

Path and Motion Planning

Jan Faigl

Department of Computer Science

Faculty of Electrical Engineering

Czech Technical University in Prague

Lecture 03

B4M36UIR – Artificial Intelligence in Robotics



Overview of the Lecture

- Part 1 – Path and Motion Planning
 - Introduction to Path Planning
 - Notation and Terminology
 - Path Planning Methods



Part I

Part 1 – Path and Motion Planning



Outline

- Introduction to Path Planning
- Notation and Terminology
- Path Planning Methods



Robot Motion Planning – Motivational problem

- How to transform high-level task specification (provided by humans) into a low-level description suitable for controlling the actuators?

*To develop **algorithms** for such a transformation.*

The motion planning algorithms provide transformations how to move a robot (object) considering all operational constraints.

It encompasses several disciplines, e.g., mathematics, robotics, computer science, control theory, artificial intelligence, computational geometry, etc.

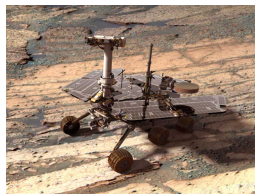


Robot Motion Planning – Motivational problem

- How to transform high-level task specification (provided by humans) into a low-level description suitable for controlling the actuators?

To develop *algorithms* for such a transformation.

The motion planning algorithms provide transformations how to move a robot (object) considering all operational constraints.



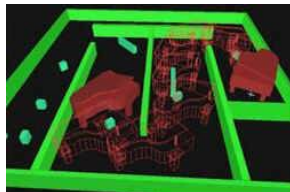
It encompasses several disciplines, e.g., mathematics, robotics, computer science, control theory, artificial intelligence, computational geometry, etc.



Piano Mover's Problem

A classical motion planning problem

Having a CAD model of the piano, model of the environment, the problem is how to move the piano from one place to another without hitting anything.



Basic motion planning algorithms are focused primarily on rotations and translations.

- We need **notion** of model representations and formal definition of the problem.
- Moreover, we also need a context about the problem and **realistic assumptions**.

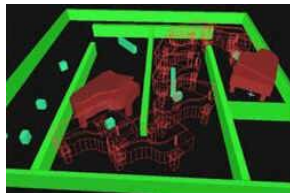
The plans have to be admissible and feasible.



Piano Mover's Problem

A classical motion planning problem

Having a CAD model of the piano, model of the environment, the problem is how to move the piano from one place to another without hitting anything.



Basic motion planning algorithms are focused primarily on rotations and translations.

- We need **notion** of model representations and formal definition of the problem.
- Moreover, we also need a context about the problem and **realistic assumptions**.

The plans have to be admissible and feasible.



Robotic Planning Context

Mission Planning

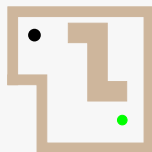
Tasks and Actions Plans

symbol level



Motion Planning

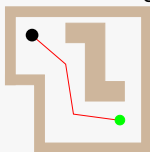
Problem



"geometric" level

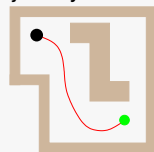
Models of
robot and
workspace

Path Planning



Path

Trajectory Planning



Trajectory



Robot Control

Sensing and Acting

"physical" level

feedback control
controller – drives (motors) – sensors



Robotic Planning Context

Mission Planning

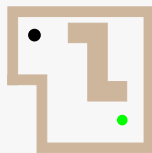
Tasks and Actions Plans

symbol level



Motion Planning

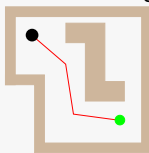
Problem



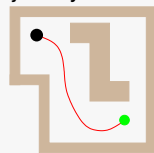
"geometric" level

Models of
robot and
workspace

Path Planning



Trajectory Planning



Trajectory

Open-loop control?



Robot Control

Sensing and Acting

"physical" level

feedback control
controller – drives (motors) – sensors



Robotic Planning Context

Mission Planning

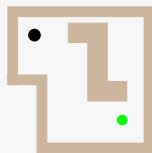
Tasks and Actions Plans

symbol level



Motion Planning

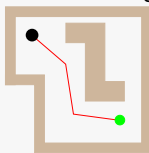
Problem



"geometric" level

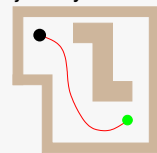
Models of
robot and
workspace

Path Planning



Path

Trajectory Planning



Trajectory

Open-loop control?



Robot Control

Sensing and Acting

"physical" level

feedback control
controller – drives (motors) – sensors

Sources of uncertainties
because of real environment



Robotic Planning Context

Mission Planning

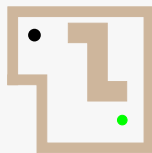
Tasks and Actions Plans

symbol level



Motion Planning

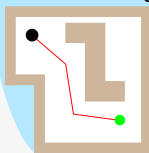
Problem



"geometric" level

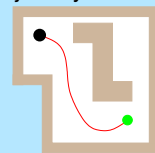
Models of
robot and
workspace

Path Planning



Path

Trajectory Planning



Trajectory

Open-loop control?



Robot Control

Sensing and Acting

"physical" level

feedback control
controller – drives (motors) – sensors

Sources of uncertainties
because of real environment



Robotic Planning Context

Mission Planning

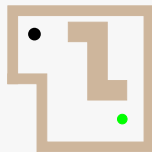
Tasks and Actions Plans

symbol level



Motion Planning

Problem

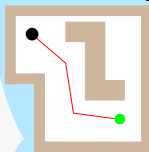


"geometric" level

*Models of
robot and
workspace*

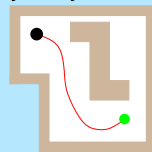


Path Planning



Path

Trajectory Planning



Trajectory

Open-loop control?



Robot Control

"physical" level

Sensing and Acting

*feedback control
controller – drives (motors) – sensors*

*Sources of uncertainties
because of real environment*



Real Mobile Robots

In a real deployment, the problem is a more complex.

- The world is changing
- Robots update the knowledge about the environment
localization, mapping and navigation
- New decisions have to be made
- A feedback from the environment
Motion planning is a part of the mission replanning loop.



Josef Štrunc, Bachelor thesis, CTU, 2009.

An example of **robotic mission**:

Multi-robot exploration of unknown environment

How to deal with real-world complexity?

Relaxing constraints and considering realistic assumptions.



Real Mobile Robots

In a real deployment, the problem is a more complex.

- The world is changing
- Robots update the knowledge about the environment
localization, mapping and navigation
- New decisions have to be made
- A feedback from the environment
Motion planning is a part of the mission replanning loop.



Josef Štrunc, Bachelor thesis, CTU, 2009.

An example of **robotic mission**:

Multi-robot exploration of unknown environment

How to deal with real-world complexity?

Relaxing constraints and considering realistic assumptions.



Real Mobile Robots

In a real deployment, the problem is a more complex.

- The world is changing
- Robots update the knowledge about the environment
localization, mapping and navigation
- New decisions have to be made
- A feedback from the environment
Motion planning is a part of the mission replanning loop.



Josef Štrunc, Bachelor thesis, CTU, 2009.

An example of **robotic mission**:

Multi-robot exploration of unknown environment

How to deal with real-world complexity?

Relaxing constraints and considering realistic assumptions.



Outline

- Introduction to Path Planning
- Notation and Terminology
- Path Planning Methods



Notation

- \mathcal{W} – **World model** describes the robot workspace and its boundary determines the obstacles \mathcal{O}_i .

2D world, $\mathcal{W} = \mathbb{R}^2$

- A **Robot** is defined by its geometry, parameters (kinematics) and it is controllable by the motion plan.

- \mathcal{C} – **Configuration space (\mathcal{C} -space)**

A concept to describe possible configurations of the robot. The robot's **configuration** completely specify the robot location in \mathcal{W} including specification of all degrees of freedom.

E.g., a robot with rigid body in a plane $\mathcal{C} = \{x, y, \varphi\} = \mathbb{R}^2 \times S^1$.

- Let \mathcal{A} be a subset of \mathcal{W} occupied by the robot, $\mathcal{A} = \mathcal{A}(q)$.
- A subset of \mathcal{C} occupied by obstacles is

$$\mathcal{C}_{obs} = \{q \in \mathcal{C} : \mathcal{A}(q) \cap \mathcal{O}_i, \forall i\}$$

- **Collision-free configurations** are

$$\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}.$$



Notation

- \mathcal{W} – **World model** describes the robot workspace and its boundary determines the obstacles \mathcal{O}_i .

2D world, $\mathcal{W} = \mathbb{R}^2$

- A **Robot** is defined by its geometry, parameters (kinematics) and it is controllable by the motion plan.

- \mathcal{C} – **Configuration space (\mathcal{C} -space)**

A concept to describe possible configurations of the robot. The robot's **configuration** completely specify the robot location in \mathcal{W} including specification of all degrees of freedom.

E.g., a robot with rigid body in a plane $\mathcal{C} = \{x, y, \varphi\} = \mathbb{R}^2 \times S^1$.

- Let \mathcal{A} be a subset of \mathcal{W} occupied by the robot, $\mathcal{A} = \mathcal{A}(q)$.
- A subset of \mathcal{C} occupied by obstacles is

$$\mathcal{C}_{obs} = \{q \in \mathcal{C} : \mathcal{A}(q) \cap \mathcal{O}_i, \forall i\}$$

- **Collision-free configurations** are

$$\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}.$$



Notation

- \mathcal{W} – **World model** describes the robot workspace and its boundary determines the obstacles \mathcal{O}_i .

2D world, $\mathcal{W} = \mathbb{R}^2$

- A **Robot** is defined by its geometry, parameters (kinematics) and it is controllable by the motion plan.

- \mathcal{C} – **Configuration space (\mathcal{C} -space)**

A concept to describe possible configurations of the robot. The robot's **configuration** completely specify the robot location in \mathcal{W} including specification of all degrees of freedom.

E.g., a robot with rigid body in a plane $\mathcal{C} = \{x, y, \varphi\} = \mathbb{R}^2 \times S^1$.

- Let \mathcal{A} be a subset of \mathcal{W} occupied by the robot, $\mathcal{A} = \mathcal{A}(q)$.
- A subset of \mathcal{C} occupied by obstacles is

$$\mathcal{C}_{obs} = \{q \in \mathcal{C} : \mathcal{A}(q) \cap \mathcal{O}_i, \forall i\}$$

- **Collision-free configurations** are

$$\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}.$$



Notation

- \mathcal{W} – **World model** describes the robot workspace and its boundary determines the obstacles \mathcal{O}_i .

2D world, $\mathcal{W} = \mathbb{R}^2$

- A **Robot** is defined by its geometry, parameters (kinematics) and it is controllable by the motion plan.

- \mathcal{C} – **Configuration space (\mathcal{C} -space)**

A concept to describe possible configurations of the robot. The robot's **configuration** completely specify the robot location in \mathcal{W} including specification of all degrees of freedom.

E.g., a robot with rigid body in a plane $\mathcal{C} = \{x, y, \varphi\} = \mathbb{R}^2 \times S^1$.

- Let \mathcal{A} be a subset of \mathcal{W} occupied by the robot, $\mathcal{A} = \mathcal{A}(q)$.
- A subset of \mathcal{C} occupied by obstacles is

$$\mathcal{C}_{obs} = \{q \in \mathcal{C} : \mathcal{A}(q) \cap \mathcal{O}_i, \forall i\}$$

- **Collision-free configurations** are

$$\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}.$$



Notation

- \mathcal{W} – **World model** describes the robot workspace and its boundary determines the obstacles \mathcal{O}_i .

2D world, $\mathcal{W} = \mathbb{R}^2$

- A **Robot** is defined by its geometry, parameters (kinematics) and it is controllable by the motion plan.
- \mathcal{C} – **Configuration space** (**C-space**)

A concept to describe possible configurations of the robot. The robot's **configuration** completely specify the robot location in \mathcal{W} including specification of all degrees of freedom.

E.g., a robot with rigid body in a plane $\mathcal{C} = \{x, y, \varphi\} = \mathbb{R}^2 \times S^1$.

- Let \mathcal{A} be a subset of \mathcal{W} occupied by the robot, $\mathcal{A} = \mathcal{A}(q)$.
- A subset of \mathcal{C} occupied by obstacles is

$$\mathcal{C}_{obs} = \{q \in \mathcal{C} : \mathcal{A}(q) \cap \mathcal{O}_i, \forall i\}$$

- **Collision-free configurations** are

$$\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}.$$



Path / Motion Planning Problem

- **Path** is a continuous mapping in \mathcal{C} -space such that

$$\pi : [0, 1] \rightarrow \mathcal{C}_{free}, \text{ with } \pi(0) = q_0, \text{ and } \pi(1) = q_f,$$

Only geometric considerations

- **Trajectory** is a path with explicate parametrization of time, e.g., accompanied by a description of the motion laws ($\gamma : [0, 1] \rightarrow \mathcal{U}$, where \mathcal{U} is robot's action space).

It includes dynamics.

$$[T_0, T_f] \ni t \rightsquigarrow \tau \in [0, 1] : q(t) = \pi(\tau) \in \mathcal{C}_{free}$$

The planning problem is determination of the function $\pi(\cdot)$.

Additional requirements can be given:

- Smoothness of the path
- Kinodynamic constraints
- Optimality criterion

E.g., considering friction forces

shortest vs fastest (length vs curvature)



Path / Motion Planning Problem

- **Path** is a continuous mapping in \mathcal{C} -space such that

$$\pi : [0, 1] \rightarrow \mathcal{C}_{free}, \text{ with } \pi(0) = q_0, \text{ and } \pi(1) = q_f,$$

Only geometric considerations

- **Trajectory** is a path with explicate parametrization of time, e.g., accompanied by a description of the motion laws ($\gamma : [0, 1] \rightarrow \mathcal{U}$, where \mathcal{U} is robot's action space).

It includes dynamics.

$$[T_0, T_f] \ni t \rightsquigarrow \tau \in [0, 1] : q(t) = \pi(\tau) \in \mathcal{C}_{free}$$

The planning problem is determination of the function $\pi(\cdot)$.

Additional requirements can be given:

- Smoothness of the path
- Kinodynamic constraints
- Optimality criterion

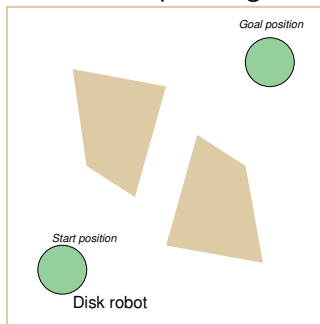
E.g., considering friction forces

shortest vs fastest (length vs curvature)

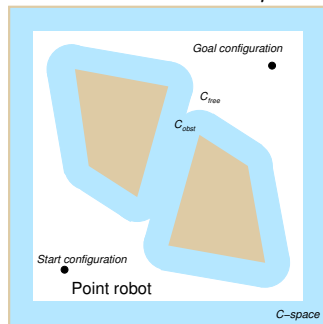


Planning in \mathcal{C} -space

Robot motion planning robot for a disk robot with a radius ρ .



Motion planning problem in geometrical representation of \mathcal{W}



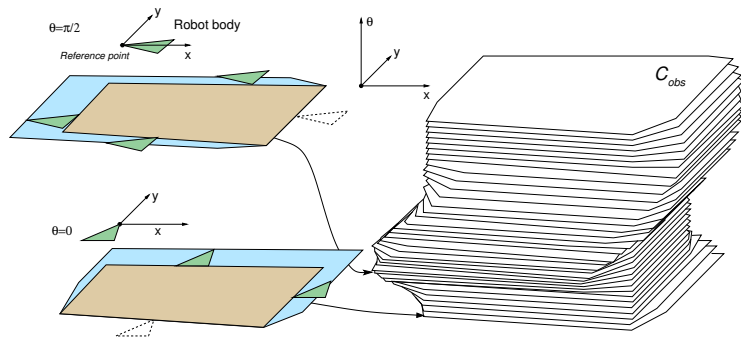
Motion planning problem in \mathcal{C} -space representation

\mathcal{C} -space has been obtained by enlarging obstacles by the disk \mathcal{A} with the radius ρ .

By applying Minkowski sum: $\mathcal{O} \oplus \mathcal{A} = \{x + y \mid x \in \mathcal{O}, y \in \mathcal{A}\}$.



Example of C_{obs} for a Robot with Rotation



A simple 2D obstacle \rightarrow has a complicated C_{obs}

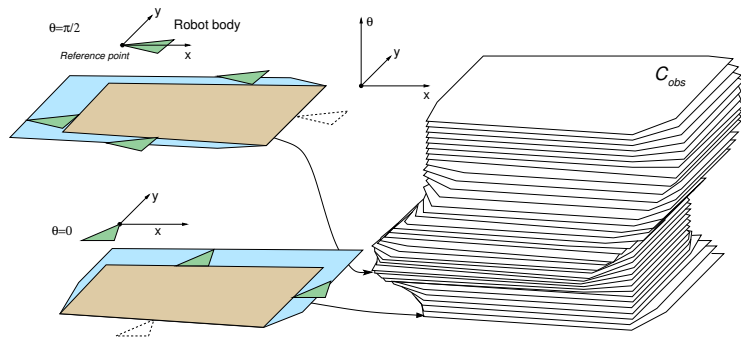
- Deterministic algorithms exist

*Requires exponential time in C dimension,
J. Canny, PAMI, 8(2):200–209, 1986*

- Explicit representation of C_{free} is impractical to compute.



Example of \mathcal{C}_{obs} for a Robot with Rotation



A simple 2D obstacle \rightarrow has a complicated \mathcal{C}_{obs}

- Deterministic algorithms exist

Requires exponential time in \mathcal{C} dimension,

J. Canny, PAMI, 8(2):200–209, 1986

- Explicit representation of \mathcal{C}_{free} is impractical to compute.



Representation of \mathcal{C} -space

How to deal with continuous representation of \mathcal{C} -space?

Continuous Representation of \mathcal{C} -space



Discretization

processing critical geometric events, (random) sampling
roadmaps, cell decomposition, potential field



Graph Search Techniques
BFS, Gradient Search, A*



Representation of \mathcal{C} -space

How to deal with continuous representation of \mathcal{C} -space?

Continuous Representation of \mathcal{C} -space



Discretization

processing critical geometric events, (random) sampling
roadmaps, cell decomposition, potential field



Graph Search Techniques

BFS, Gradient Search, A*



Outline

- Introduction to Path Planning
- Notation and Terminology
- **Path Planning Methods**



Planning Methods - Overview

(selected approaches)

■ Roadmap based methods

Create a connectivity graph of the free space.

- Visibility graph

(complete but impractical)

- Cell decomposition
- Voronoi graph

- Discretization into a **grid-based** (or lattice-based) representation
(resolution complete)

- **Potential field methods** *(complete only for a "navigation function", which is hard to compute in general)*

Classic path planning algorithms

■ Randomized sampling-based methods

- Creates a roadmap from connected random samples in \mathcal{C}_{free}
- Probabilistic roadmaps

samples are drawn from some distribution

- Very successful in practice



Planning Methods - Overview

(selected approaches)

■ Roadmap based methods

Create a connectivity graph of the free space.

- Visibility graph

(complete but impractical)

- Cell decomposition
- Voronoi graph

- Discretization into a **grid-based** (or lattice-based) representation
(resolution complete)

- **Potential field methods** *(complete only for a "navigation function", which is hard to compute in general)*

Classic path planning algorithms

■ Randomized sampling-based methods

- Creates a roadmap from connected random samples in \mathcal{C}_{free}
- Probabilistic roadmaps

samples are drawn from some distribution

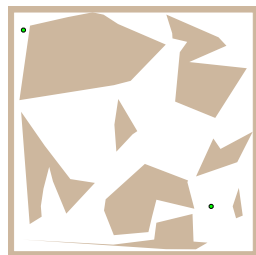
- Very successful in practice



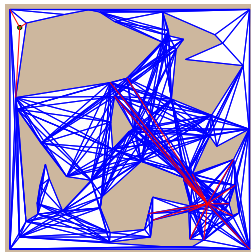
Visibility Graph

1. Compute visibility graph
2. Find the shortest path

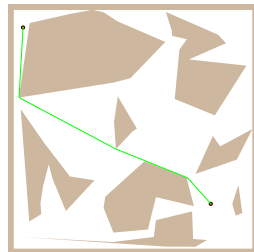
E.g., by Dijkstra's algorithm



Problem



Visibility graph



Found shortest path

Constructions of the visibility graph:

- Naïve – all segments between n vertices of the map $O(n^3)$
- Using rotation trees for a set of segments – $O(n^2)$

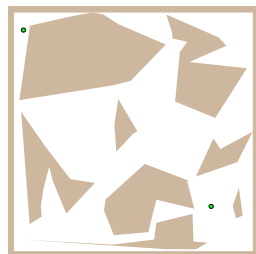
M. H. Overmars and E. Welzl, 1988



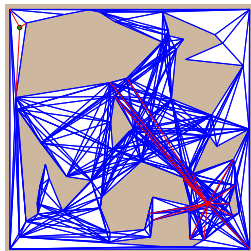
Visibility Graph

1. Compute visibility graph
2. Find the shortest path

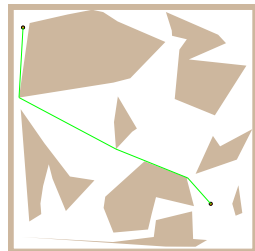
E.g., by Dijkstra's algorithm



Problem



Visibility graph



Found shortest path

Constructions of the visibility graph:

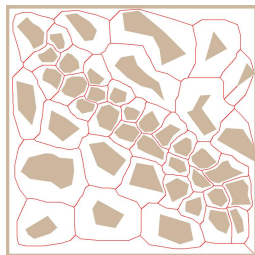
- Naïve – all segments between n vertices of the map $O(n^3)$
- Using rotation trees for a set of segments – $O(n^2)$

M. H. Overmars and E. Welzl, 1988

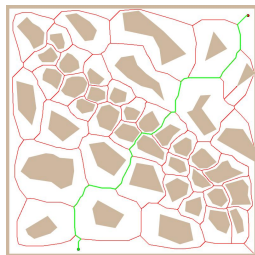


Voronoi Graph

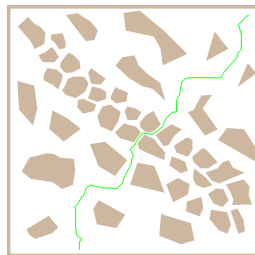
1. Roadmap is Voronoi graph that **maximizes clearance** from the obstacles
2. Start and goal positions are connected to the graph
3. Path is found using a graph search algorithm



Voronoi graph



Path in graph



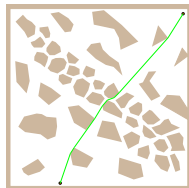
Found path



Visibility Graph vs Voronoi Graph

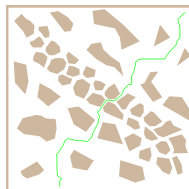
Visibility graph

- Shortest path, but it is close to obstacles. We have to consider safety of the path
An error in plan execution can lead to a collision.
- Complicated in higher dimensions



Voronoi graph

- It maximizes clearance, which can provide conservative paths
- Small changes in obstacles can lead to large changes in the graph
- Complicated in higher dimensions



A combination is called Visibility-Voronoi – R. Wein, J. P. van den Berg, D. Halperin, 2004

For higher dimensions we need other roadmaps.



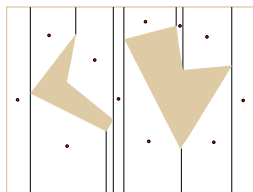
Cell Decomposition

1. Decompose free space into parts.

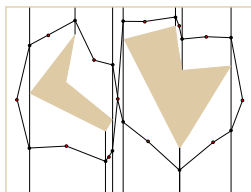
Any two points in a convex region can be directly connected by a segment.

2. Create an adjacency graph representing the connectivity of the free space.
3. Find a path in the graph.

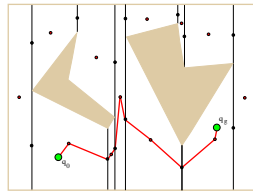
Trapezoidal decomposition



Centroids represent cells



Connect adjacency cells



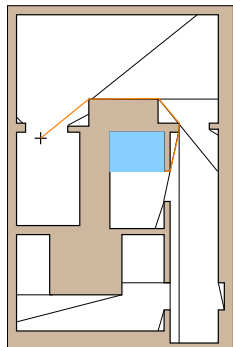
Find path in the adjacency graph

Other decomposition (e.g., triangulation) are possible.



Shortest Path Map (SPM)

- Speedup computation of the shortest path towards a particular goal location p_g for a polygonal domain \mathcal{P} with n vertices
- A partitioning of the free space into cells with respect to the particular location p_g
- Each cell has a vertex on the shortest path to p_g
- Shortest path from any point p is found by determining the cell (in $O(\log n)$ using point location alg.) and then traversing the shortest path with up to k bends, i.e., it is found in $O(\log n + k)$
- Determining the SPM using “wavefront” propagation based on *continuous Dijkstra paradigm*



Joseph S. B. Mitchell: *A new algorithm for shortest paths among obstacles in the plane*, *Annals of Mathematics and Artificial Intelligence*, 3(1):83–105, 1991.

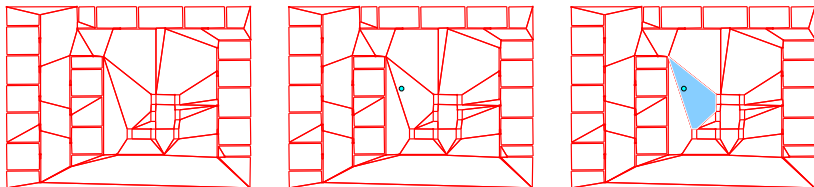
- SPM is a precompute structure for the given \mathcal{P} and p_g
 - single-point query

A similar structure can be found for two-point query, e.g., H. Guo, A. Maheshwari, J.-R. Sack, 2008



Point Location Problem

- For a given partitioning of the polygonal domain into a discrete set of cells, determine the cell for a given point p



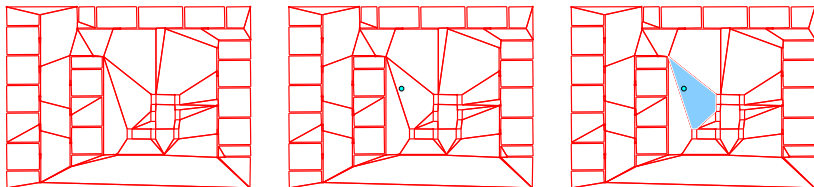
Masato Edahiro, Iwao Kokubo and Takao Asano: *A new point-location algorithm and its practical efficiency: comparison with existing algorithms*, *ACM Trans. Graph.*, 3(2):86–109, 1984.

- It can be implemented using **interval trees** – slabs and slices



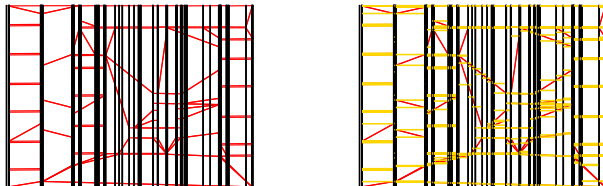
Point Location Problem

- For a given partitioning of the polygonal domain into a discrete set of cells, determine the cell for a given point p



Masato Edahiro, Iwao Kokubo and Takao Asano: *A new point-location algorithm and its practical efficiency: comparison with existing algorithms*, *ACM Trans. Graph.*, 3(2):86–109, 1984.

- It can be implemented using **interval trees** – slabs and slices

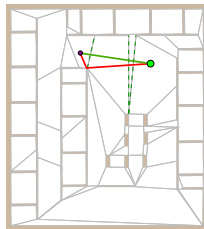
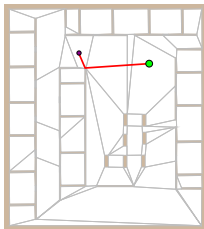


Point location problem, SPM and similarly problems are from the **Computational Geometry** field



Approximate Shortest Path and Navigation Mesh

- We can use any convex partitioning of the polygonal map to speed up shortest path queries
 - Precompute all shortest paths from map vertices to p_g using visibility graph
 - Then, an estimation of the shortest path from p to p_g is the shortest path among the one of the cell vertex



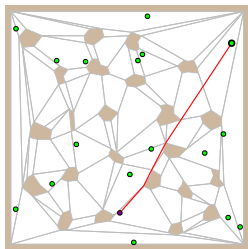
- The estimation can be further improve by “ray-shooting” technique combined with walking in triangulation (convex partitioning)

(Faigl, 2010)

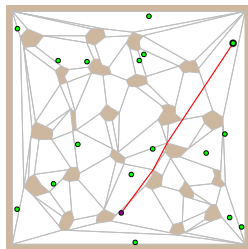


Path Refinement

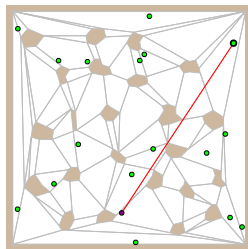
- Testing collision of the point p with particular vertices of the estimation of the shortest path
 - Let the initial path estimation from p to p_g be a sequence of k vertices $(p, v_1, \dots, v_k, p_g)$
 - We can iteratively test if the segment (p, v_i) , $1 < i \leq k$ is collision free up to (p, p_g)



path over v_0



path over v_1



full refinement

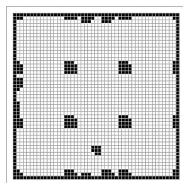
With precomputed structures, it allows to estimate the shortest path in units of microseconds



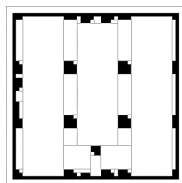
Navigation Mesh

- In addition to robotic approaches, fast shortest path queries are studied in computer games
- There is a class of algorithms based on navigation mesh
 - A supporting structure representing the free space

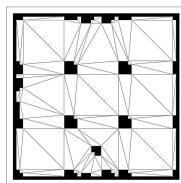
*It usually originated from the grid based maps, but it is represented as **CDT – Constrained Delaunay triangulation***



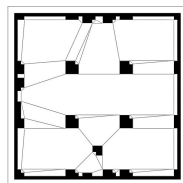
Grid mesh



Merged grid mesh



CDT mesh



Merged CDT mesh

- E.g., **Polyanya** algorithm based on navigation mesh and best-first search
 M. Cui, D. Harabor, A. Grastien: *Compromise-free Pathfinding on a Navigation Mesh*,
IJCAI 2017, 496–502.

<https://bitbucket.org/dharabor/pathfinding>

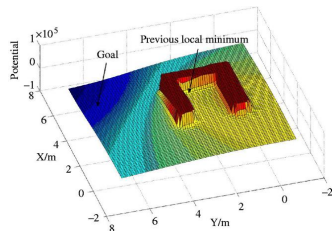
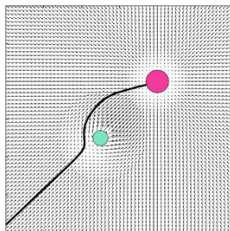
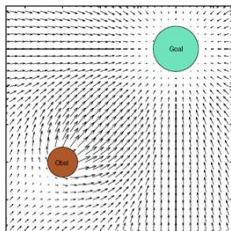
Informative



Artificial Potential Field Method

- The idea is to create a function f that will provide a direction towards the goal for any configuration of the robot.
- Such a function is called **navigation function** and $-\nabla f(q)$ points to the goal.
- Create a **potential field** that will **attract robot towards the goal** q_f while obstacles will generate **repulsive potential** repelling the robot away from the obstacles.

The navigation function is a sum of potentials.



Such a potential function can have several local minima.



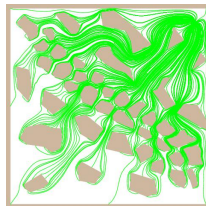
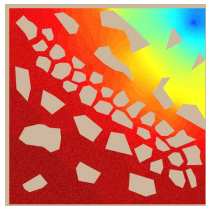
Avoiding Local Minima in Artificial Potential Field

- Consider harmonic functions that have only one extremum

$$\nabla^2 f(q) = 0$$

- Finite element method

Dirichlet and Neumann boundary conditions



J. Mačák, Master thesis, CTU, 2009



Summary of the Lecture



Topics Discussed

- Motion planning problem
- Path planning methods – overview
- Notation of configuration space
- Shortest-Path Roadmaps
- Voronoi diagram based planning
- Cell decomposition method
- Artificial potential field method

- Next: Grid-based path planning



Topics Discussed

- Motion planning problem
- Path planning methods – overview
- Notation of configuration space
- Shortest-Path Roadmaps
- Voronoi diagram based planning
- Cell decomposition method
- Artificial potential field method

- Next: Grid-based path planning

