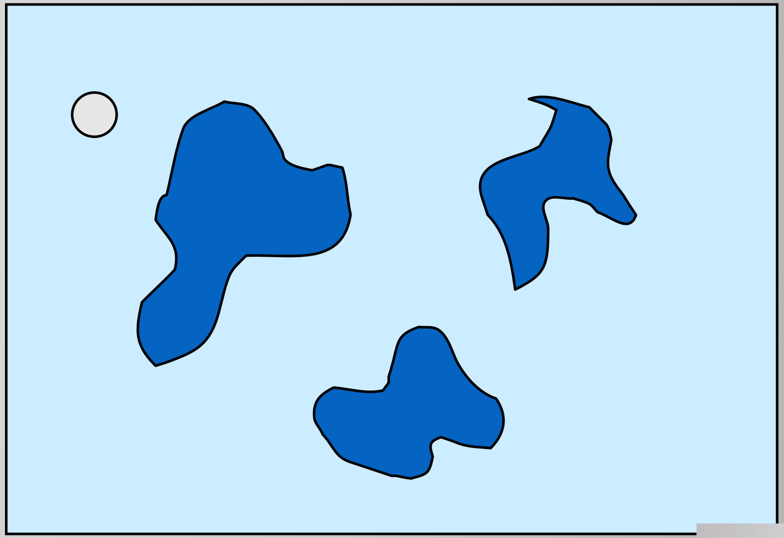
1. Compute random configuration

- Collision checker

2. Connect configuration

- Collision checker
- Local method

3. Goto 1



Preprocessing: Learning Phase

Iterative algorithm

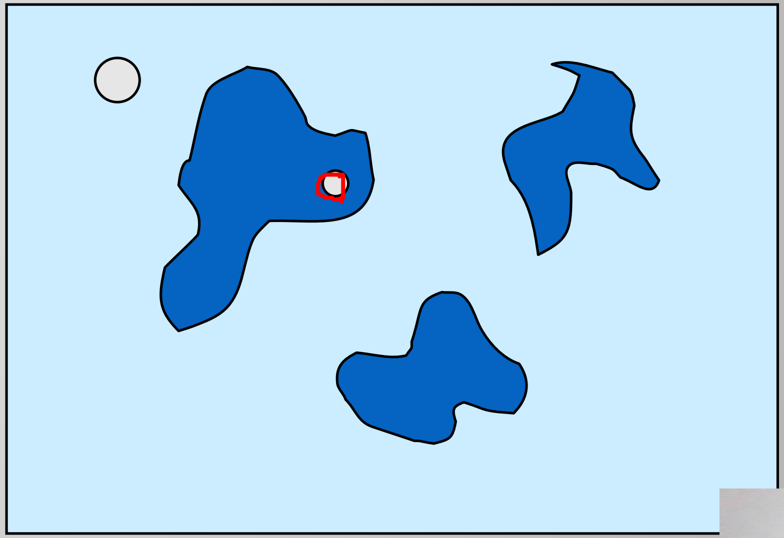
1. Compute random configuration

- Collision checker

2. Connect configuration

- Collision checker
- Local method

3. Goto 1



Preprocessing: learning phase

Iterative algorithm

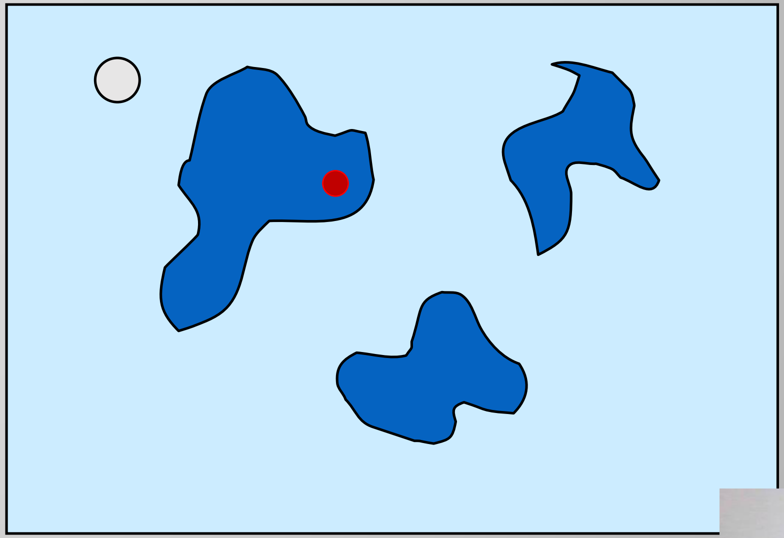
1. Compute random configuration

- Collision checker

2. Connect configuration

- Collision checker
- Local method

3. Goto 1



Preprocessing: learning phase

Iterative algorithm

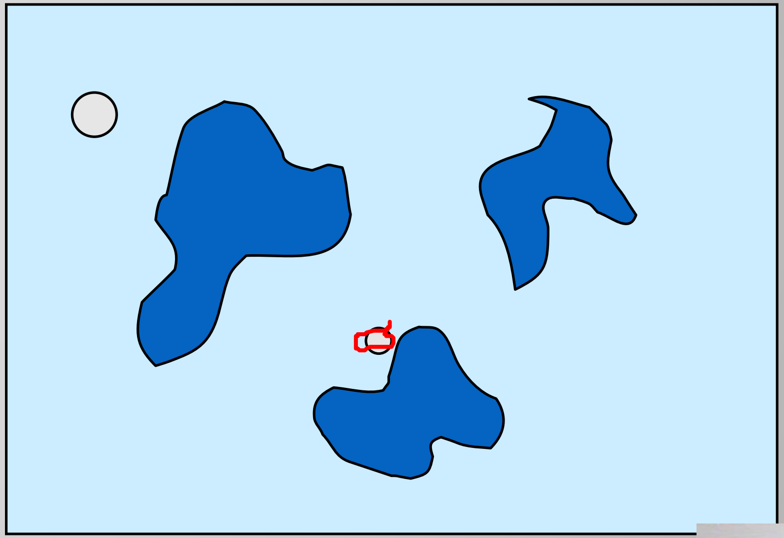
1. Compute random configuration

- Collision checker

2. Connect configuration

- Collision checker
- Local method

3. Goto 1



Preprocessing: learning phase

Iterative algorithm

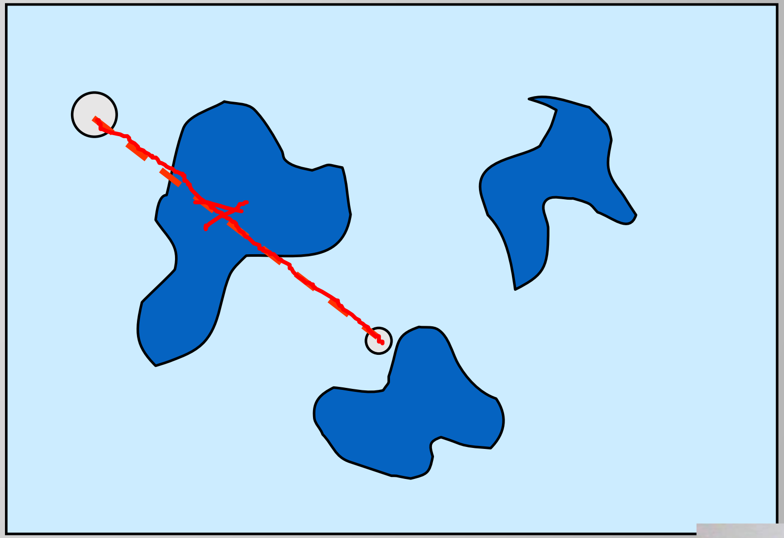
1. Compute random configuration

- Collision checker

2. Connect configuration

- Collision checker
- Local method

3. Goto 1



PREPROCESSING: LEARNING PHASE

ITERATIVE ALGORITHM

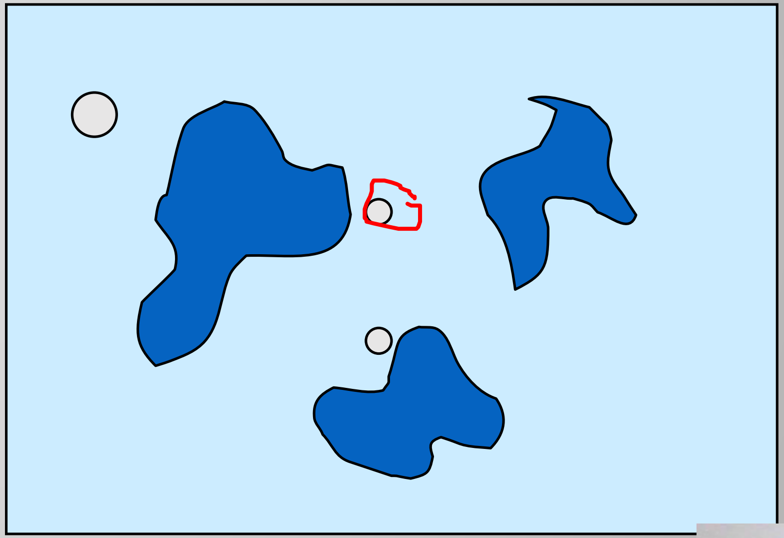
1. COMPUTE RANDOM CONFIGURATION

- Collision checker

2. CONNECT CONFIGURATION

- Collision checker
- Local method

3. GOTO 1



Preprocessing: learning phase

Iterative algorithm

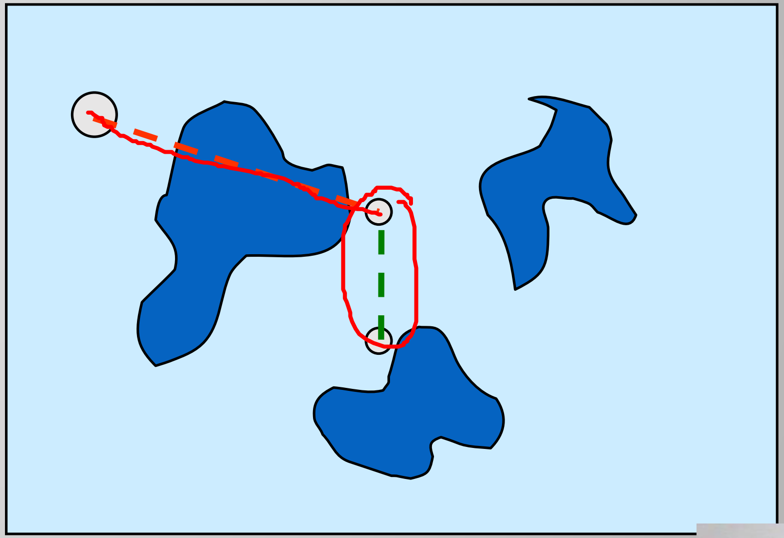
1. Compute random configuration

- Collision checker

2. Connect configuration

- Collision checker
- Local method

3. Goto 1



PREPROCESSING: LEARNING PHASE

ITERATIVE ALGORITHM

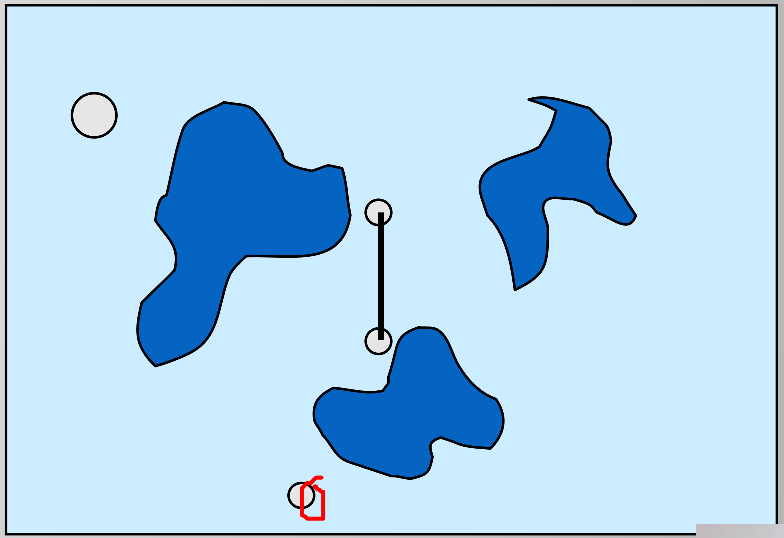
1. COMPUTE RANDOM CONFIGURATION

- Collision checker

2. CONNECT CONFIGURATION

- Collision checker
- Local method

3. GOTO 1



PREPROCESSING: LEARNING PHASE

ITERATIVE ALGORITHM

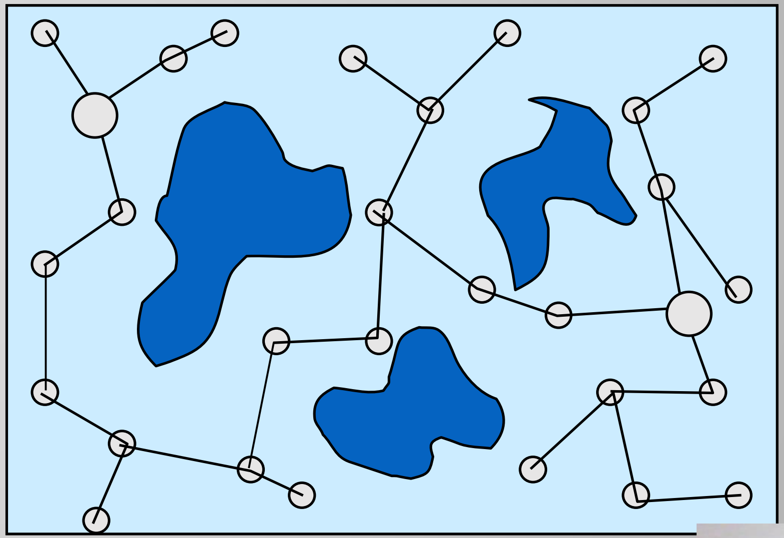
1. COMPUTE RANDOM CONFIGURATION

- Collision checker

2. CONNECT CONFIGURATION

- Collision checker
- Local method

3. GOTO 1



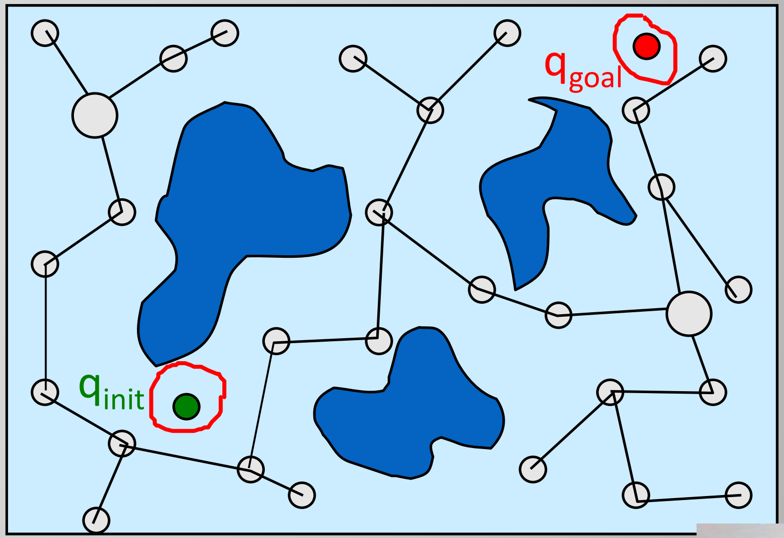
Query Phase

Roadmap is reused for solving queries

1. Connect desired initial and final configurations

2. If corresponding nodes belong to the same connected component, a solution exists

3. Graph Search



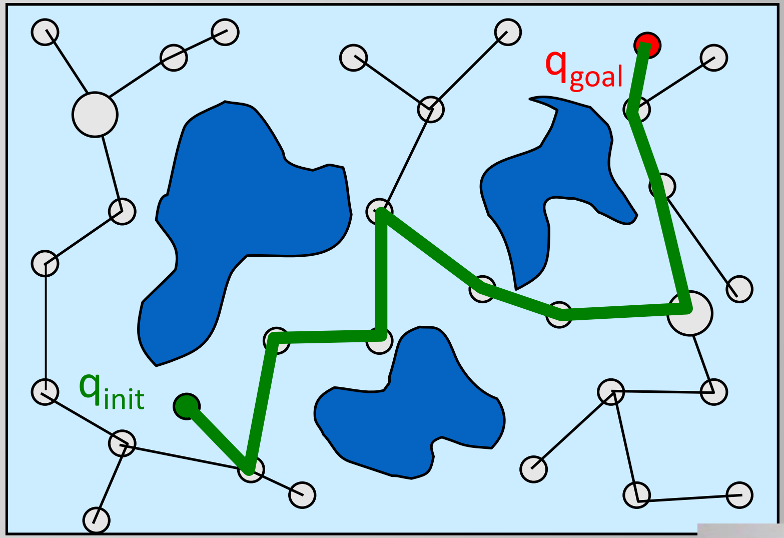
Query Phase

Roadmap is reused for solving queries

1. Connect desired initial and final configurations

2. If corresponding nodes belong to the same connected component, a solution exists

3. Graph Search



Query Phase

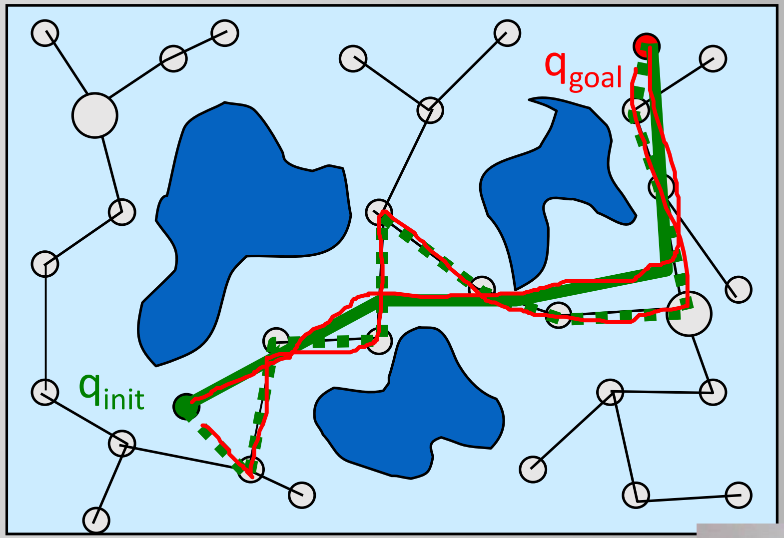
Roadmap is reused for solving queries

1. Connect desired initial and final configurations

2. If corresponding nodes belong to the same connected component, a solution exists

3. Graph Search

4. Optimization

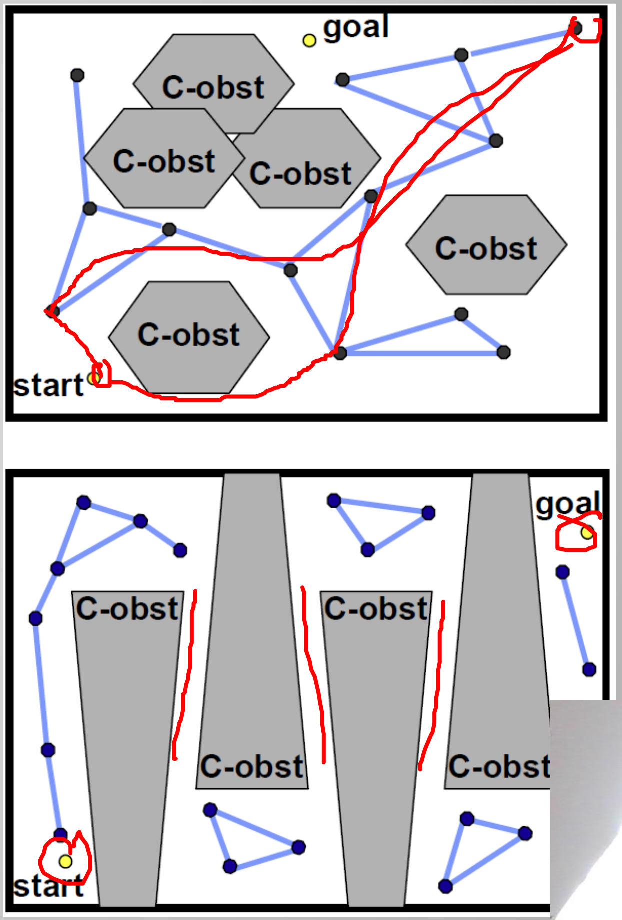


Good & bad news

Good News

- Probabilistically complete
- Do not construct the C-space
- Apply easily to high-dimensional C-space
- support fast queries w/ enough preprocessing

Many success stories where PRMs solve previously unsolved problems



The Bad News

- Don’t work as well for some problems:
  - unlikely to sample nodes in *narrow passages*
  - hard to sample/connect nodes on constraint surfaces
- No optimality or completeness

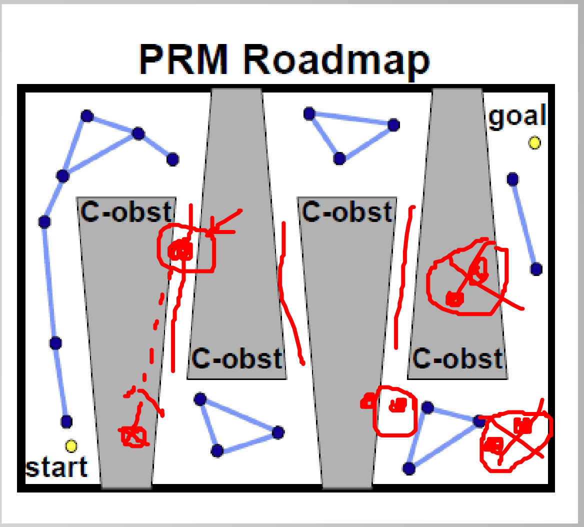
Sample Near Obstacles

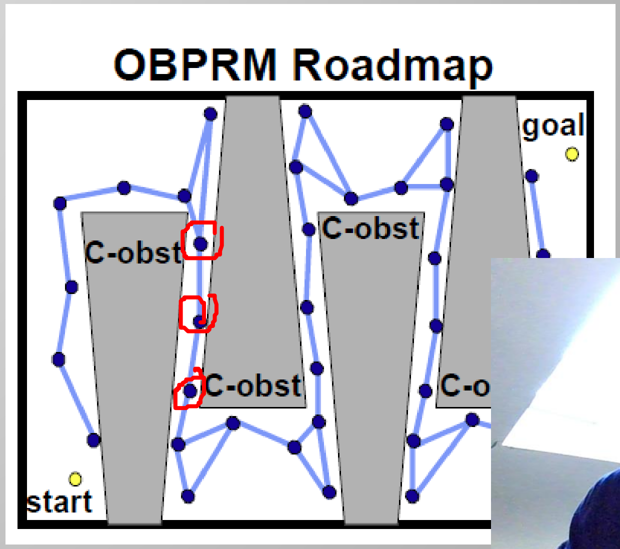
OBPRM

- q_{in} found in collision
- Generate random direction v
- Find q_{out} in direction v that is free
- Binary search from q_{in} to obstacle boundary to generate node

Gaussian Sampler

- Find a q_1
- Find another q_2 picked from a Gaussian distribution centered at q_1
- If they are both in collision or free, discard. Otherwise, keep the free

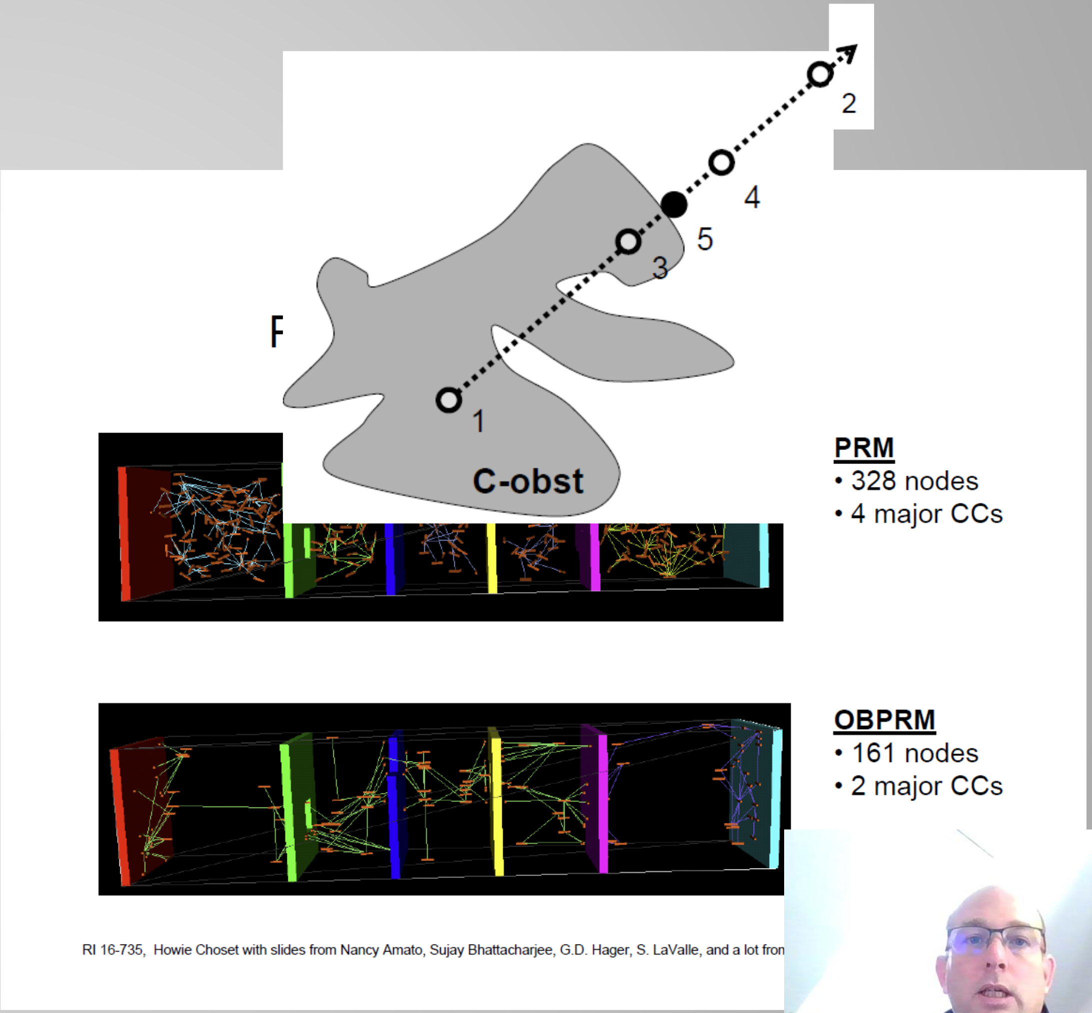




OBPRM: Finding Points on C-obstacles

Basic Idea:

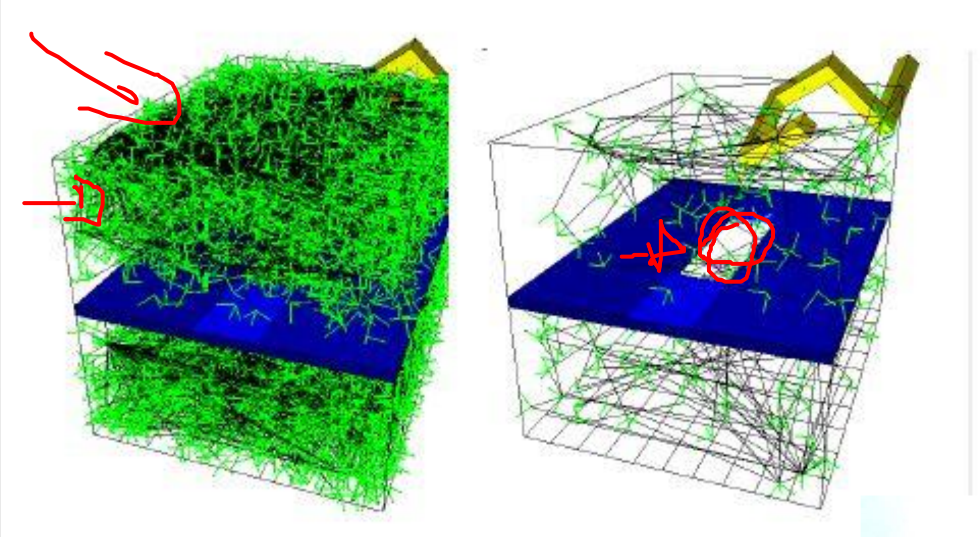
1. Find a point in S's C-obstacle (robot placement colliding with S)
2. Select a random direction in C-space
3. Find a free point in that direction
4. Find boundary point between them using binary search (collision checks)



Sampling Strategy

Highly constrained problems result in huge roadmaps:

- Construction is time consuming
- Search is time consuming



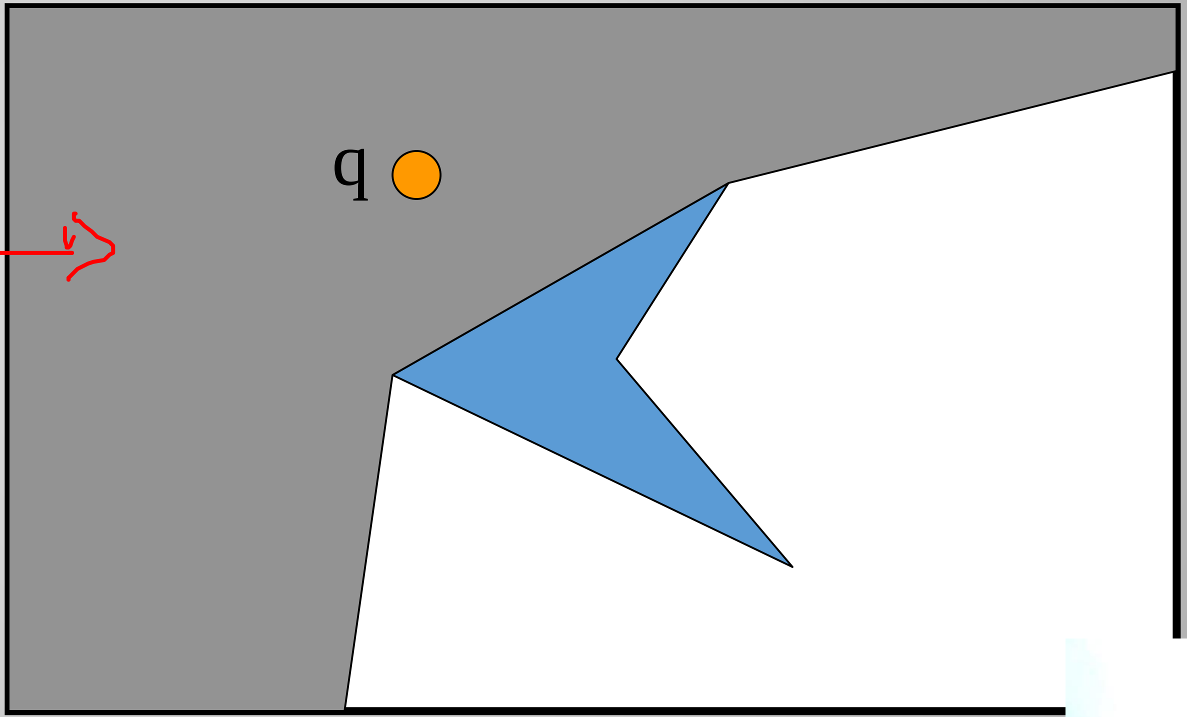
Sampling strategies help in reducing the roadmap size

Example:

- Visibility-PRM

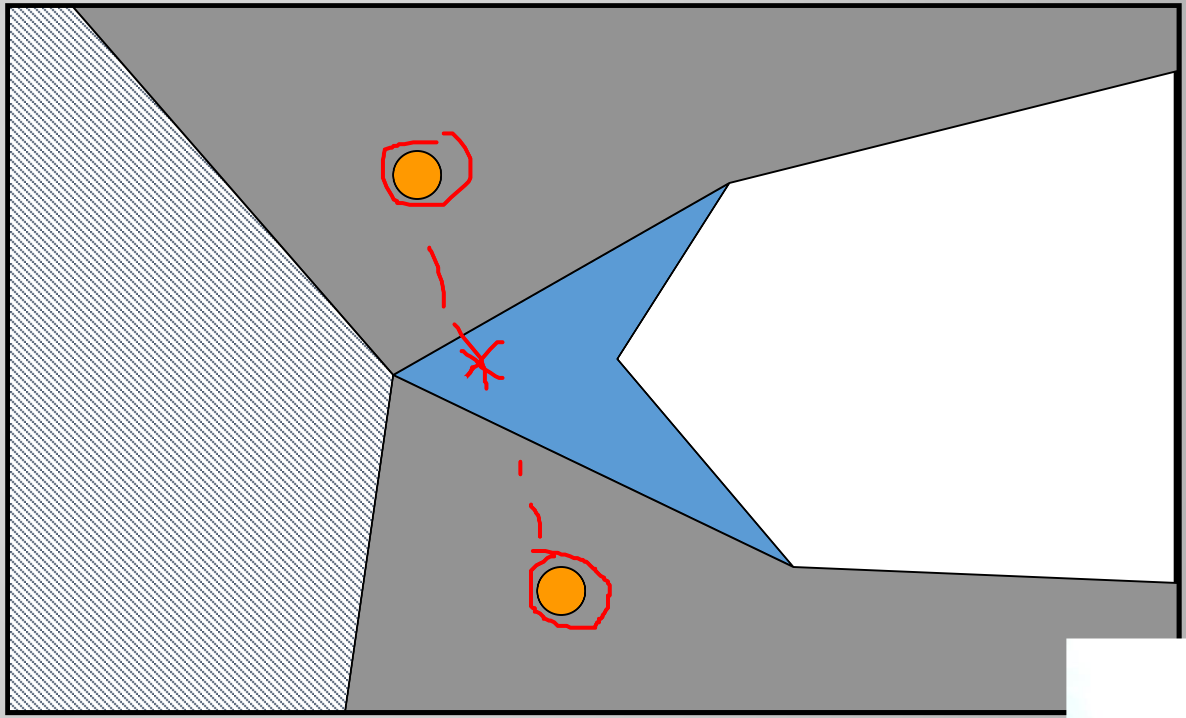
Visibility-PRM

Visibility Domain of configuration q:



Visibility-PRM

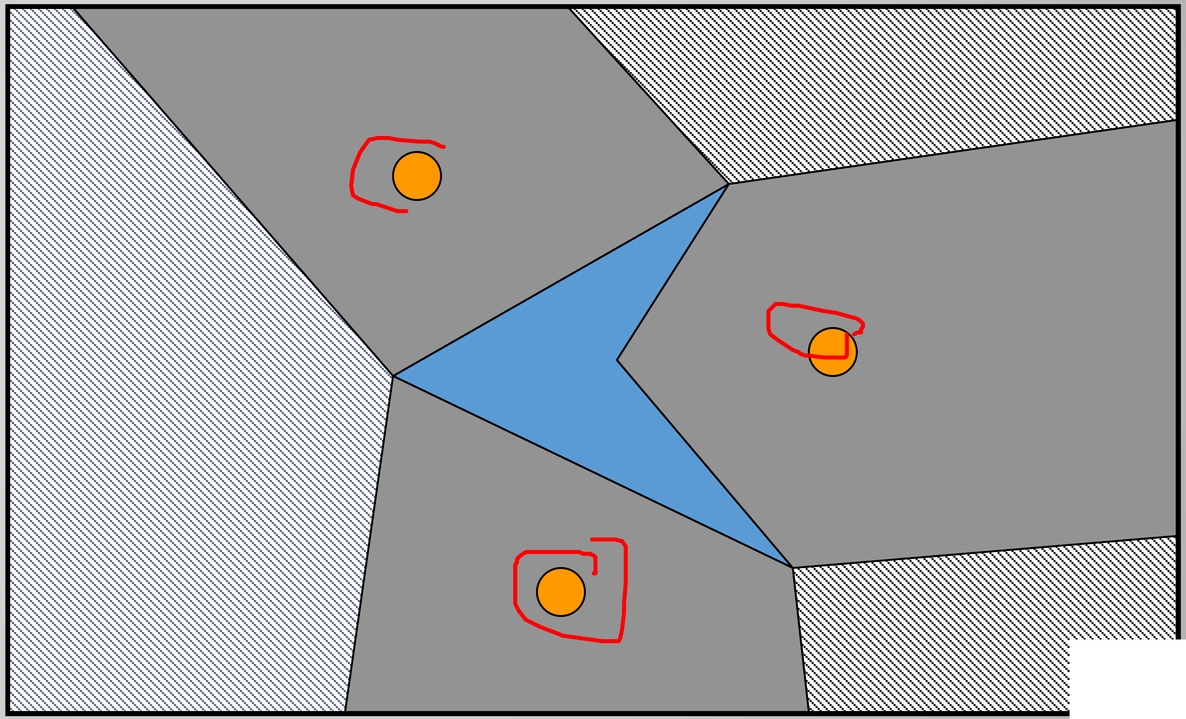
A new configuration is retained only if out of the visibility domain of other configurations



Visibility-PRM

A new configuration is retained only if out of the visibility domain of other configurations

These configurations are called "guardians"

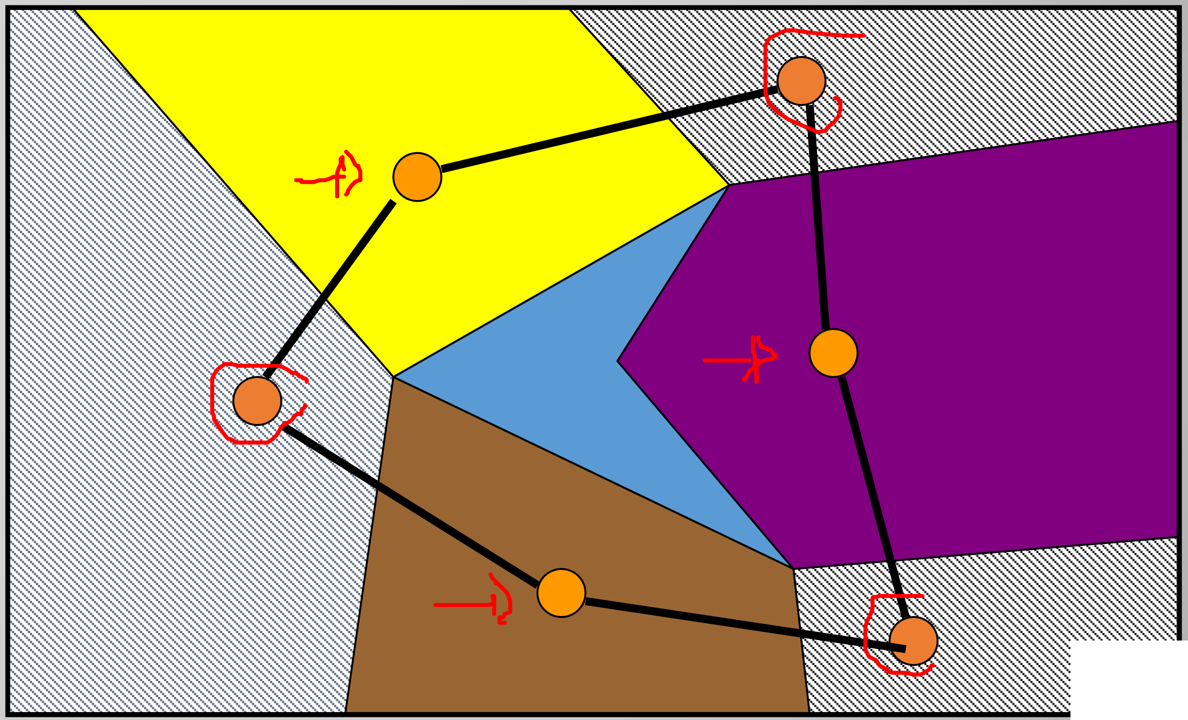


Visibility-PRM

A new configuration is retained only if out of the visibility domain of other configurations

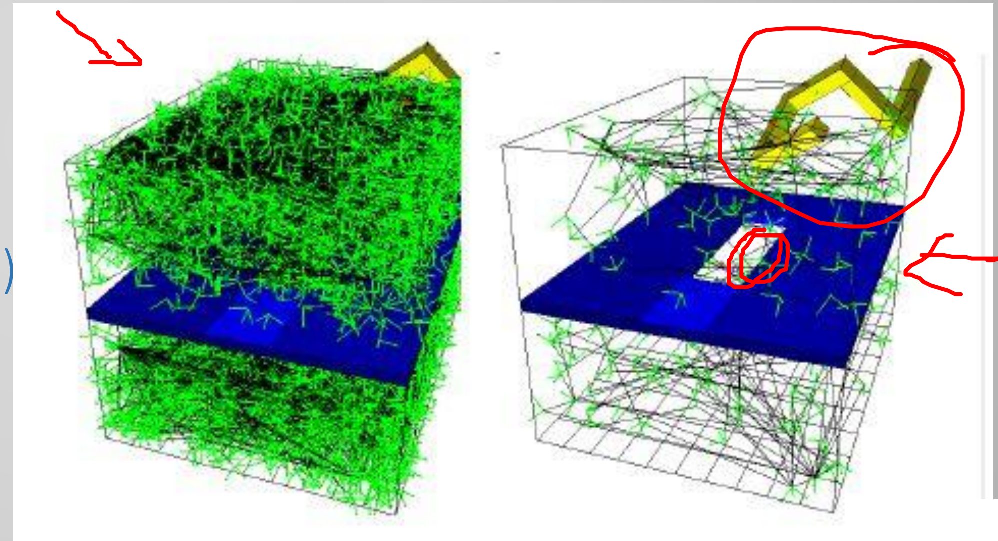
Or if allow to connect 2 guardians

These configurations are called "connectors"



VISIBILITY-PRM

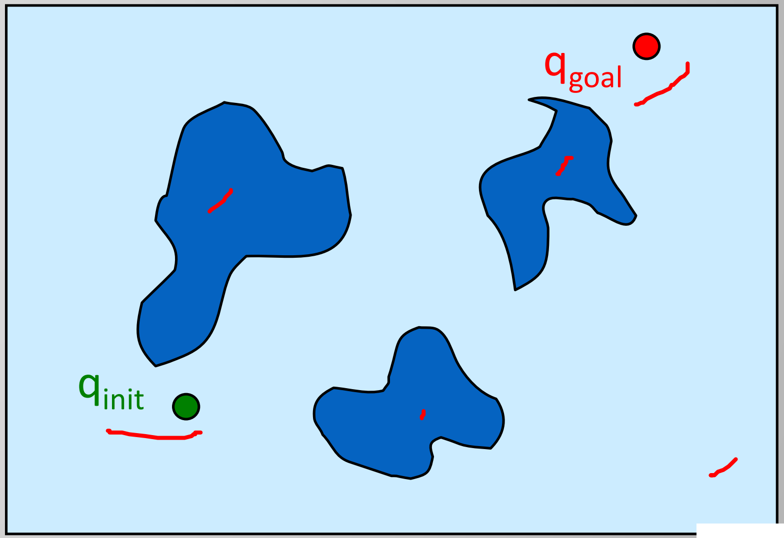
(THIS IS A 6-DIMENSIONAL C-SPACE IN 3-D)



RRT: Rapidly-exploring Random trees

PRM is a multi-query method: the same roadmap is reused to solve different queries

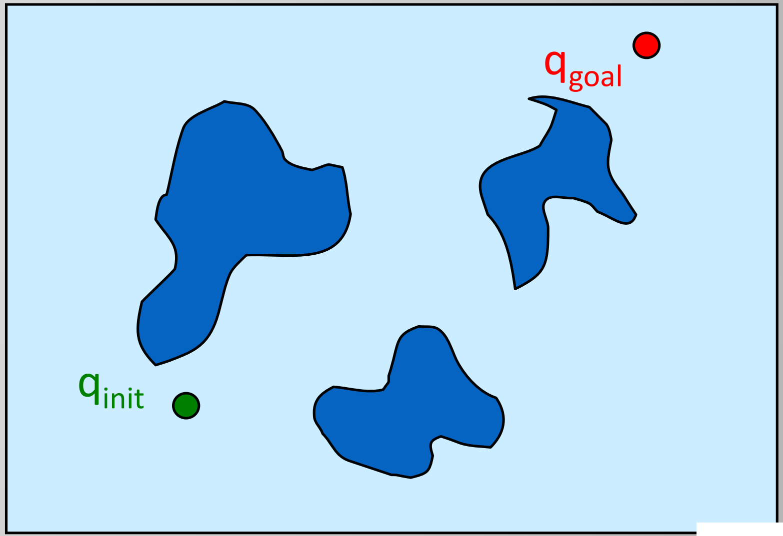
RRT is single-query: the problem is solved without preliminary exploration of C-free



RRT: Rapidly-exploring Random trees

Iterative algorithm:

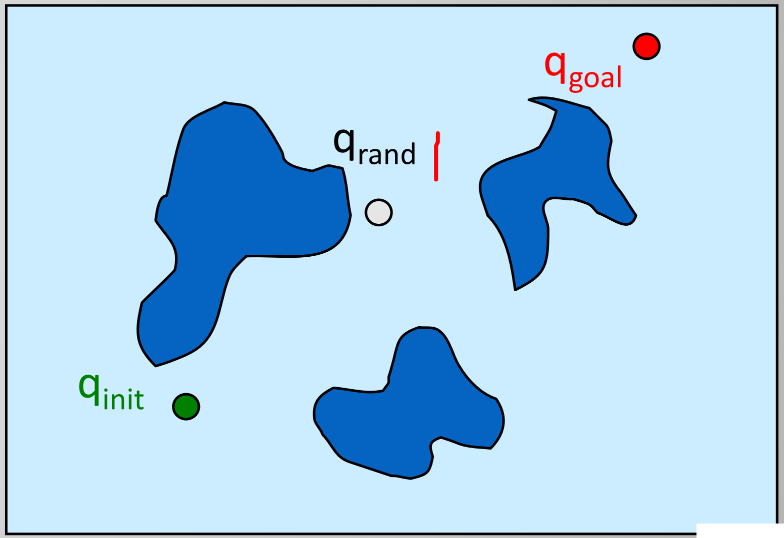
1. Compute q_{rand}
2. Connect to q_{near}
3. Insert q_{new}
4. Goto 1



RRT: Rapidly-exploring Random trees

Iterative algorithm:

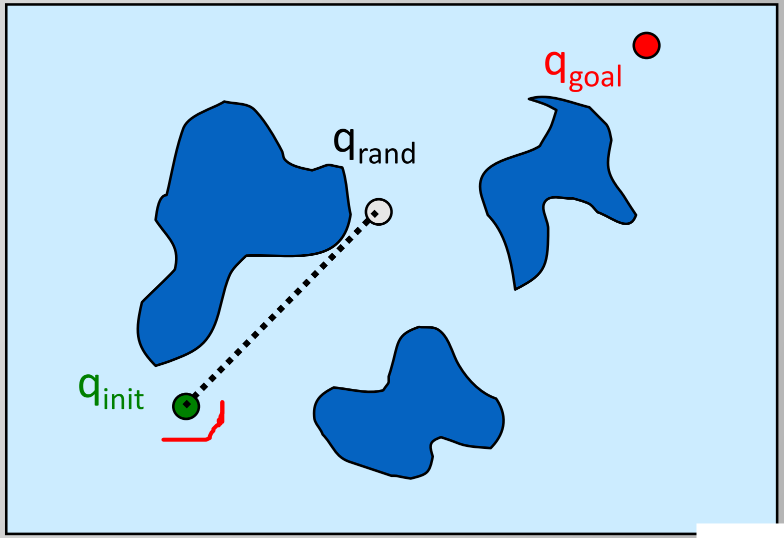
1. Compute q_{RAND}
2. Connect to q_{NEAR}
3. Insert q_{NEW}
4. Goto 1



RRT: Rapidly-exploring Random trees

Iterative algorithm:

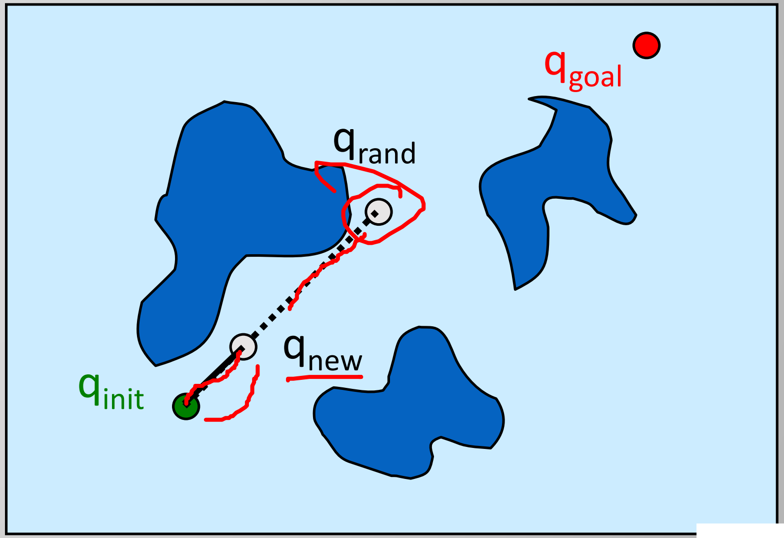
1. Compute q_{rand}
2. Connect to q_{near}
3. Insert q_{new}
4. Goto 1



RRT: Rapidly-exploring Random trees

Iterative algorithm:

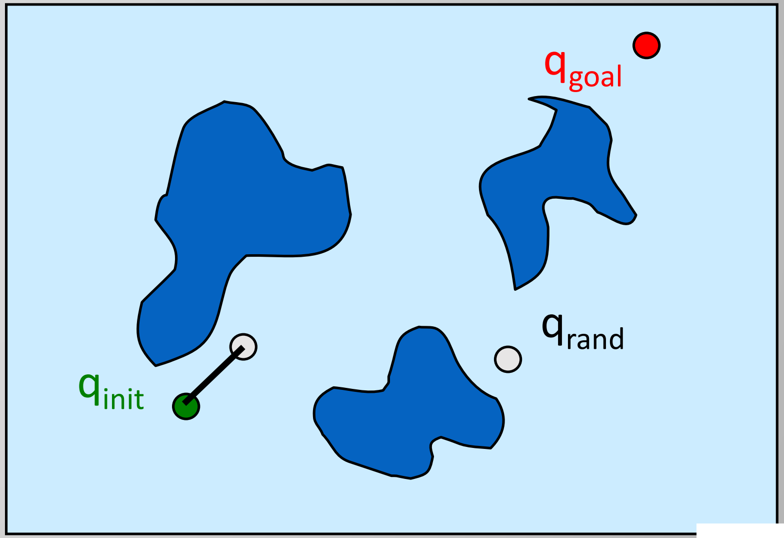
1. Compute q_{RAND}
2. Connect to q_{NEAR}
3. Insert q_{NEW}
4. Goto 1



RRT: Rapidly-exploring Random trees

Iterative algorithm:

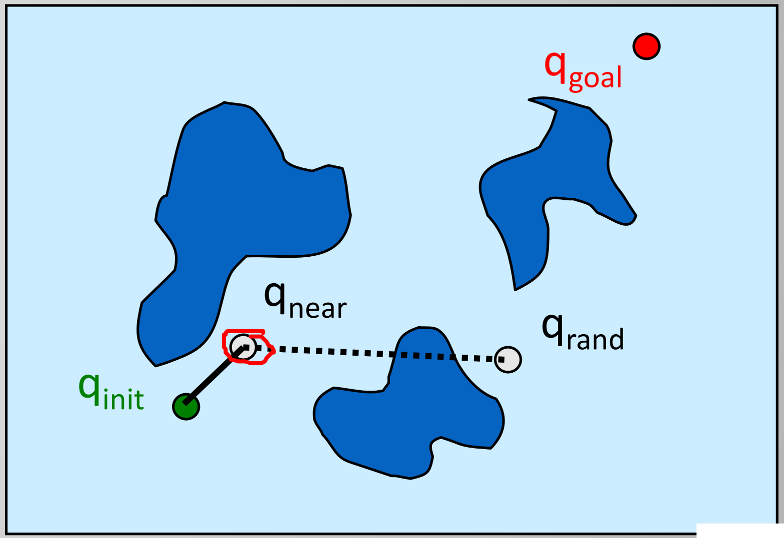
1. Compute q_{rand}
2. Connect to q_{near}
3. Insert q_{new}
4. Goto 1



RRT: Rapidly-exploring Random trees

Iterative algorithm:

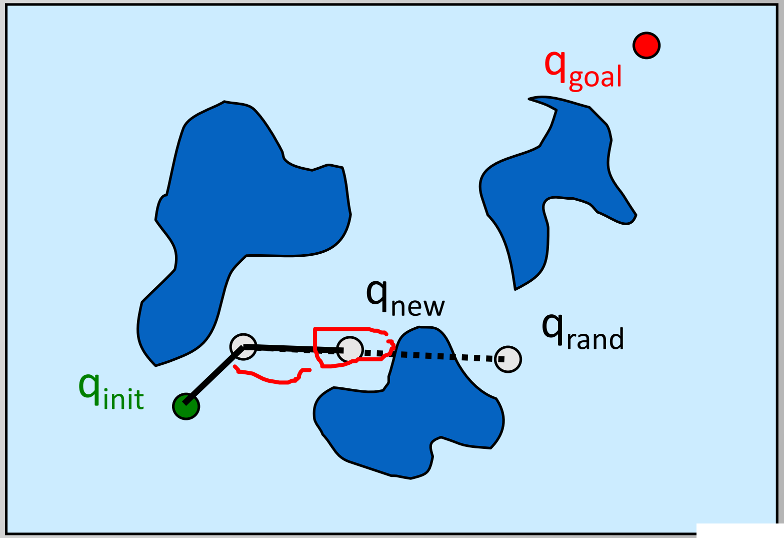
1. Compute q_{rand}
2. Connect to q_{near}
3. Insert q_{new}
4. Goto 1



RRT: Rapidly-exploring Random trees

Iterative algorithm:

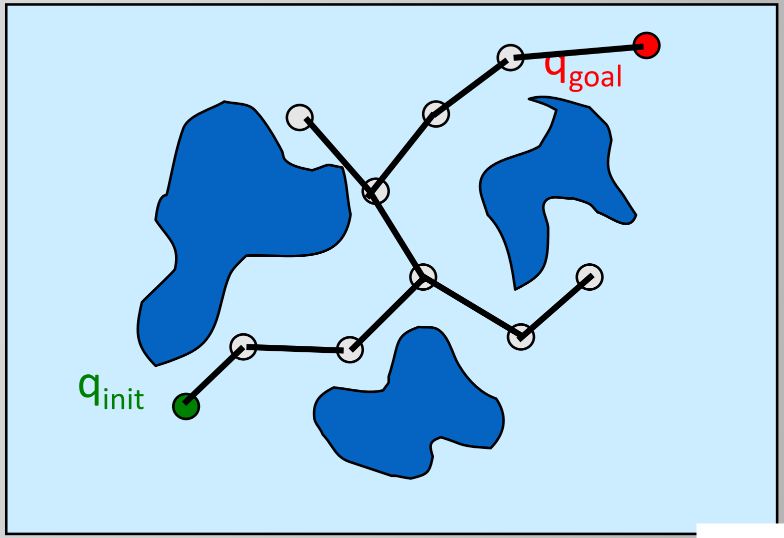
1. Compute q_{rand}
2. Connect to q_{near}
3. Insert q_{new}
4. Goto 1



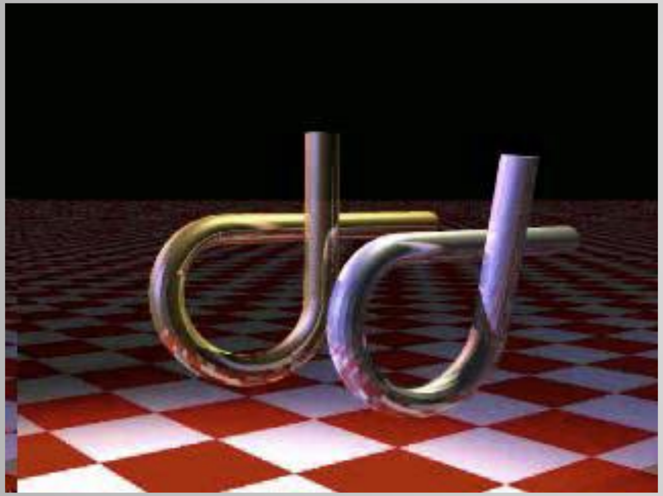
RRT: Rapidly-exploring Random trees

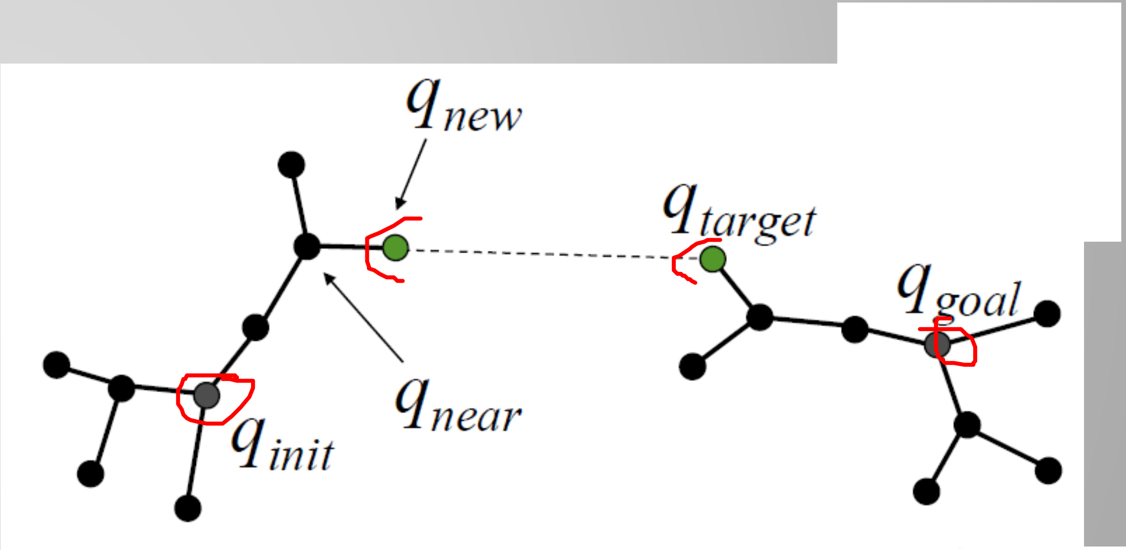
Iterative algorithm:

1. Compute q_{rand}
2. Connect to q_{near}
3. Insert q_{new}
4. until q_{new} and q_{goal} connected

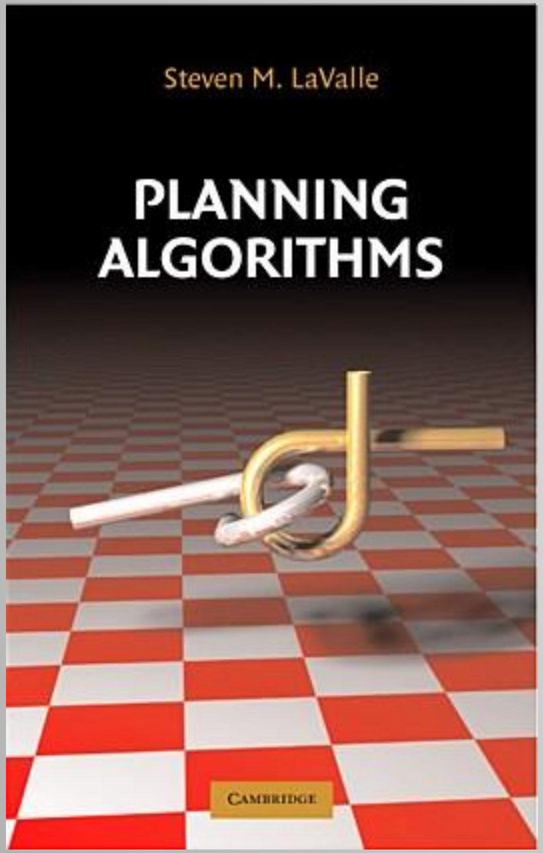


Grow two RRTs towards each other





More…



Planning Algorithms
Steven M. LaValle
http://planning.cs.uiuc.edu/