Randomized Sampling-based Motion Planning Techniques

Jan Faigl

Department of Computer Science

Faculty of Electrical Engineering

Czech Technical University in Prague (CTU)

July 8, 2014

Presented by V. Hlavac to A3M33IRO in a shortened version on May 2, 2016.

Outline

1. Motion planning and overview of techniques
2. Randomized sampling-based algorithms
3. Optimal motion planners
4. ~~Motion planning in robotic missions~~
    - ~~Multi-goal planning~~
    - ~~Autonomous data collection planning~~

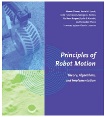
Part I

Introduction to Motion Planning

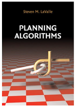
Literature

- Robot Motion Planning, *Jean-Claude Latombe,* Kluwer Academic Publishers, Boston, MA, 1991.

- Principles of Robot Motion: Theory, Algorithms, and Implementations, *H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki and S. Thrun,* MIT Press, Boston, 2005.
  http://biorobotics.ri.cmu.edu/book



- Planning Algorithms, *Steven M. LaValle,* Cambridge University Press, May 29, 2006.
  http://planning.cs.uiuc.edu



- Robot Motion Planning and Control, *Jean-Paul Laumond,* Lectures Notes in Control and Information Sciences, 2009.
  http://homepages.laas.fr/jpl/book.html

- Sampling-based algorithms for optimal motion planning, *Sertac Karaman, Emilio Frazzoli,* International Journal of Robotic Research, 30(7):846–894, 2011.
  http://sertac.scripts.mit.edu/rrtstar

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 disciples, e.g., mathematics, robotics, computer science, control theory, artificial intelligence, computational geometry, etc.

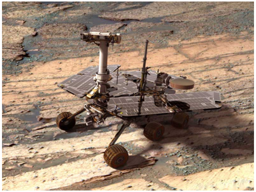
Motion Planning

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.

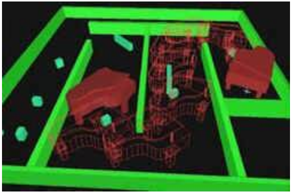


It encompasses several disciples, e.g., mathematics, robotics, computer science, control theory, artificial intelligence, computational geometry, etc.

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.

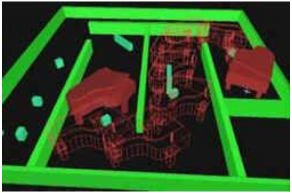
- We need a notion of model representations and formal definition of the problem.
- Moreover, we also need a context about the problem and realistic assumptions.

The plans have to be feasible and admissible.

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 a 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 feasible and admissible.

Robotic Planning Context

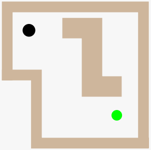
Mission Planning

Tasks and Actions Plans

symbol level

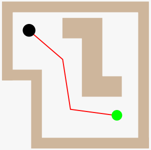
Motion Planning

Problem



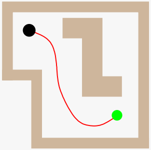
Models of robot and workspace

Path Planning



Path

Trajectory Planning



"geometric" level

Trajectory

Robot Control

Sensing and Acting

feedback control

controller – drives (motors) – sensors

"physical" level

Robotic Planning Context

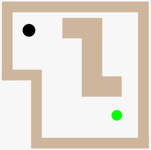
Mission Planning

Tasks and Actions Plans

symbol level

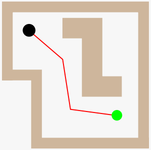
Motion Planning

Problem



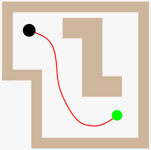
Models of robot and workspace

Path Planning



Path

Trajectory Planning



"geometric" level

Trajectory

Open–loop control?

Robot Control

Sensing and Acting

feedback control

controller – drives (motors) – sensors

"physical" level

Robotic Planning Context

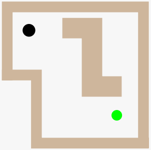
Mission Planning

Tasks and Actions Plans

symbol level

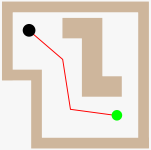
Motion Planning

Problem



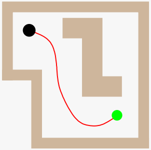
Models of robot and workspace

Path Planning



Path

Trajectory Planning



"geometric" level

Trajectory

Open–loop control?

Robot Control

Sensing and Acting

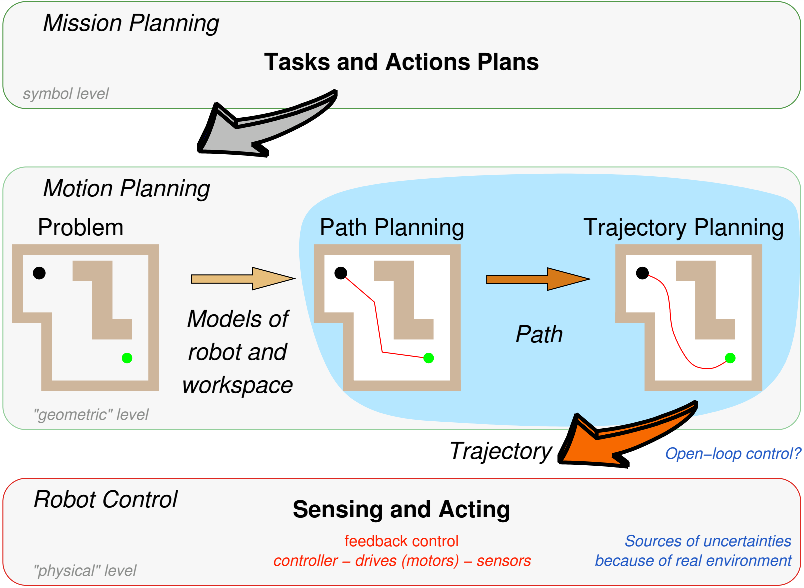
feedback control

controller – drives (motors) – sensors

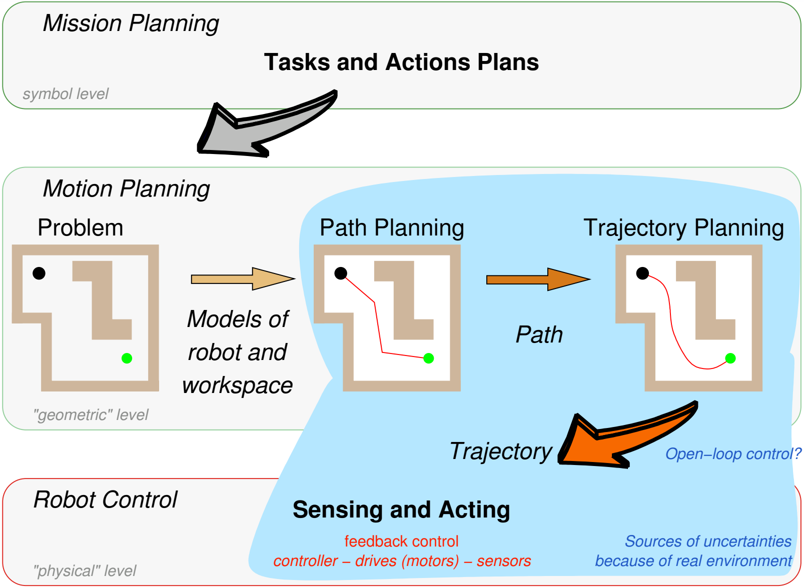
Sources of uncertainties because of real environment

"physical" level

Robotic Planning Context



Robotic Planning Context



Notation of the Configuration Space

- $\mathcal{W}$ – **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 $\mathcal{C} = \{x, y, \varphi\} = \mathbb{R}^2 \times SO(2)$.

  - 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

    $$\mathcal{C}_{free} = \mathrm{cl}(\mathcal{C} \setminus \mathcal{C}_{obs}).$$

    cl – *closure of a set*

Notation of the Configuration Space

- $\mathcal{W}$ – **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 $\mathcal{C} = \{x, y, \varphi\} = \mathbb{R}^2 \times SO(2)$.
  - 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

    $$\mathcal{C}_{free} = \mathrm{cl}(\mathcal{C} \setminus \mathcal{C}_{obs}).$$

    cl – *closure of a set*

Notation of the Configuration Space

- $\mathcal{W}$ – **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 $\mathcal{C} = \{x, y, \varphi\} = \mathbb{R}^2 \times SO(2)$.

  - 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

  $$\mathcal{C}_{free} = \mathrm{cl}(\mathcal{C} \setminus \mathcal{C}_{obs}).$$

  cl – *closure of a set*

Notation of the Configuration Space

- $\mathcal{W}$ – **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 $\mathcal{C} = \{x, y, \varphi\} = \mathbb{R}^2 \times SO(2)$.
  - 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

$$\mathcal{C}_{free} = \text{cl}(\mathcal{C} \setminus \mathcal{C}_{obs}).$$

cl – *closure of a set*

Notation of the Configuration Space

- $\mathcal{W}$ – **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 $\mathcal{C} = \{x, y, \varphi\} = \mathbb{R}^2 \times SO(2)$.

  - 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

$$\mathcal{C}_{free} = \mathrm{cl}(\mathcal{C} \setminus \mathcal{C}_{obs}).$$

cl – *closure of a set*

Path and Trajectory

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

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

  where q_0 is the initial and q_f the final robot configurations.

  Only geometric considerations

- **Trajectory** is a path with explicit parametrization of the robot motion, e.g.,
  - accompanied by a description of the motion laws

$$\gamma : [0,1] \to \mathcal{U},$$

    where $\mathcal{U}$ is the robot's action space.

    It includes dynamics.

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

Motion Planning Problem

Having

- a dynamical system with the state x and control u

$$\frac{dx}{dt} = f(x, u),$$

- set of obstacles $X_{obs} \subset \mathbb{R}^d$ and goal set $X_{goal} \subset \mathbb{R}^d$

the motion planning problem is to find control signal u such that $x(t) \notin X_{obs}$ for $t \in \mathbb{R}_+$ and $x(t) \in X_{goal}$ for all $t > T_f$ for some finite $T_f \geq 0$. Or, return no such control signal exists.

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

Additional requirements can be given:

- Smoothness of the path
- Kinodynamic constraints

  E.g., considering friction forces
- Optimality criterion

  shortest vs fastest (length vs curvature)

Motion Planning Problem

Having

- a dynamical system with the state x and control u

$$\frac{dx}{dt} = f(x, u),$$

- set of obstacles $X_{obs} \subset \mathbb{R}^d$ and goal set $X_{goal} \subset \mathbb{R}^d$

the motion planning problem is to find control signal u such that $x(t) \notin X_{obs}$ for $t \in \mathbb{R}_+$ and $x(t) \in X_{goal}$ for all $t > T_f$ for some finite $T_f \geq 0$. Or, return no such control signal exists.

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

Additional requirements can be given:

- Smoothness of the path
- Kinodynamic constraints

  E.g., considering friction forces
- Optimality criterion

  shortest vs fastest (length vs curvature)

Motion Planning Approaches

- Generalized piano mover's problem is PSPACE-hard

  Reif, 1979

- Complete algorithms exists, but are too complex to be practical
- The research has been focused on approximation algorithms

  trade full completeness of the planner for efficiency

- Full completeness vs resolution completeness

  returns valid solution (if exists) if the resolution parameter is fine enough

- Most successful approaches
  - Cell decomposition methods
  - Randomized sampling based planners (PRM, RRT)

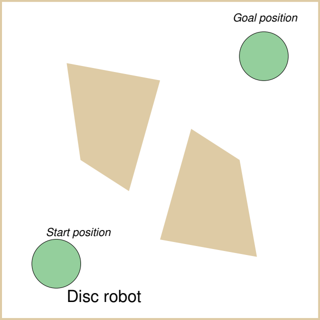
    sacrifice optimality for a feasibility and computational efficiency

  - Probabilistic optimal sampling based planners (RRG)

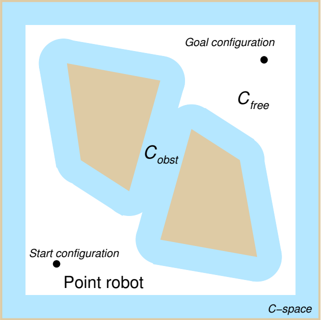
    Karaman and Frazzoli, 2011

Example of Simple Planning in $\mathcal{C}$-space

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



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

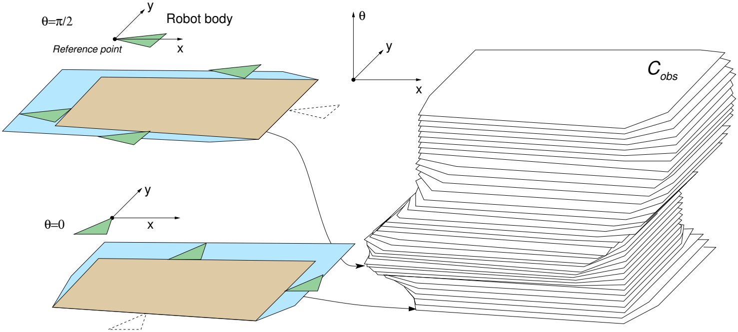


Motion planning problem in $\mathcal{C}$-space representation

$\mathcal{C}$-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 $\mathcal{C}_{obs}$ for a Robot with Rotation



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

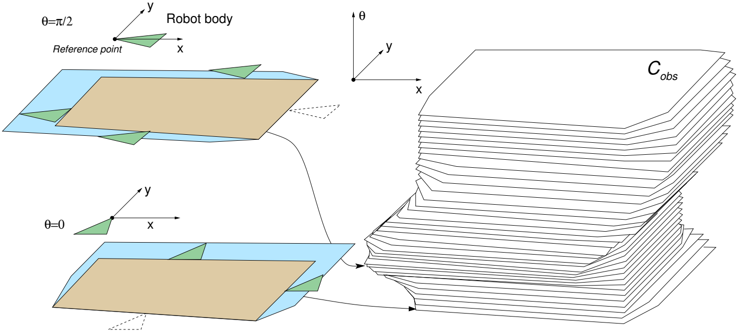
- Deterministic algorithms exist

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.

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



A simple 2D obstacle → 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*

Planning Methods Overview

(selected approaches)

- Roadmap based methods

  Create a connectivity graph of the free space.

  - Visibility graph

    (complete but impractical)

  - Cell decomposition
  - Voronoi diagram

- Discretization into a grid-based representation

  (resolution complete)

- Potential field methods

  (complete only for a "navigation function", which is hard to compute in general)

- **Sampling-based approaches**
  - 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 diagram

- Discretization into a grid-based representation

  (resolution complete)

- Potential field methods

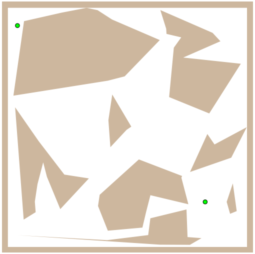
  (complete only for a "navigation function", which is hard to compute in general)

- **Sampling-based approaches**
  - Creates a roadmap from connected random samples in $\mathcal{C}_{free}$
  - Probabilistic roadmaps

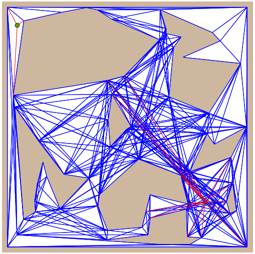
    samples are drawn from some distribution

  - Very successful in practice

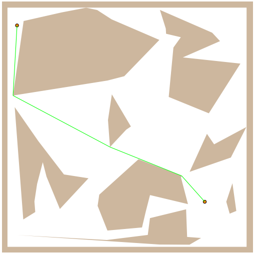
Simple Roadmap Construction – Visibility Graph



Problem



Visibility graph



Found shortest path

- Shortest path is found in the created visibility graph

  E.g., by Dijkstra's algorithm

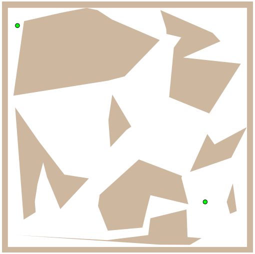
- Constructions of the visibility graph can be done in $O(n^3)$ or in $O(n^2)$ using rotation trees for a set of segments

  M. H. Overmars and E. Welzl, 1988

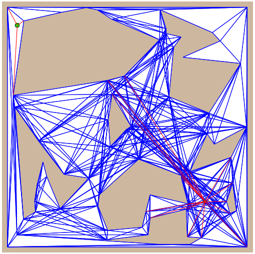
- Can be used for enlarged obstacles and a point robot

  However, it is not practical for complex robots

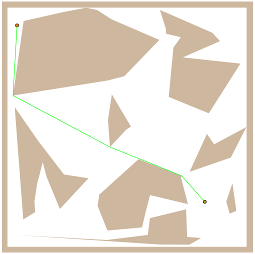
Simple Roadmap Construction – Visibility Graph



Problem



Visibility graph



Found shortest path

- Shortest path is found in the created visibility graph

  E.g., by Dijkstra's algorithm

- Constructions of the visibility graph can be done in $O(n^3)$ or in $O(n^2)$ using rotation trees for a set of segments

  M. H. Overmars and E. Welzl, 1988

- Can be used for enlarged obstacles and a point robot

  However, it is not practical for complex robots

Part II

Randomized Sampling-based Motion Planning Algorithms

Sampling-based Motion Planning

- Avoids explicit representation of the obstacles in $\mathcal{C}$-*space*
  - A "black-box" function is used to evaluate a configuration q is a collision free

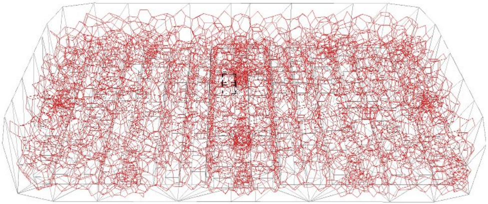
    (E.g., based on geometrical models and testing collisions of the models)
- It creates a discrete representation of $\mathcal{C}_{free}$
- Configurations in $\mathcal{C}_{free}$ are sampled randomly and connected to a roadmap (**probabilistic roadmap**)
- Rather than full completeness they provides **probabilistic completeness** or resolution completeness

  Probabilistic complete algorithms: with increasing number of samples an admissible solution would be found (if exists)

Probabilistic Roadmap

A discrete representation of the continuous $\mathcal{C}$-space generated by randomly sampled configurations in $\mathcal{C}_{free}$ that are connected into a graph.

- **Nodes** of the graph represent admissible configuration of the robot.
- **Edges** represent a feasible path (trajectory) between the particular configurations.



Having the graph, the final path (trajectory) is found by a graph search technique.

Probabilistic Roadmap Strategies

Multi-Query (Batch)

- Generate a single roadmap that is then used for planning queries several times.
  - Probabilistic RoadMap – PRM

Kavraki and Latombe, 1996

Single-Query (**Incremental**)

- For each planning problem constructs a new roadmap to characterize the subspace of $\mathcal{C}$-space that is relevant to the problem.
  - Rapidly-exploring Random Tree – RRT

    LaValle, 1998
  - Expansive-Space Tree – EST

    Hsu et al., 1997
  - Sampling-based Roadmap of Trees – SRT

    (combination of multiple–query and single–query approaches)

    Plaku et al., 2005

Probabilistic RoadMap (PRM) Planner

Build a roadmap (graph) representing the environment

- Learning phase
  1. Sample n points in $\mathcal{C}_{free}$
  2. Connect the random configurations using a local planner
- Query phase
  1. Connect start and goal configurations with the PRM

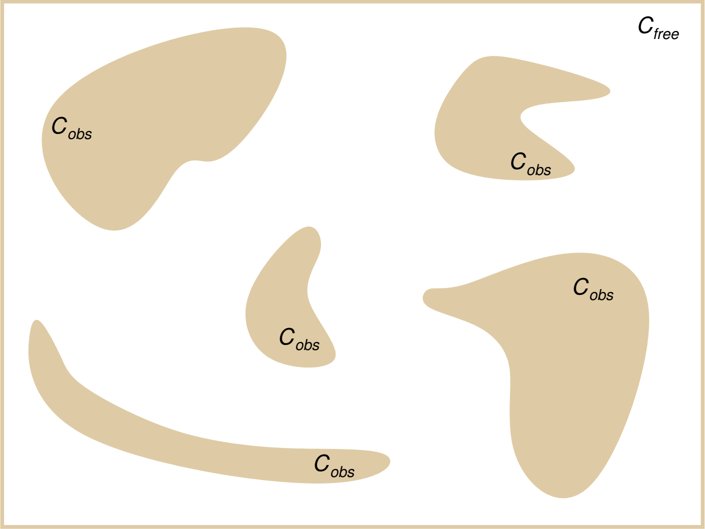
     E.g., using a "local planner"
  2. Use the graph search to find the path

Probabilistic Roadmaps for Path Planning in High Dimensional Configuration Spaces
Lydia E. Kavraki and Petr Svestka and Jean-Claude Latombe and Mark H. Overmars,
IEEE Transactions on Robotics and Automation, 12(4):566–580, 1996.

First planner that demonstrates ability to solve general planning problems in more than 4-5 dimensions.

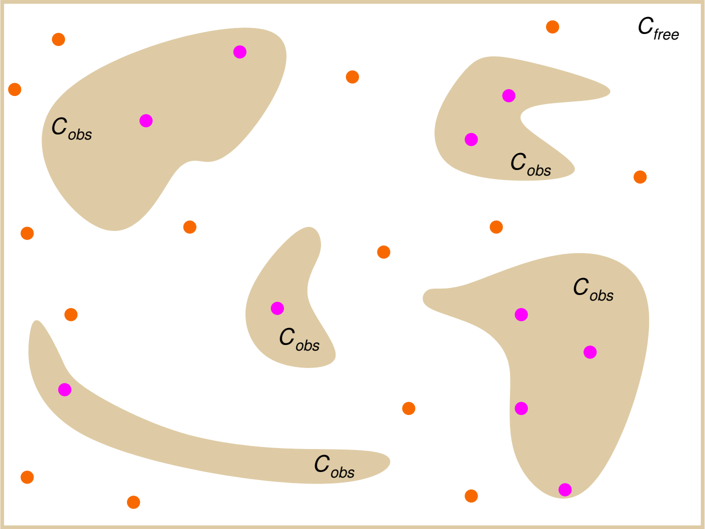
PRM Construction/Query

Given problem domain



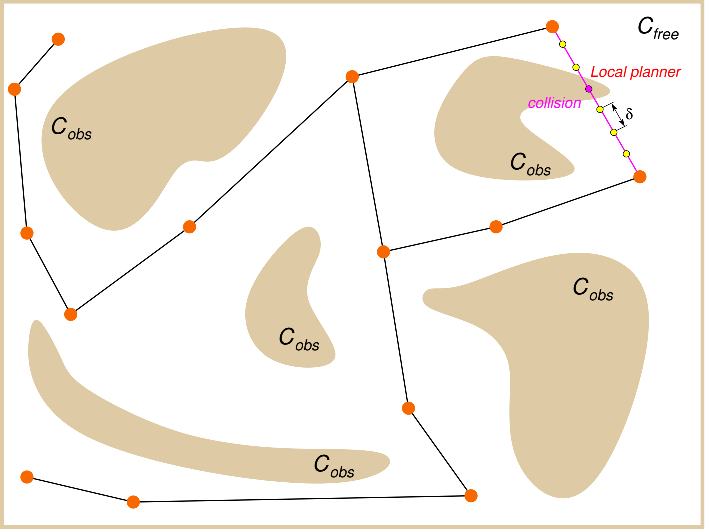
PRM Construction/Query

Random configurations



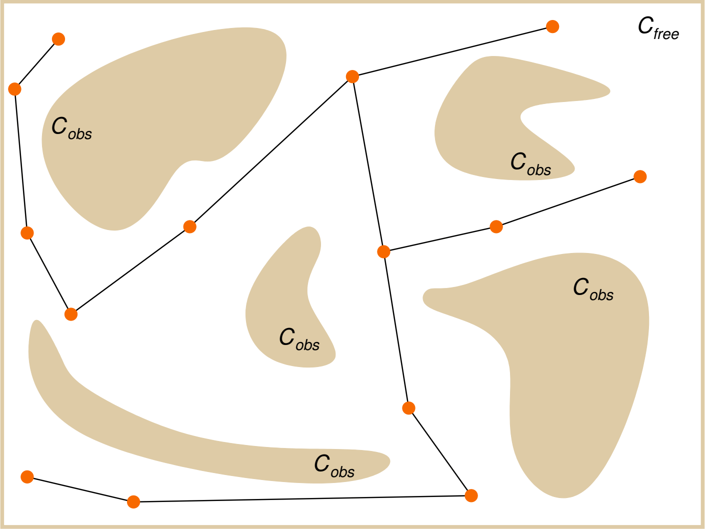
PRM Construction/Query

Connecting random samples



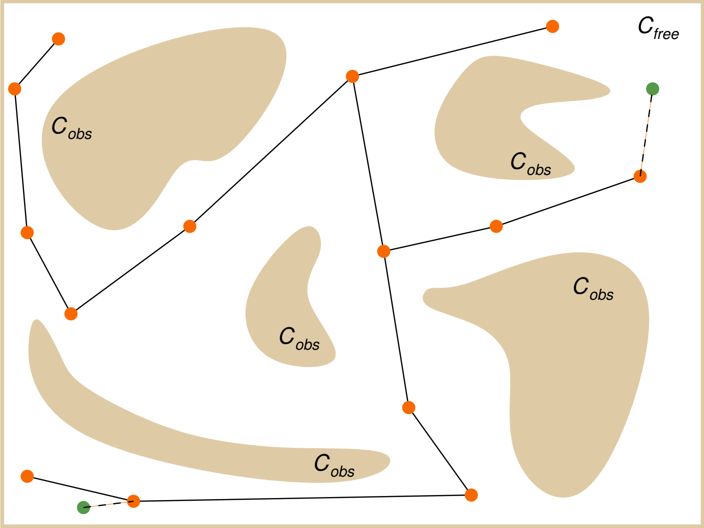
PRM Construction/Query

Connected roadmap



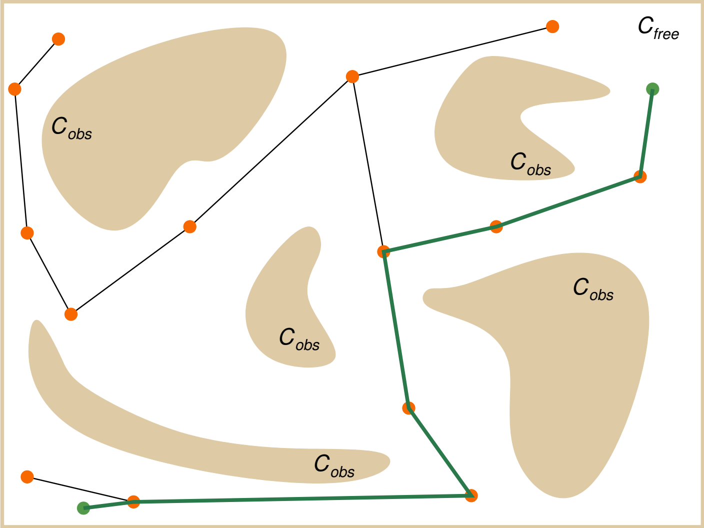
PRM Construction/Query

Query configurations



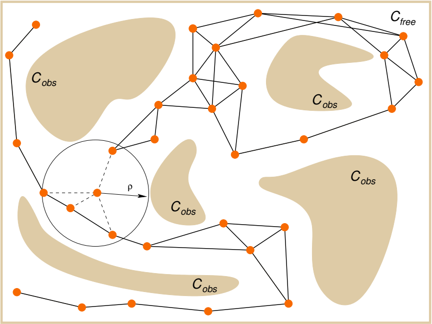
PRM Construction/Query

Final found path



Practical PRM

- Incremental construction
- Connect nodes in a radius ρ
- Local planner tests collisions up to selected resolution δ
- Path can be found by Dijkstra's algorithm

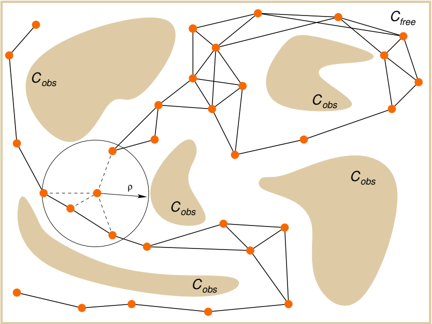


What are the properties of the PRM algorithm?

We need a couple of more formalism.

Practical PRM

- Incremental construction
- Connect nodes in a radius ρ
- Local planner tests collisions up to selected resolution δ
- Path can be found by Dijkstra's algorithm

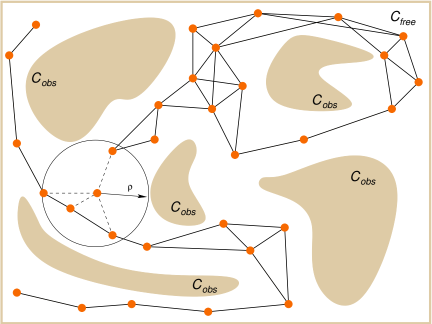


What are the properties of the PRM algorithm?

We need a couple of more formalism.

Practical PRM

- Incremental construction
- Connect nodes in a radius ρ
- Local planner tests collisions up to selected resolution δ
- Path can be found by Dijkstra's algorithm



What are the properties of the PRM algorithm?

We need a couple of more formalism.

Path Planning Problem Formulation

- Path planning problem is defined by a triplet

$$\mathcal{P} = (\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal}),$$

  - $\mathcal{C}_{free} = \mathrm{cl}(\mathcal{C} \setminus \mathcal{C}_{obs})$, $\mathcal{C} = (0,1)^d$, for $d \in \mathbb{N}$, $d \geq 2$
  - $q_{init} \in \mathcal{C}_{free}$ is the initial configuration (condition)
  - $\mathcal{G}_{goal}$ is the goal region defined as an open subspace of $\mathcal{C}_{free}$

- Function $\pi : [0,1] \to \mathbb{R}^d$ of *bounded variation* is called :
  - **path** if it is continuous;
  - **collision-free path** if it is path and $\pi(\tau) \in \mathcal{C}_{free}$ for $\tau \in [0,1]$;
  - **feasible** if it is collision-free path, and $\pi(0) = q_{init}$ and $\pi(1) \in \mathrm{cl}(\mathcal{Q}_{goal})$.

- A function π with the total variation $\mathrm{TV}(\pi) < \infty$ is said to have bounded variation, where $\mathrm{TV}(\pi)$ is the total variation

$$\mathrm{TV}(\pi) = \sup_{\{n \in \mathbb{N}, 0=\tau_0<\tau_1<\ldots<\tau_n=s\}} = \sum_{i=1}^{n} |\pi(\tau_i) - \pi(\tau_{i-1})|$$

- The total variation $\mathrm{TV}(\pi)$ is de facto a path length.

Path Planning Problem Formulation

- Path planning problem is defined by a triplet

$$\mathcal{P} = (\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal}),$$

  - $\mathcal{C}_{free} = \mathrm{cl}(\mathcal{C} \setminus \mathcal{C}_{obs})$, $\mathcal{C} = (0,1)^d$, for $d \in \mathbb{N}$, $d \geq 2$
  - $q_{init} \in \mathcal{C}_{free}$ is the initial configuration (condition)
  - $\mathcal{G}_{goal}$ is the goal region defined as an open subspace of $\mathcal{C}_{free}$
- Function $\pi : [0,1] \to \mathbb{R}^d$ of *bounded variation* is called :
  - **path** if it is continuous;
  - **collision-free path** if it is path and $\pi(\tau) \in \mathcal{C}_{free}$ for $\tau \in [0,1]$;
  - **feasible** if it is collision-free path, and $\pi(0) = q_{init}$ and $\pi(1) \in \mathrm{cl}(\mathcal{Q}_{goal})$.
- A function π with the total variation $\mathrm{TV}(\pi) < \infty$ is said to have bounded variation, where $\mathrm{TV}(\pi)$ is the total variation

$$\mathrm{TV}(\pi) = \sup_{\{n \in \mathbb{N}, 0=\tau_0<\tau_1<\ldots<\tau_n=s\}} = \sum_{i=1}^{n} |\pi(\tau_i) - \pi(\tau_{i-1})|$$

- The total variation $\mathrm{TV}(\pi)$ is de facto a path length.

Path Planning Problem

- **Feasible path planning**:
  For a path planning problem $(\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal})$
  - Find a feasible path $\pi : [0,1] \rightarrow \mathcal{C}_{free}$ such that $\pi(0) = q_{init}$ and $\pi(1) \in \text{cl}(\mathcal{Q}_{goal})$, if such path exists.
  - Report failure if no such path exists.

- **Optimal path planning**:

  The optimality problem ask for a feasible path with the minimum cost.

  For $(\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal})$ and a cost function $c : \Sigma \rightarrow \mathbb{R}_{\geq 0}$
  - Find a feasible path π^* such that $c(\pi^*) = \min\{c(\pi) : \pi \textit{ is feasible}\}$.
  - Report failure if no such path exists.

  The cost function is assumed to be monotonic and bounded, i.e., there exists k_c such that $c(\pi) \leq k_c \text{TV}(\pi)$.

Path Planning Problem

- **Feasible path planning**:
  For a path planning problem $(\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal})$
  - Find a feasible path $\pi : [0,1] \rightarrow \mathcal{C}_{free}$ such that $\pi(0) = q_{init}$ and $\pi(1) \in \text{cl}(\mathcal{Q}_{goal})$, if such path exists.
  - Report failure if no such path exists.

- **Optimal path planning**:

  The optimality problem ask for a feasible path with the minimum cost.

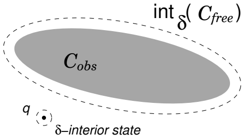
  For $(\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal})$ and a cost function $c : \Sigma \rightarrow \mathbb{R}_{\geq 0}$
  - Find a feasible path π^* such that $c(\pi^*) = \min\{c(\pi) : \pi \textit{ is feasible}\}$.
  - Report failure if no such path exists.

  The cost function is assumed to be monotonic and bounded, i.e., there exists k_c such that $c(\pi) \leq k_c \, \text{TV}(\pi)$.

Probabilistic Completeness 1/2

First, we need **robustly feasible** path planning problem $(\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal})$.

- $q \in \mathcal{C}_{free}$ is *δ-interior state* of $\mathcal{C}_{free}$ if the closed ball of radius δ centered at q lies entirely inside $\mathcal{C}_{free}$.



- *δ-interior* of $\mathcal{C}_{free}$ is $\text{int}_\delta(\mathcal{C}_{free}) = \{q \in \mathcal{C}_{free} | \mathcal{B}_{/,\delta} \subseteq \mathcal{C}_{free}\}$.

A collection of all δ-interior states.

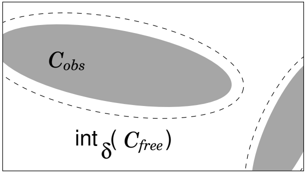
- A collision free path π has strong δ clearance, if π lies entirely inside $\text{int}_\delta(\mathcal{C}_{free})$.
- $(\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal})$ is *robustly feasible* if a solution exists and it is a feasible path with *strong δ-clearance*, for δ>0.

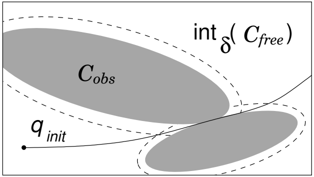
Probabilistic Completeness 2/2

An algorithm $\mathcal{ALG}$ is **probabilistically complete** if, for any robustly feasible path planning problem $\mathcal{P} = (\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal})$

$$\lim_{n \to 0} Pr(\mathcal{ALG} \text{ returns a solution to } \mathcal{P}) = 1.$$

- It is a "*relaxed*" notion of completeness
- Applicable only to problems with a **robust solution.**





We need some space, where random configurations can be sampled

Asymptotic Optimality 1/4

Asymptotic optimality relies on a notion of weak δ-clearance

Notice, we use strong δ-clearance for probabilistic completeness

- Function $\psi : [0, 1] \to \mathcal{C}_{free}$ is called **homotopy**, if $\psi(0) = \pi_1$ and $\psi(1) = \pi_2$ and $\psi(\tau)$ is collision-free path for all $\tau \in [0, 1]$.
- A collision-free path π_1 is **homotopic** to π_2 if there exists homotopy function ψ.

A path homotopic to π can be continuously transformed to π through $\mathcal{C}_{free}$.

Asymptotic Optimality 1/4

Asymptotic optimality relies on a notion of weak δ-clearance

Notice, we use strong δ-clearance for probabilistic completeness

- Function $\psi : [0, 1] \rightarrow \mathcal{C}_{free}$ is called **homotopy**, if $\psi(0) = \pi_1$ and $\psi(1) = \pi_2$ and $\psi(\tau)$ is collision-free path for all $\tau \in [0, 1]$.
- A collision-free path π_1 is **homotopic** to π_2 if there exists homotopy function ψ.

A path homotopic to π can be continuously transformed to π through $\mathcal{C}_{free}$.

Asymptotic Optimality 1/4

Asymptotic optimality relies on a notion of weak δ-clearance

Notice, we use strong δ-clearance for probabilistic completeness

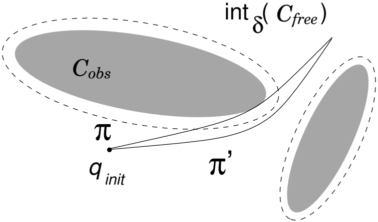
- Function $\psi : [0, 1] \rightarrow \mathcal{C}_{free}$ is called **homotopy**, if $\psi(0) = \pi_1$ and $\psi(1) = \pi_2$ and $\psi(\tau)$ is collision-free path for all $\tau \in [0, 1]$.
- A collision-free path π_1 is **homotopic** to π_2 if there exists homotopy function ψ.

A path homotopic to π can be continuously transformed to π through $\mathcal{C}_{free}$.

Asymptotic Optimality 2/4

- A collision-free path $\pi : [0, s] \rightarrow \mathcal{C}_{free}$ has **weak δ-clearance** if there exists a path π' that has strong δ-clearance and homotopy ψ with $\psi(0) = \pi$, $\psi(1) = \pi'$, and for all $\alpha \in (0, 1]$ there exists $\delta_\alpha > 0$ such that $\psi(\alpha)$ has strong δ-clearance.

Weak δ-clearance does not require points along a path to be at least a distance δ away from obstacles.



- A path π with a weak δ-clearance
- π' lies in $\text{int}_\delta(\mathcal{C}_{free})$ and it is the same homotopy class as π

Asymptotic Optimality 3/4

- It is applicable with a **robust optimal solution** that can be obtained as a limit of robust (non-optimal) solutions.
- A collision-free path π^* is **robustly optimal solution** if it has *weak δ-clearance* and for any sequence of collision free paths $\{\pi_n\}_{n\in\mathbb{N}}$, $\pi_n \in \mathcal{C}_{free}$ such that $\lim_{n\to\infty} \pi_n = \pi^*$,

$$\lim_{n\to\infty} c(\pi_n) = c(\pi^*).$$

There exists a path with strong δ-clearance, and π^* is homotopic to such path and π^* is of the lower cost.

- Weak δ-clearance implies robustly feasible solution problem

(thus, probabilistic completeness)

Asymptotic Optimality 4/4

An algorithm $\mathcal{ALG}$ is **asymptotically optimal** if, for any path planning problem $\mathcal{P} = (\mathcal{C}_{free}, q_{init}, \mathcal{Q}_{goal})$ and cost function c that admit a robust optimal solution with the finite cost c^*

$$Pr\left(\left\{\lim_{i\to\infty} Y_i^{\mathcal{ALG}} = c^*\right\}\right) = 1.$$

- $Y_i^{\mathcal{ALG}}$ is the extended random variable corresponding to the minimum-cost solution included in the graph returned by $\mathcal{ALG}$ at the end of iteration i.

Properties of the PRM Algorithm

- Completeness for the standard PRM has not been provided when it was introduced
- A simplified version of the PRM (called sPRM) has been mostly studied
- sPRM is probabilistically complete

What are the differences between PRM and sPRM?

PRM vs simplified PRM (sPRM)

PRM

Input: q_{init}, number of samples n, radius ρ
Output: PRM – $G = (V, E)$

$V \leftarrow \emptyset$; $E \leftarrow \emptyset$;
for $i = 0, \ldots, n$ **do**
  $q_{rand} \leftarrow$ SampleFree;
  $U \leftarrow$ Near($G = (V, E), q_{rand}, \rho$);
  $V \leftarrow V \cup \{q_{rand}\}$;
  foreach $u \in U$, *with increasing* $||u - q_r||$ **do**
    if q_{rand} *and* u *are not in the same connected component of* $G = (V, E)$ **then**
      if CollisionFree(q_{rand}, u) **then**
        $E \leftarrow E \cup \{(q_{rand}, u), (u, q_{rand})\}$;
      end
    end
  end
end
return $G = (V, E)$;

sPRM Algorithm

Input: q_{init}, number of samples n, radius ρ
Output: PRM – $G = (V, E)$

$V \leftarrow \{q_{init}\} \cup \{\text{SampleFree}_i\}_{i=1,\ldots,n-1}$; $E \leftarrow \emptyset$;
foreach $v \in V$ **do**
  $U \leftarrow$ Near($G = (V, E), v, \rho) \setminus \{v\}$;
  foreach $u \in U$ **do**
    if CollisionFree(v, u) **then**
      $E \leftarrow E \cup \{(v, u), (u, v)\}$;
    end
  end
end
return $G = (V, E)$;

There are several ways for the set U of vertices to connect them

- k-nearest neighbors to v
- variable connection radius ρ as a function of n

PRM – Properties

- sPRM (simplified PRM)
  - **Probabilistically complete and asymptotically optimal**
  - Processing complexity $O(n^2)$
  - Query complexity $O(n^2)$
  - Space complexity $O(n^2)$
- Heuristics practically used are usually not probabilistic complete
  - k-nearest sPRM is not probabilistically complete
  - variable radius sPRM is not probabilistically complete

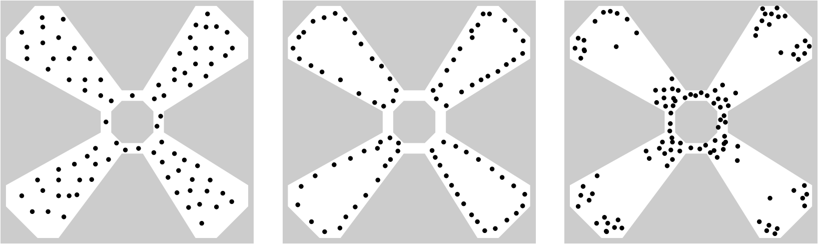
Based on analysis of Karaman and Frazzoli

PRM algorithm is:

\+ very simple implementation
\+ Completeness (for sPRM)
\- Differential constraints (car-like vehicles) are not straightforward

Comments about Random Sampling 1/2

- Different sampling strategies (distributions) may be applied



- Notice, one of the main issue of the randomized sampling-based approaches is the narrow passage
- Several modifications of sampling based strategies have been proposed in the last decades

Comments about Random Sampling 2/2

- A solution can be found using only a few samples.

Do you know the Oraculum? (from Alice in Wonderland)

- Sampling strategies are important
  - Near obstacles
  - Narrow passages
  - Grid-based
  - Uniform sampling must be carefully considered.

James J. Kuffner, Effective Sampling and Distance Metrics for 3D Rigid Body Path Planning, ICRA, 2004.

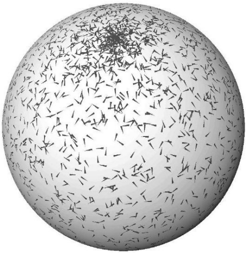
Comments about Random Sampling 2/2

- A solution can be found using only a few samples.

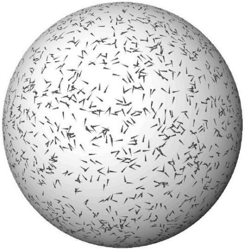
Do you know the Oraculum? (from Alice in Wonderland)

- Sampling strategies are important
  - Near obstacles
  - Narrow passages
  - Grid-based
  - Uniform sampling must be carefully considered.

James J. Kuffner, Effective Sampling and Distance Metrics for 3D Rigid Body Path Planning, ICRA, 2004.



Naïve sampling



Uniform sampling of SO(3) using Euler angles

Rapidly Exploring Random Tree (RRT)

- Motivation is a single query and *control-based* path finding
- It incrementally builds a graph (tree) towards the goal area.

RRT Algorithm

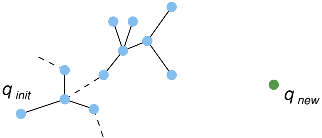
Input: q_{init}, number of samples n
Output: Roadmap $G = (V, E)$

$V \leftarrow \{q_{init}\}; E \leftarrow \emptyset;$
for $i = 1, \ldots, n$ **do**
  $q_{rand} \leftarrow$ SampleFree;
  $q_{nearest} \leftarrow$ Nearest($G = (V, E), q_{rand}$);
  $q_{new} \leftarrow$ Steer($q_{nearest}, q_{rand}$);
  if CollisionFree($q_{nearest}, q_{new}$) **then**
    $V \leftarrow V \cup \{x_{new}\}; E \leftarrow E \cup \{(x_{nearest}, x_{new})\};$
  end
end
return $G = (V, E)$;

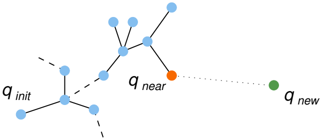
Extend tree by a small step, but often a direct control $u \in \mathcal{U}$ that will move robot to the position closest to q_{new} is selected (applied for dt).

Rapidly-exploring random trees: A new tool for path planning
S. M. LaValle,
Technical Report 98-11, Computer Science Dept., Iowa State University, 1998

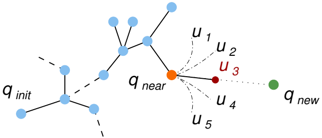
RRT Construction



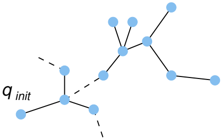
#1 new random configuration



#2 the closest node



#3 possible actions from q_{near}



#4 extended tree

Expansion is repeated until $\mathcal{Q}_{goal}$ is achieved or maximum iteration is reached.

Properties of RRT Algorithms

- Rapidly explores the space

  q_{new} will more likely be generated in large not yet covered parts of $\mathcal{C}_{free}$.
- Allows considering kinodynamic/dynamic constraints

  during the expansion
- Can provide trajectory or a sequence of direct control commands for robot controllers
- A collision detection test is usually used as a "black-box".

  E.g., RAPID, Bullet libraries.
- Poor performance in narrow passage problems

  similarly to PRM
- Provides feasible paths

  more expansions can improve paths; however, . . .
- Many variants of RRT have been proposed

Car-Like Robot

- Configuration

$$\boldsymbol{x}(t) = \begin{pmatrix} x \\ y \\ \phi \end{pmatrix}$$

position and orientation

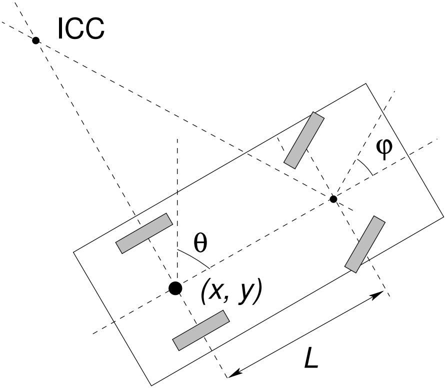
- Controls

$$\boldsymbol{u}(t) = \begin{pmatrix} v \\ \varphi \end{pmatrix}$$

forward velocity, steering angle

- System equation

$$\begin{aligned} \dot{x} &= v \cos\phi \\ \dot{y} &= v \sin\phi \\ \dot{\varphi} &= \frac{v}{L}\tan\varphi \end{aligned}$$



Kinematic constraints $\dim(\vec{\boldsymbol{u}}) < \dim(\vec{\boldsymbol{x}})$

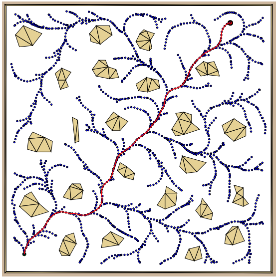
Differential constraints on possible $\dot{q}$:

$$\dot{x}\sin(\phi) - \dot{y}\cos(\phi) = 0$$

Control-Based Sampling

- Select a configuration q from the tree T of the current configurations
- Pick a control input $\boldsymbol{u} = (v, \varphi)$ and integrate system (motion) equation over a short period

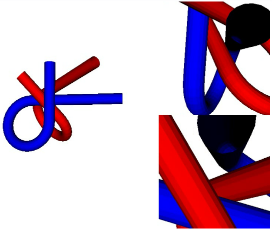
$$\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta \varphi \end{pmatrix} = \int_t^{t+\Delta t} \begin{pmatrix} v \cos \phi \\ v \sin \phi \\ \frac{v}{L} \tan \varphi \end{pmatrix} dt$$



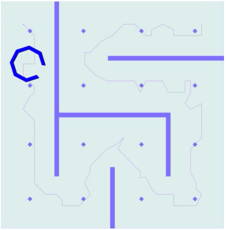
- If the motion is collision-free, add the endpoint to the tree

E.g., considering k configurations for $k\delta t = dt$.

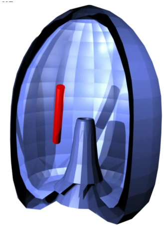
RRT – Examples 1/2



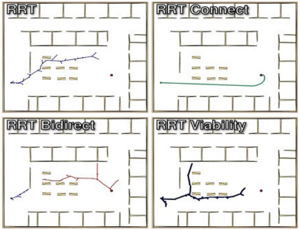
Alpha puzzle benchmark



Apply rotations to reach the goal



Bugtrap benchmark

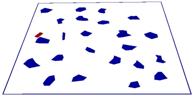


Variants of RRT algorithms

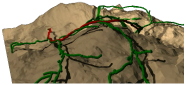
Courtesy of V. Vonásek and P. Vaněk

RRT – Examples 2/2

- Planning for a car-like robot

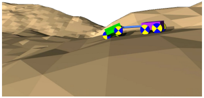


- Planning on a 3D surface



- Planning with dynamics

(friction forces)



Courtesy of V. Vonásek and P. Vaněk

RRT and Quality of Solution

- RRT provides a feasible solution without quality guarantee

  Despite of that, it is successfully used in many practical applications

- In 2011, a systematical study of the asymptotic behaviour of randomized sampling-based planners has been published

  It shows, that in some cases, they converge to a non-optimal value with a probability 1.

Sampling-based algorithms for optimal motion planning
Sertac Karaman, Emilio Frazzoli
International Journal of Robotic Research, 30(7):846–894, 2011.

RRT and Quality of Solution

- RRT provides a feasible solution without quality guarantee

  Despite of that, it is successfully used in many practical applications

- In 2011, a systematical study of the asymptotic behaviour of randomized sampling-based planners has been published

  It shows, that in some cases, they converge to a non-optimal value with a probability 1.

Sampling-based algorithms for optimal motion planning
Sertac Karaman, Emilio Frazzoli
International Journal of Robotic Research, 30(7):846–894, 2011.

RRT and Quality of Solution

- RRT provides a feasible solution without quality guarantee

  Despite of that, it is successfully used in many practical applications

- In 2011, a systematical study of the asymptotic behaviour of randomized sampling-based planners has been published

  It shows, that in some cases, they converge to a non-optimal value with a probability 1.

Sampling-based algorithms for optimal motion planning
Sertac Karaman, Emilio Frazzoli
International Journal of Robotic Research, 30(7):846–894, 2011.

RRT and Quality of Solution 1/2

- Let Y_i^{RRT} be the cost of the best path in the RRT at the end of iteration i.
- Y_i^{RRT} converges to a random variable

$$\lim_{i \to \infty} Y_i^{RRT} = Y_\infty^{RRT}.$$

- The random variable Y_∞^{RRT} is sampled from a distribution with zero mass at the optimum, and

$$Pr[Y_\infty^{RRT} > c^*] = 1.$$

Karaman and Frazzoli, 2011

- The best path in the RRT converges to a sub-optimal solution almost surely.

RRT and Quality of Solution 2/2

- RRT does not satify a necessary condition for the asymptotic optimality
  - For $0 < R < \inf_{q \in \mathcal{Q}_{goal}} ||q - q_{init}||$, the event $\{\lim_{n\to\infty} Y_n^{RTT} = c^*\}$ occurs only if the k-th branch of the RRT contains vertices outside the R-ball centered at q_{init} for infinitely many k.

See Appendix B in Karaman&Frazzoli, 2011

- It is required the root node will have infinitely many subtrees that extend at least a distance ϵ away from q_{init}

The sub-optimality is caused by disallowing new better paths to be discovered.