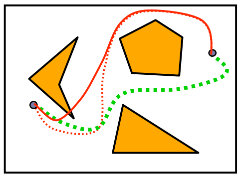
Path Planning for Point Robots

Problem

- 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



Framework

continuous representation
(configuration space formulation)

↓

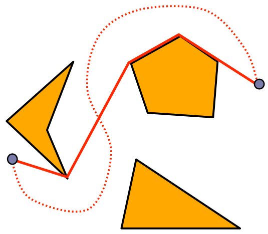
discretization
(random sampling, processing critical geometric events)

↓

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

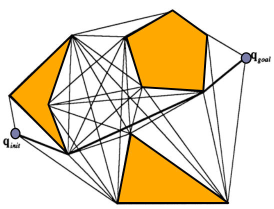
Visibility graph method

- **Observation**: If there is a a collision-free path between two points, then there is a polygonal path that bends only at the obstacles vertices.
- Why?
  Any collision-free path can be transformed into a polygonal path that bends only at the obstacle vertices.



- A **polygonal path** is a piecewise linear curve.

What is a visibility graph?



- A **visibility graph** is a graph such that
  - Nodes: q_{init}, q_{goal}, or an obstacle vertex.
  - Edges: An edge exists between nodes u and v if the line segment between u and v is an obstacle edge or it does not intersect the obstacles.

A simple algorithm for building visibility graphs

```
Input: qinit, qgoal, 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.
```

Computational efficiency

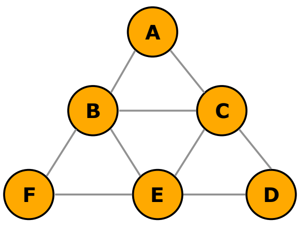
```
1: for every pair of nodes u,v                        O(n^2)
2:  if segment(u,v) is an obstacle edge then          O(n)
3:    insert edge(u,v) into G;
4:  else
5:    for every obstacle edge e                       O(n)
6:      if segment(u,v) intersects e
7:        go to (1);
8:    insert edge(u,v)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

- Simple algorithm $O(n^3)$ time
- More efficient algorithms
  - Rotational sweep $O(n^2 \log n)$ time
  - Optimal algorithm $O(n^2)$ time
  - Output sensitive algorithms
- $O(n^2)$ space

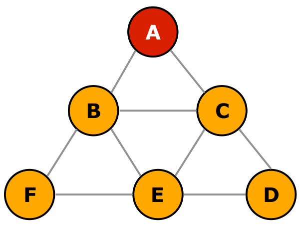
Framework

continuous representation
(configuration space formulation)
↓
discretization
(random sampling, processing critical geometric events)
↓
graph searching
(breadth-first, best-first, A*)

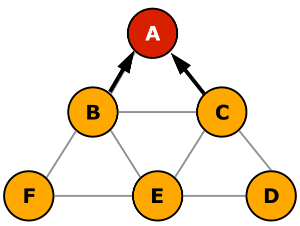
Breadth-first search



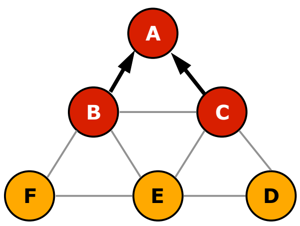
Breadth-first search



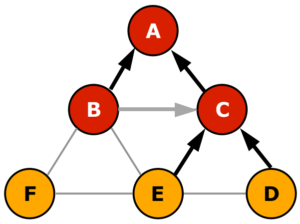
Breadth-first search



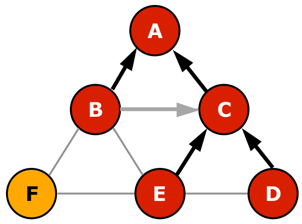
Breadth-first search



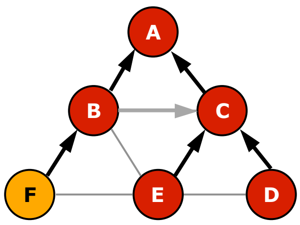
Breadth-first search



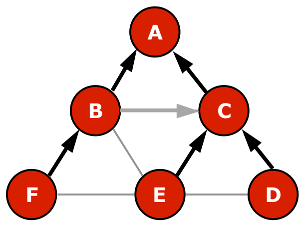
Breadth-first search



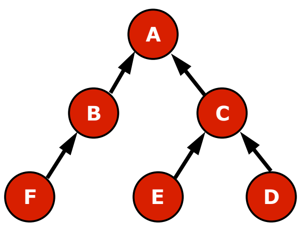
Breadth-first search



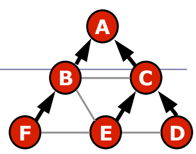
Breadth-first search



Breadth-first search



Breadth-first search



Input: q_{init}, q_{goal}, visibility graph G
Output: a path between q_{init} and q_{goal}

```
Q = new queue;
Q.enqueue(q_init);
mark q_init as visited;
while Q is not empty
  curr = Q.dequeue();
  if curr == q_goal then
    return curr;
  for each w adjacent to curr
     if w is not visited
       w.parent = curr;
       Q.enqueue(w)
       mark w as visited
```

Other graph search algorithms

- Depth-first
- Best-first, A*

Framework

continuous representation

↓

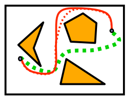
discretization

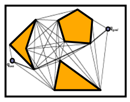
construct visibility graph

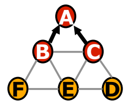
↓

graph searching

breadth-first search







Summary

- Discretize the space by constructing visibility graph
- Search the visibility graph with breadth-first search

- How to perform the intersection test?

Computational efficiency

- Running time $O(n^3)$
  - Compute the visibility graph
  - Search the graph
  - An optimal $O(n^2)$ time algorithm exists.
- Space $O(n^2)$

- **Can we do better?**

Classic path planning approaches

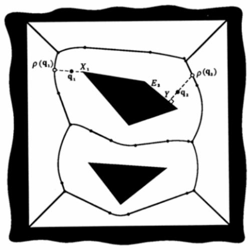
- **Roadmap**
  Represent the connectivity of the free space by a network of 1-D curves
- **Cell decomposition**
  Decompose the free space into simple cells and represent the connectivity of the free space by the adjacency graph of these cells
- **Potential field**
  Define a potential function over the free space that has a global minimum at the goal and follow the steepest descent of the potential function

Classic path planning approaches

- **Roadmap**
  Represent the connectivity of the free space by a network of 1-D curves
- **Cell decomposition**
  Decompose the free space into simple cells and represent the connectivity of the free space by the adjacency graph of these cells
- **Potential field**
  Define a potential function over the free space that has a global minimum at the goal and follow the steepest descent of the potential function

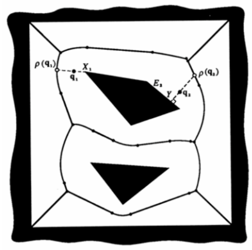
Roadmap

- Visibility graph
  Shakey Project, SRI [Nilsson, 1969]
- **Voronoi diagram**
  Introduced by computational geometry researchers. Generate paths that maximizes clearance.



Voronoi diagram method

- Space $O(n)$
- Running time $O(n \log n)$
- Applicable mostly to 2-D configuration spaces



Other roadmap methods

- Silhouette
  First complete general method that applies to spaces of any dimensions and is singly **exponential** in the number of dimensions [Canny, 87]
- **Probabilistic roadmaps**

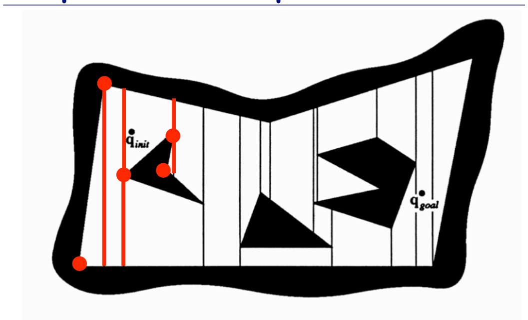
Classic path planning approaches

- **Roadmap**
  Represent the connectivity of the free space by a network of 1-D curves
- **Cell decomposition**
  Decompose the free space into **simple** cells and represent the connectivity of the free space by the adjacency graph of these cells
- **Potential field**
  Define a potential function over the free space that has a global minimum at the goal and follow the steepest descent of the potential function

Cell-decomposition methods

- **Exact cell decomposition**
  The free space F is represented by a collection of non-overlapping simple cells whose union **is exactly** F.
  - Examples of cells: trapezoids, triangles

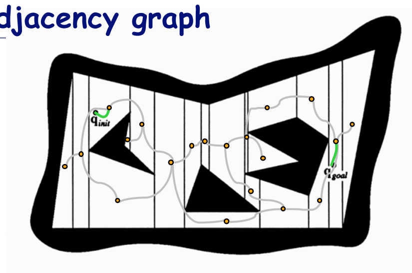
Trapezoidal decomposition



Computational efficiency

- Running time $O(n \log n)$ by planar sweep
- Space $O(n)$
- Mostly for 2-D configuration spaces

Adjacency graph



- **Nodes**: cells
- **Edges**: There is an edge between every pair of nodes whose corresponding cells are adjacent.

Framework

continuous representation

↓

discretization

construct an adjacency graph of the cells

↓

graph searching

search the adjacency graph

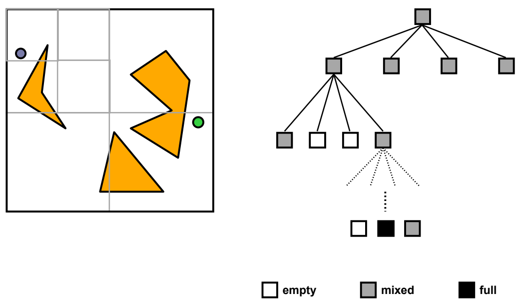
A brief summary

- Discretize the space by constructing an adjacency graph of the cells
- Search the adjacency graph

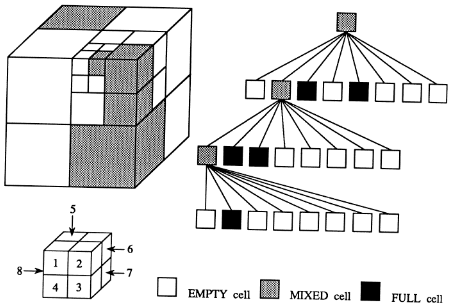
Cell-decomposition methods

- Exact cell decomposition
- **Approximate cell decomposition**
  The free space F is represented by a collection of non-overlapping cells whose union is **contained** in F.
  - Cells usually have simple, regular shapes, *e.g.*, rectangles, squares.
  - Facilitate hierarchical space decomposition

Quadtree decomposition



Octree decomposition



Sketch of the algorithm

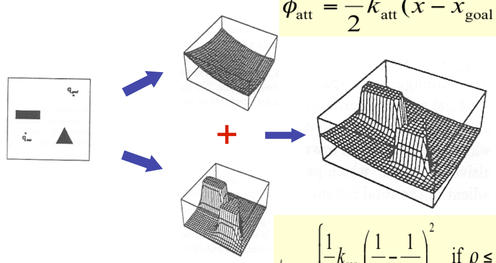
1. Decompose the free space F into cells.
2. Search for a sequence of **mixed** or **free** cells that connect the initial and goal positions.
3. Further decompose the mixed.
4. Repeat (2) and (3) until a sequence of **free** cells is found.

Classic path planning approaches

- **Roadmap**
  Represent the connectivity of the free space by a network of 1-D curves
- **Cell decomposition**
  Decompose the free space into **simple** cells and represent the connectivity of the free space by the adjacency graph of these cells
- **Potential field**
  Define a potential function over the free space that has a global minimum at the goal and follow the steepest descent of the potential function

Algorithm in pictures

$$\phi_{\text{att}} = \frac{1}{2} k_{\text{att}} (x - x_{\text{goal}})^2$$



$$\phi_{\text{rep}} = \begin{cases} \frac{1}{2} k_{\text{rep}} \left(\frac{1}{\rho} - \frac{1}{\rho_0} \right)^2 & \text{if } \rho \le \rho_0, \\ 0 & \text{if } \rho > \rho_0 \end{cases}$$

Attractive & repulsive fields

$$F_{\text{att}} = -\nabla \phi_{\text{att}} = -k_{\text{att}} (x - x_{\text{goal}})$$

$$F_{\text{rep}} = -\nabla \phi_{\text{rep}} = \begin{cases} k_{\text{rep}} \left(\frac{1}{\rho} - \frac{1}{\rho_0} \right) \frac{1}{\rho^2} \frac{\partial \rho}{\partial x} & \text{if } \rho \le \rho_0, \\ 0 & \text{if } \rho > \rho_0 \end{cases}$$

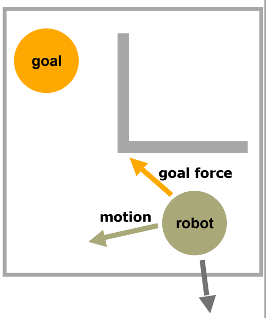
$k_{\text{att}}, k_{\text{rep}}$: positive scaling factors

x : position of the robot

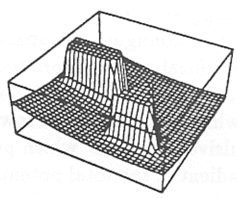
ρ : distance to the obstacle

ρ_0 : distance of influence

[Khatib, 1986]



Local minima



- What can we do?
  - Escape from local minima by taking random walks
  - Build an ideal potential field – navigation function – that does not have local minima

Sketch of the algorithm

- 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

Potential field

- Initially proposed for real-time collision avoidance [Khatib, 1986]. Hundreds of papers published on this topic.
- A potential field is a scalar function over the free space.
- To navigate, the 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

Question

- Can such an ideal potential field be constructed efficiently in general?

Completeness

- A **complete** **motion planner** always returns a solution when one exists and indicates that no such solution exists otherwise.
  - Is the visibility graph algorithm complete? Yes.
  - How about the exact cell decomposition algorithm and the potential field algorithm?