Robot motion planning

and the connection to nearest-neighbor search

Computational Geometry Course

Michal Kleinbort

Tel Aviv University, Dec 2020

A robot

- A mechanical device capable of sensing its environment and controlled by a computing system
- Operates in a real-world workspace, populated by physical objects
- Performs tasks by executing motions in the workspace
- An autonomous robot is required to plan its own motions automatically in order to achieve a given task

Some examples

Robotic vacuum cleaners

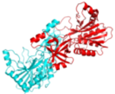
Self-driving cars

Robotic arms

Drones

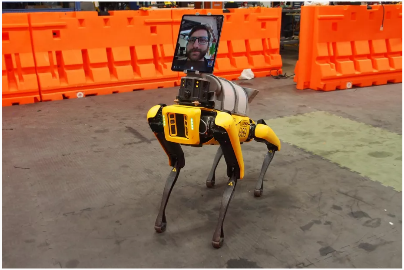
Robotic arms for medical use

Multi-robot settings



Proteins can be considered as robots that execute motion in order to fold

Some examples



In the context of COVID-19

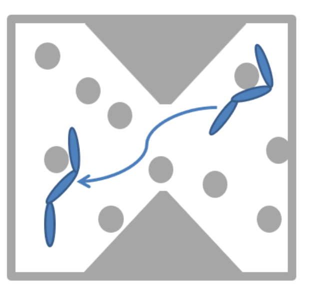
The motion-planning problem

Given:

- A robot R
- A workspace $\mathcal{W}$ (with obstacles)
- Initial and final positions

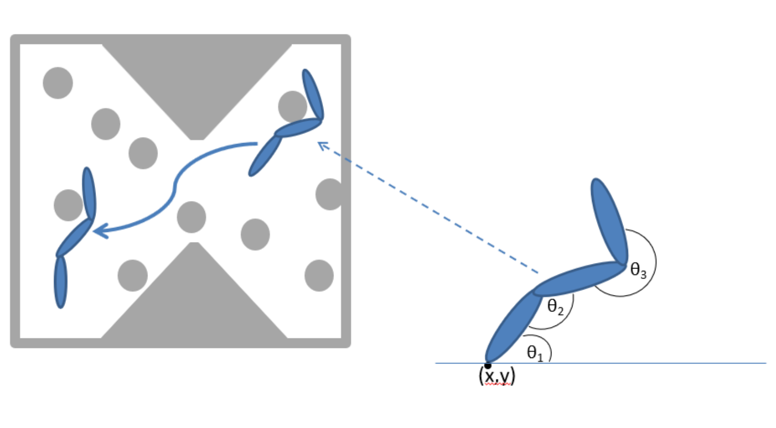
Goal:

- Plan a continuous path for the robot from the initial position to the final position, while avoiding collision with obstacles and self collisions of the robot



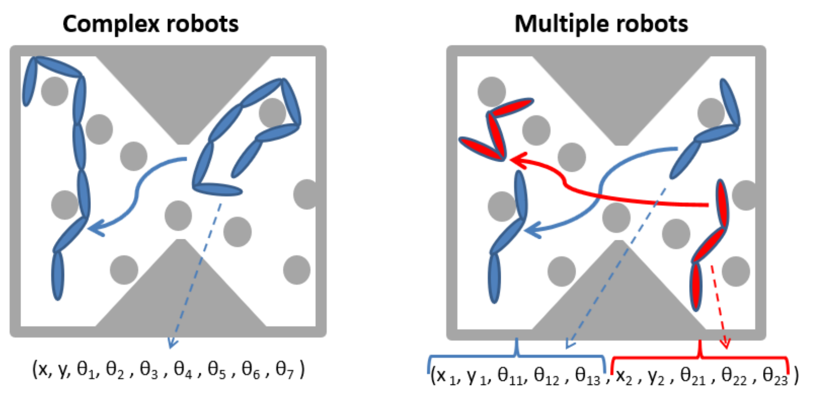
A configuration of the robot

A configuration of the robot is a complete specification of the position of every point of the robot, e.g., $(x, y, \Theta_1, \Theta_2, \Theta_3)$



The dimension

The dimension of the motion-planning problem (or the number of degrees of freedom) is the smallest number $d \geq 1$ of coordinates needed to represent a configuration of the robot.

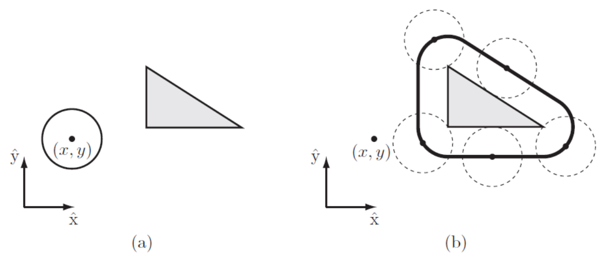


The configuration space

The d-dimensional space $\mathcal{C}$ containing all possible configurations of the robot is called the configuration space (C-space).

A subset $\mathcal{F} \subset \mathcal{C}$ of all the collision-free configurations is called the free space.

The C-obstacles, defined as $\mathcal{C}_{\text{forb}} = \mathcal{C} \setminus \mathcal{F}$, are rarely represented exactly (may have a complex mathematical representation).



Figures from [Lynch and Park, 16]

An alternative formulation of the MP problem

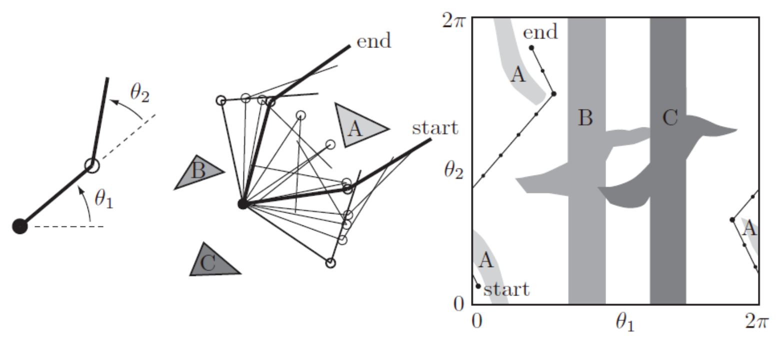
Given:

- A point robot
- A d-dimensional configuration space $\mathcal{C}$ (C-space)
- C-obstacles (often not explicitly given) $\mathcal{C}_{\text{forb}}$
- Free space $\mathcal{F} = \mathcal{C} \setminus \mathcal{C}_{\text{forb}}$
- Initial and final configurations

Goal:

- Plan a continuous path in the free space from the initial configuration to the final configuration

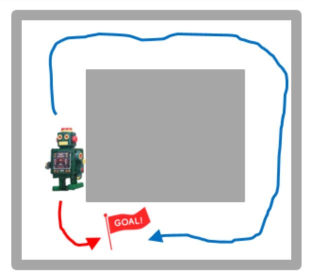
An alternative formulation of the MP problem



Figures from [Lynch and Park, 16]

Challenges

- High-dimensional problems are "hard" to solve
- Finding an optimal path is harder than finding a path
    - minimal path length
    - maximal distance from obstacles
    - smoothness



Sampling-based methods for solving the problem

- Attempt to capture the structure of the C-space by constructing a graph with n randomly sampled nodes (called a roadmap)
  - The nodes are collision-free configurations sampled at random
  - Two nearby nodes are connected by an edge if the path between them is collision-free
- We can often say something about the asymptotic behavior of the algorithm (as $n \to \infty$):
  - probabilistically complete (PC) algorithm: With high probability finds a solution as $n \to \infty$, if one exists

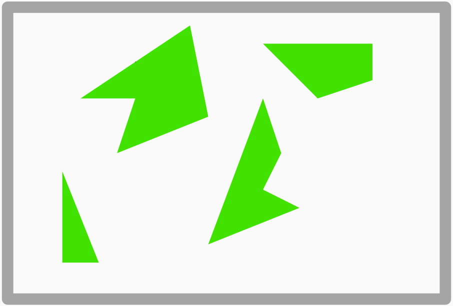
Primitive operations in sampling-based methods

- Collision detection (CD)
  - Determines whether a configuration or a C-space path between two configurations is collision-free. The latter is termed local planning (LP)
  - Complexity usually depends on both the complexity of the workspace obstacles and the complexity of the robot
- Nearest-neighbor search (NN)
  - Returns the nearest neighbor (or neighbors) of a given configuration
  - Complexity depends on the number n of nodes and the dimension d

The main practical computational bottleneck is typically considered to be CD (including LP)

Probabilistic roadmaps (PRM)

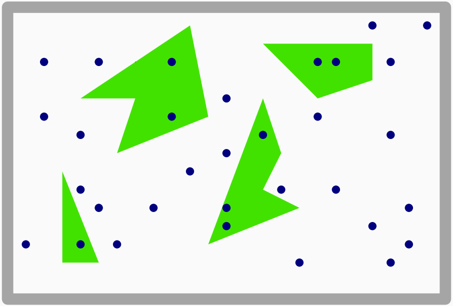
Multi-query algorithm that generates a roadmap (graph) that is embedded in the free space [Kavraki et al., 95]



Query: Givenstart and two confifdsfsdfsdfsdfdsffdsfsdfsdfsdfsd-dddddddddddddddddddddddddddddddddsfsdfsdgurations

Probabilistic roadmaps (PRM)

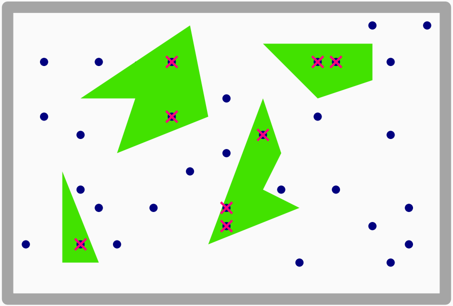
Multi-query algorithm that generates a roadmap (graph) that is embedded in the free space [Kavraki et al., 95]



Query: Givenstart and two confifdsfsdfsdfsdfdsffdsfsdfsdfsdfsd-dddddddddddddddddddddddddddddddddsfsdfsdgurations

Probabilistic roadmaps (PRM)

Multi-query algorithm that generates a roadmap (graph) that is embedded in the free space [Kavraki et al., 95]

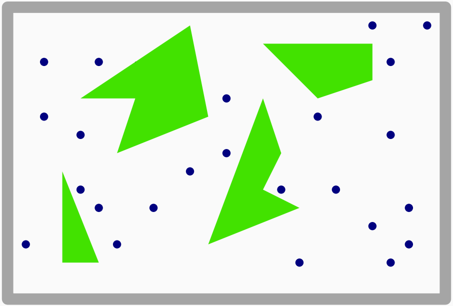


Involves CD operation

Query: Given configurations, find the

Probabilistic roadmaps (PRM)

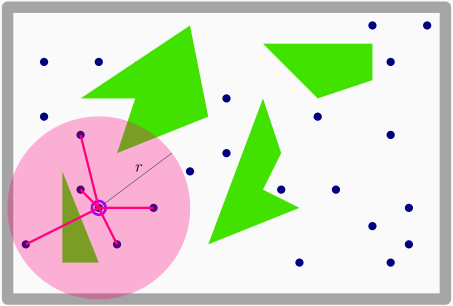
Multi-query algorithm that generates a roadmap (graph) that is embedded in the free space [Kavraki et al., 95]



Query: Givenstart and two confifdsfsdfsdfsdfdsffdsfsdfsdfsdfsd-dddddddddddddddddddddddddddddddddsfsdfsdgurations

Probabilistic roadmaps (PRM)

Multi-query algorithm that generates a roadmap (graph) that is embedded in the free space [Kavraki et al., 95]

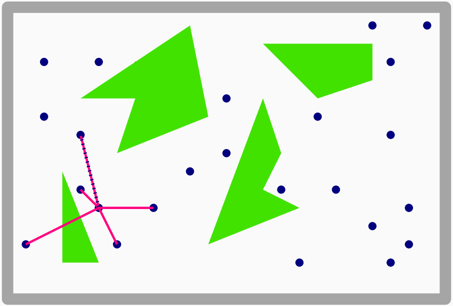


Involves NN operation (r-nearest neighbors or k-nearest neighbors)

Query: Given configurations, find the

Probabilistic roadmaps (PRM)

Multi-query algorithm that generates a roadmap (graph) that is embedded in the free space [Kavraki et al., 95]

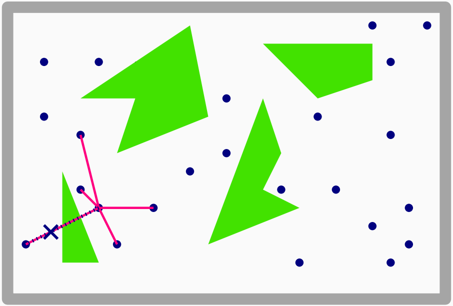


Involves CD operation (as an LP sub-procedure)

Query: Given configurations, find the shortest path b

Probabilistic roadmaps (PRM)

Multi-query algorithm that generates a roadmap (graph) that is embedded in the free space [Kavraki et al., 95]

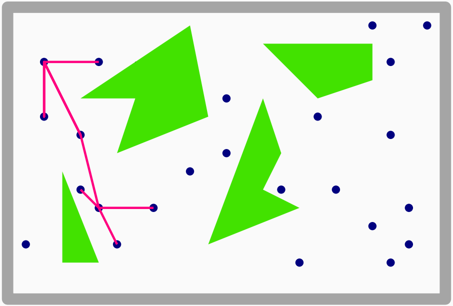


Involves CD operation (as an LP sub-procedure)

Query: Given configurations, find the shortest path b

Probabilistic roadmaps (PRM)

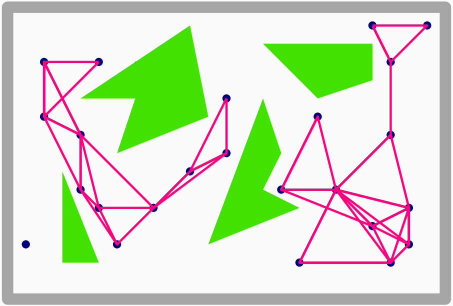
Multi-query algorithm that generates a roadmap (graph) that is embedded in the free space [Kavraki et al., 95]



Query: Givenstart and two confifdsfsdfsdfsdfdsffdsfsdfsdfsdfsd-dddddddddddddddddddddddddddddddddsfsdfsdgurations

Probabilistic roadmaps (PRM)

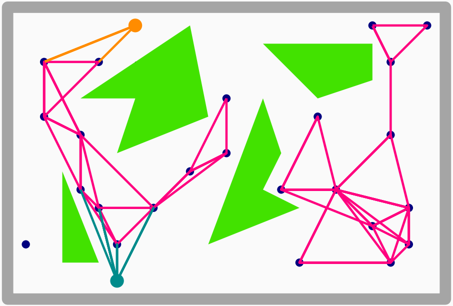
Multi-query algorithm that generates a roadmap (graph) that is embedded in the free space [Kavraki et al., 95]



Query: Givenstart and two confifdsfsdfsdfsdfdsffdsfsdfsdfsdfsd-ddddddddddddddddddddddddddddddsfsdfsdgurations

Probabilistic roadmaps (PRM)

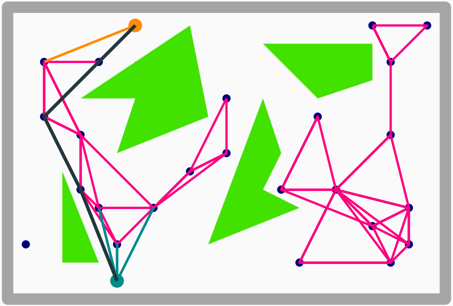
Multi-query algorithm that generates a roadmap (graph) that is embedded in the free space [Kavraki et al., 95]



Given a query—start and goal configurations—add them to the roadmap and find a roadmap path

Probabilistic roadmaps (PRM)

Multi-query algorithm that generates a roadmap (graph) that is embedded in the free space [Kavraki et al., 95]



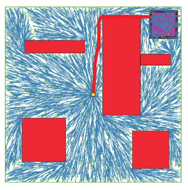
Given a query—start and goal configurations—add them to the roadmap and find a roadmap path

Asymptotic optimality

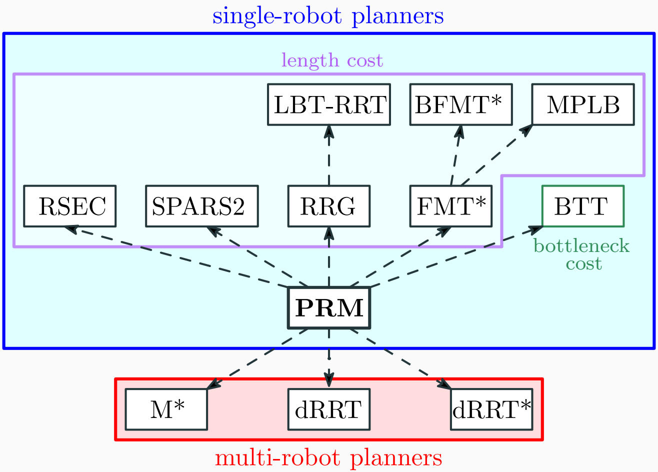
An asymptotically optimal (AO) algorithm is guaranteed to return a solution that converges to the optimum as $n \to \infty$.

Results from the seminal work of [Karaman and Frazzoli, 11]:

- PRM*—PRM with connection radius $r_n > \gamma \left(\frac{\log n}{n}\right)^{1/d}$ for some $\gamma > 0$—is AO
- r_n cannot be smaller than $\gamma' n^{-1/d}$, for some $\gamma' > 0$
- Two single query AO planners: RRT*, RRG



Sampling-based planners



The bottleneck in SB planners [K., Salzman and Halperin, 16]

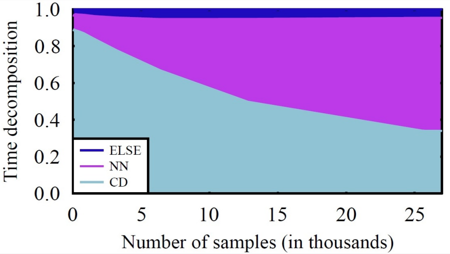
The main practical computational bottleneck is typically considered to be CD (including LP).

We formally prove that the complexity of NN search dominates the asymptotic running time of several AO algorithms.

*appeared in: WAFR 2016

The bottleneck in SB planners [K., Salzman and Halperin, 16]

We characterize settings in which the role of NN is far from negligible and show experimentally that NN may dominate CD after finite time.



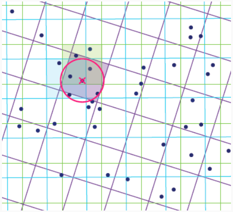
*appeared in: WAFR 2016

NN-sensitive settings

Efficient, specifically-tailored NN data structures can be used in such settings to reduce the overall time of the motion-planning algorithm

Adapting "all-pairs rNN" for sampling-based planners [K., Salzman and Halperin, 15]

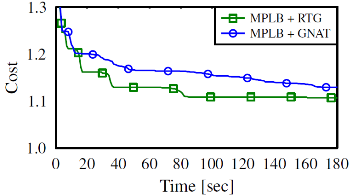
- In several planning algorithms "all-pairs" r-NN are used with a predefined value $r(n) = O((\frac{\log n}{n})^{1/d})$ to achieve AO
- Randomly transformed grids (RTG) [Aiger, Kaplan, Sharir, 14] is a novel method for approximate all-pairs r-NN

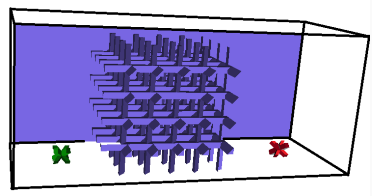


*appeared in: ICRA 2015

Adapting "all-pairs rNN" for sampling-based planners [K., Salzman and Halperin, 15]

- We implemented RTG and used it for certain (NN-sensitive) sampling-based algorithms
- We obtain significant speedups improving: the construction time, the time to find an initial solution, and the time to converge to high-quality solutions



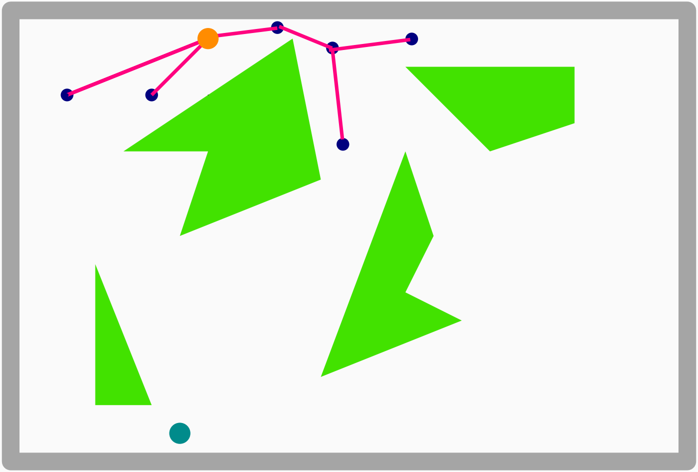


Faster convergence to high-quality solutions (6D non-Euclidean C-space)

*appeared in: ICRA 2015

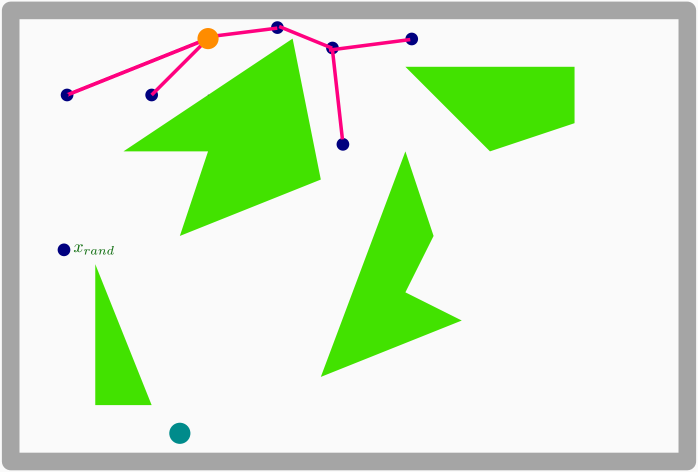
Rapidly exploring random tree (RRT)

Single-query algorithm that generates a tree that is embedded in the free space [LaValle and Kuffner, 01]



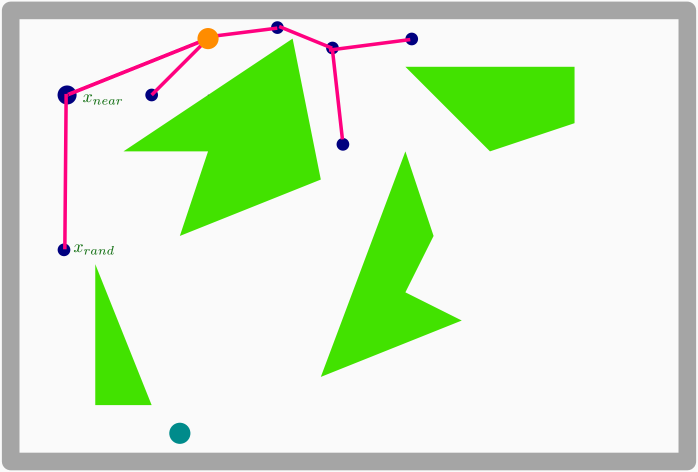
Rapidly exploring random tree (RRT)

Single-query algorithm that generates a tree that is embedded in the free space [LaValle and Kuffner, 01]



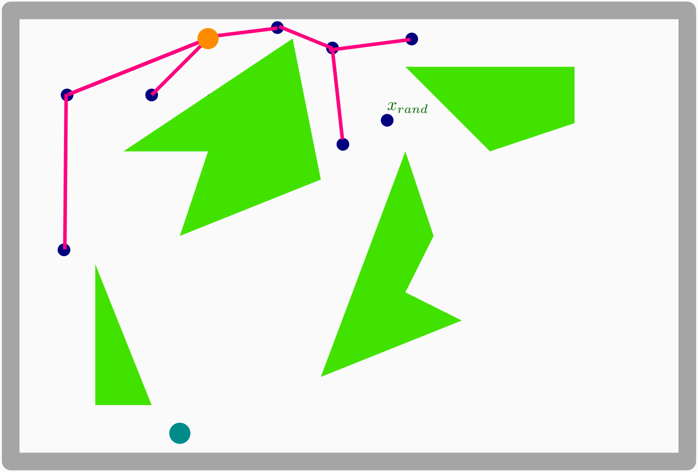
Rapidly exploring random tree (RRT)

Single-query algorithm that generates a tree that is embedded in the free space [LaValle and Kuffner, 01]



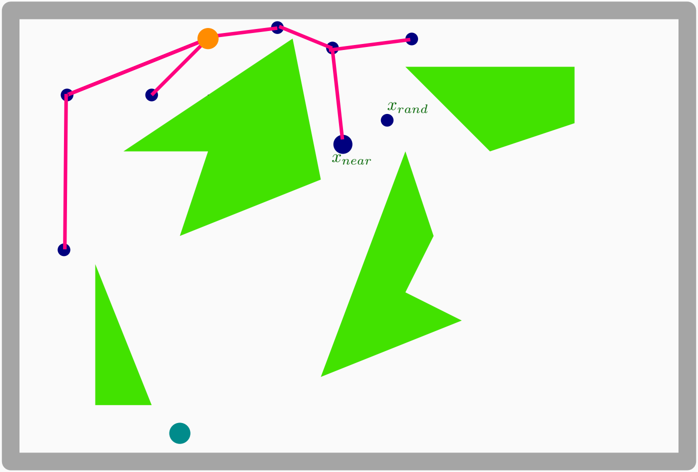
Rapidly exploring random tree (RRT)

Single-query algorithm that generates a tree that is embedded in the free space [LaValle and Kuffner, 01]



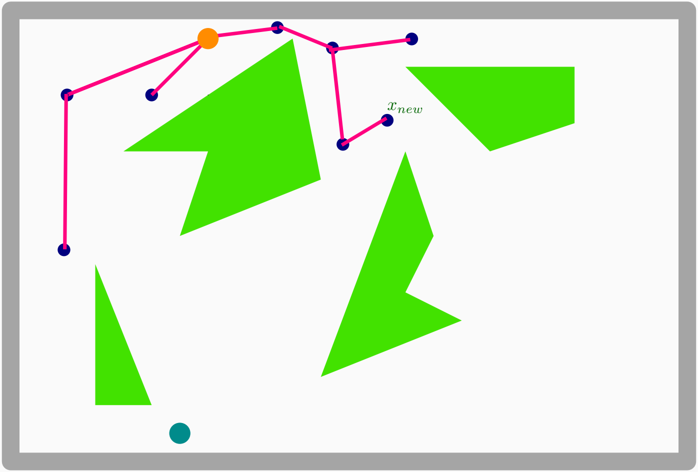
Rapidly exploring random tree (RRT)

Single-query algorithm that generates a tree that is embedded in the free space [LaValle and Kuffner, 01]



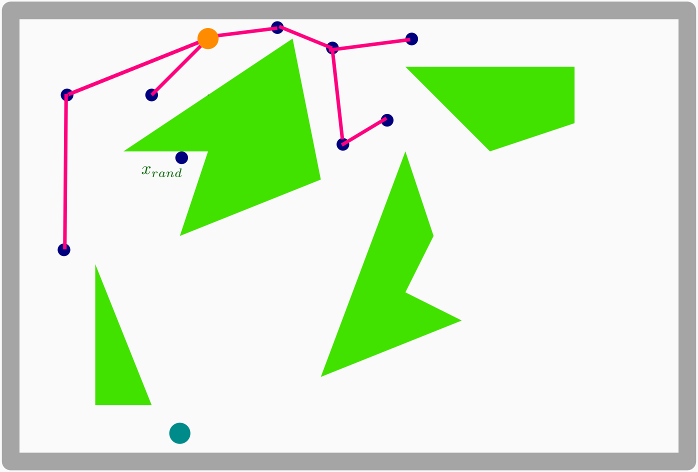
Rapidly exploring random tree (RRT)

Single-query algorithm that generates a tree that is embedded in the free space [LaValle and Kuffner, 01]



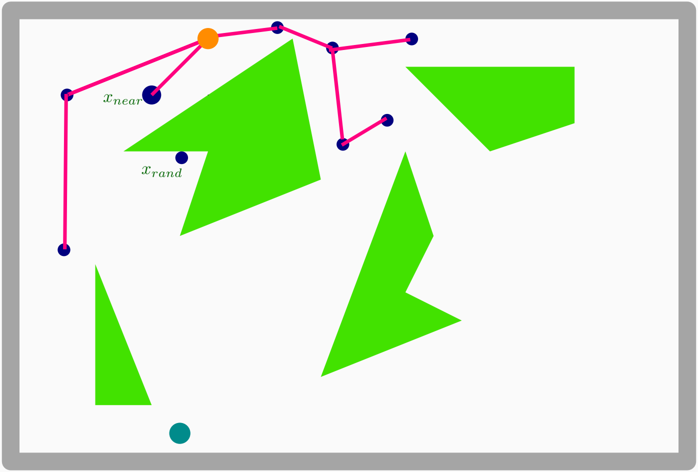
Rapidly exploring random tree (RRT)

Single-query algorithm that generates a tree that is embedded in the free space [LaValle and Kuffner, 01]



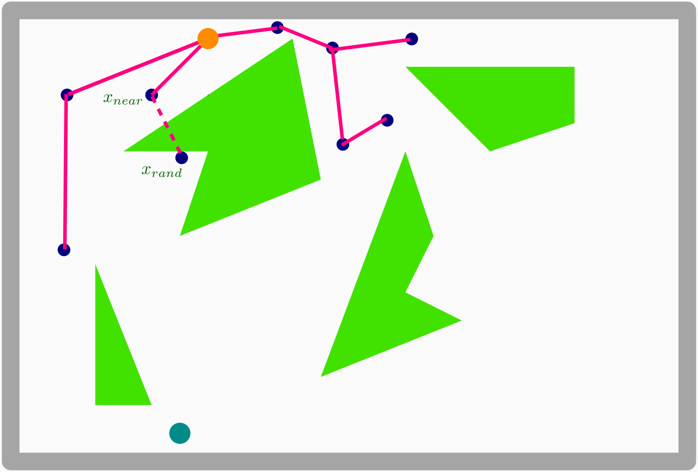
Rapidly exploring random tree (RRT)

Single-query algorithm that generates a tree that is embedded in the free space [LaValle and Kuffner, 01]



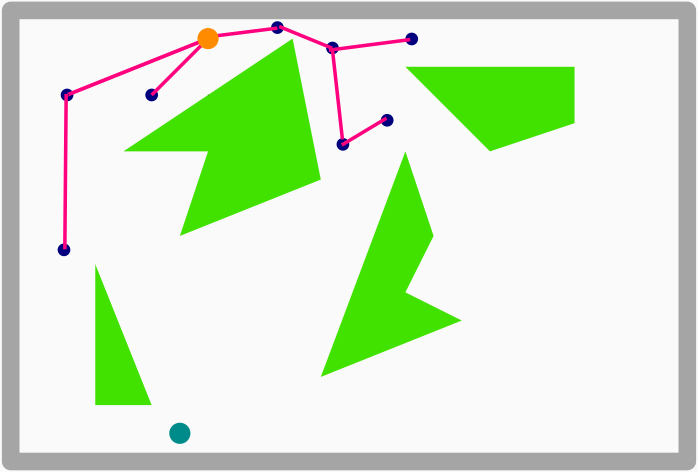
Rapidly exploring random tree (RRT)

Single-query algorithm that generates a tree that is embedded in the free space [LaValle and Kuffner, 01]



Rapidly exploring random tree (RRT)

Single-query algorithm that generates a tree that is embedded in the free space [LaValle and Kuffner, 01]



Rapidly exploring random tree (RRT)

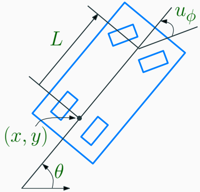
GEOM-RRT($x_{init}, X_{goal}, k, \eta$):

```
T.init(x_init)
for i = 1 to k do
    x_rand ← RANDOM_STATE()
    x_near ← NEAREST_NEIGHBOR(x_rand, T)
    x_new ← NEW_STATE(x_rand, x_near, η)
    if COLLISION_FREE(x_near, x_new) then
        T.add_vertex(x_new)
        T.add_edge(x_near, x_new)
return T
```

More about RRT

- Probably the most commonly used planner
- Well suited to complex tasks involving kinodynamic constraints:
  Example: a kinodynamic car
  Each state keeps $(x, y, \theta, \dot{x}, \dot{y}, \dot{\theta})$ and there are two control inputs (signed speed u_s and steering angle u_ϕ)

$$\begin{aligned}
\dot{x} &= u_s \cos\theta, \\
\dot{y} &= u_s \sin\theta, \\
\dot{\theta} &= \frac{u_s}{L} \tan u_\phi,
\end{aligned}$$



More about RRT

- Probably the most commonly used planner
- Well suited to complex tasks involving kinodynamic constraints:

  Does not require a steering function that returns a trajectory between two states—corresponds to solving the Two-point boundary value problem (BVP)

Probabilistic completeness of RRT

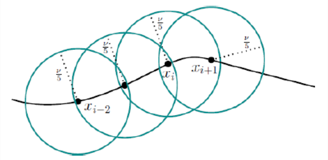
In [K. et al., 19] we devise a rigorous proof for the PC of RRT

Notation:

- The free space $\mathcal{F} \subset \mathcal{C}$
- Euclidean metric $\|\cdot\|$
- A valid path $\pi : [0, t_\pi] \to \mathcal{F}$,
  $\pi(0) = x_{\text{init}}$,
  $\pi(t_\pi) = x_{\text{goal}} \in X_{\text{goal}}$
- $\delta > 0$ is the clearance of π from the obstacles

PC proof of (geometric) RRT

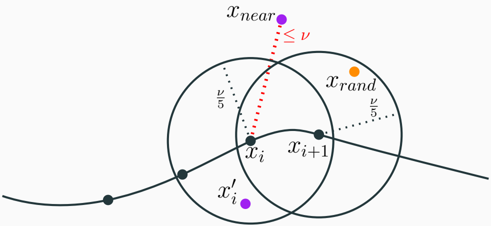
- Wlog, assume that a valid trajectory π exists, whose length is L
- Let $m = 5L/\nu$, where $\nu = min(\delta, \eta)$
- Let $x_0 = x_{\text{init}}, \ldots, x_m = x_{\text{goal}}$ be a sequence of $m + 1$ points along π
  - Dividing π into sub-trajectories of length $\nu/5$
- Let $\mathcal{B}_{\nu/5}(x_0), \ldots, \mathcal{B}_{\nu/5}(x_m)$ be a set of balls of radius $\nu/5$



- We prove that with high probability RRT will generate a path that goes through these balls

Lemma 1

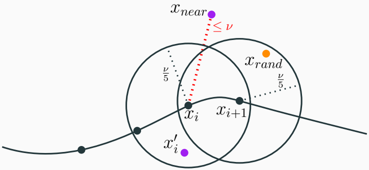
Let $x'_i \in \mathcal{B}_{\nu/5}(x_i)$ be a vertex in T and let $x_{\text{rand}} \in \mathcal{B}_{\nu/5}(x_{i+1})$. Then $\overline{x_{\text{rand}} x_{\text{near}}} \subset \mathcal{F}$, where x_{near} is the nearest neighbor of x_{rand} in T.



Proof of Lemma 1

We show that $\|x_{\text{near}} - x_i\| \leq \delta$:

- $\|x_{\text{near}} - x_i\| \leq \|x_{\text{near}} - x_{\text{rand}}\| + \|x_{\text{rand}} - x_i\|$ (triangle inequality)
- $\|x_{\text{near}} - x_{\text{rand}}\| \leq \|x_i' - x_{\text{rand}}\|$ (since x_{near} is the nearest neighbor of x_{rand})
- $\|x_i' - x_{\text{rand}}\| \leq 3 \cdot \nu/5, \quad \|x_{\text{rand}} - x_i\| \leq 2 \cdot \nu/5$
- Therefore, $\|x_{\text{near}} - x_i\| \leq 5 \cdot \nu/5 = \nu \leq \delta \Rightarrow x_{\text{near}} \in \mathcal{B}_\delta(x_i)$
- Since $x_{\text{near}}, x_{\text{rand}} \in \mathcal{B}_\delta(x_i)$ then $\overline{x_{\text{rand}} x_{\text{near}}} \subset \mathcal{F}$ $\square$



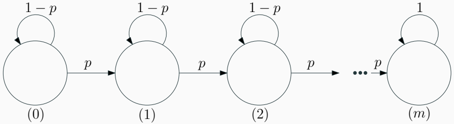
Note that $\|x_{\text{near}} - x_{\text{rand}}\| \leq 3 \cdot \nu/5 < \nu \leq \eta \quad \Rightarrow \quad x_{\text{new}} = x_{\text{rand}}$.

Theorem 1
The probability that RRT fails to reach x_{goal} from x_{init} after k iterations is $\leq ae^{-bk}$, for some $a, b \in \mathbb{R}_{>0}$

- Assume that $\mathcal{B}_{\nu/5}(x_i)$ contains an RRT vertex
- Let p be the prob. that in the next iteration an RRT vertex will be added to $\mathcal{B}_{\nu/5}(x_{i+1})$
- From Lemma 1, choosing $x_{\text{rand}} \in \mathcal{B}_{\nu/5}(x_{i+1})$ ensures this
- Since x_{rand} is drawn uniformly at random from $[0,1]^d$, we have that $p = |\mathcal{B}_{\nu/5}|/|[0,1]^d|$

Theorem: RRT is PC (cont.)

- To reach x_{goal} from x_{init} we need to repeat this step m times

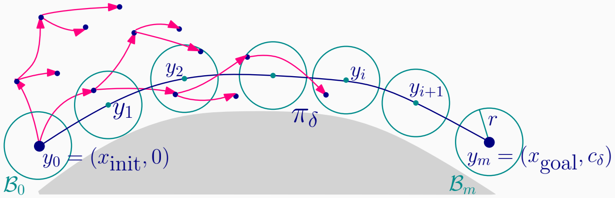


- Let X_k be the number of successes in k trials
- The prob. of failure: $\Pr[X_k < m] \leq \frac{m}{(k-1)!} k^m e^{-pk}$
- p, m are fixed and independent of k, therefore, $\Pr[X_k < m]$ decays to zero exponentially with k

$\square$

AO-RRT: an AO variant of RRT [Hauser and Zhou, 16]

- "Operates" in the state-cost space (a $(d+1)$-dimensional space)
- In [K. et al., 20] we show that the cost of the solution found approaches the optimal cost as $n \to \infty$



The End