MP-282
Dynamic Modeling and Control of Multirotor Aerial Vehicles
Chapter 10: Sampling-Based Path Planning

Prof. Dr. Davi Antônio dos Santos
Instituto Tecnológico de Aeronáutica
www.professordavisantos.com

São José dos Campos - SP
2020

Contents

Motivation . . .

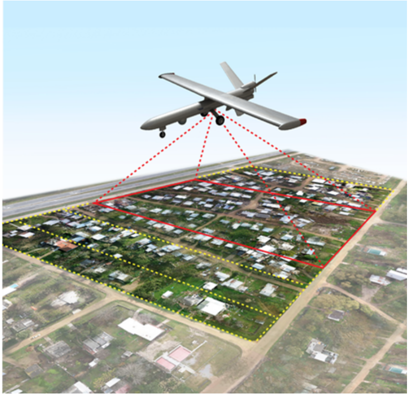
Motivation

Precise Agriculture

The MAV (or MAVs) needs to fly in a path that covers the field of interest.

Motivation

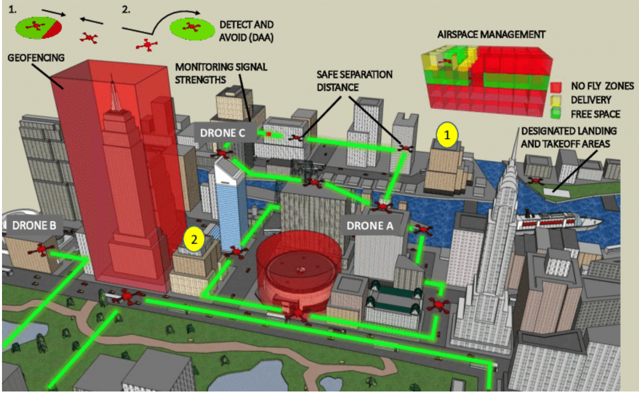
Remote Sensing in Urban Areas



The MAV (or MAVs) needs to fly in a path that covers the district of interest.

Motivation

Delivery in Urban Areas



The MAVs have to fly in a path among obstacles.

Definitions . . .

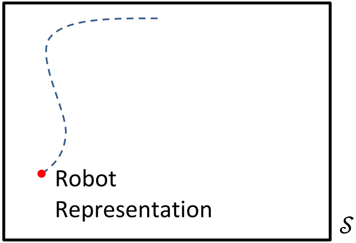
Definitions

Configuration Space

It is the set $\mathcal{S}$ where the robot can take poses.

Examples:

- For a robot with 3 DOFs of rotation and 3 DOFs of translation, $\mathcal{S} \subset \mathrm{SO}(3) \times \mathbb{R}^3$.
- For a point robot in $\mathbb{R}^2$ or $\mathbb{R}^3$, $\mathcal{S} \subset \mathbb{R}^2$ or $\mathcal{S} \subset \mathbb{R}^3$, respectively.

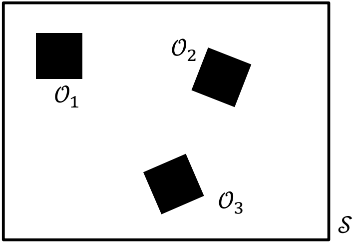


Definitions

Obstacle Region

From now on, consider $\mathcal{S} \subset \mathbb{R}^d$, with $d = 2$ or $d = 3$. Assume that each obstacle is a compact set $\mathcal{O}_i \subset \mathcal{S}$. The obstacle region is defined as

$$\mathcal{O} \triangleq \bigcup_{i=1}^{n_o} \mathcal{O}_i \qquad (1)$$

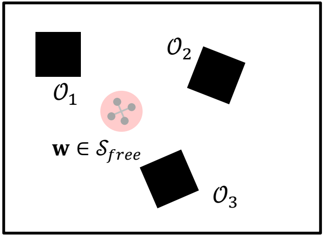


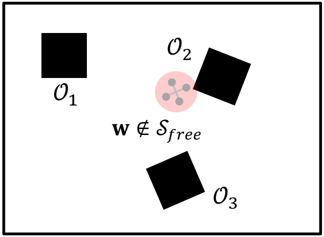
Definitions

Free Space

Assume that the MAV is limited inside a ball $\mathcal{B}_\delta(\mathbf{w})$ with radius δ centered at its configuration $\mathbf{w} \in \mathcal{S}$. This ball represents the MAV dimensions. The free space is defined as

$$\mathcal{S}_{free} \triangleq \left\{\mathbf{w} \in \mathcal{S} : \mathcal{B}_\delta(\mathbf{w}) \cap \mathcal{O} = \emptyset \ \text{ and } \ \mathcal{B}_\delta(\mathbf{w}) \cap \bar{\mathcal{S}} = \emptyset\right\} \qquad (2)$$





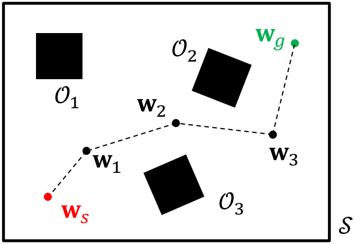
Definitions

Feasible Path

Given a starting pose $\mathbf{w}_s \in \mathcal{S}_{free}$ and a goal pose $\mathbf{w}_g \in \mathcal{S}_{free}$, a feasible path between them is a continuous function $\boldsymbol{\sigma} : [0, 1] \to \mathbb{R}^d$ such that

- $\boldsymbol{\sigma}(\tau) \in \mathcal{S}_{free}, \forall \tau \in [0, 1]$
- $\boldsymbol{\sigma}(0) = \mathbf{w}_s$ and $\boldsymbol{\sigma}(1) = \mathbf{w}_g$

In particular, we are interested in piecewise affine paths that can be parameterized by a finite sequence of waypoints $\{\mathbf{w}_i\}$, besides $\mathbf{w}_s$ and $\mathbf{w}_g$.

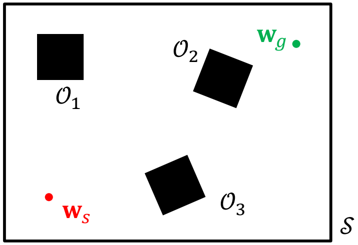


Definitions

Path Planning Problem

Consider a path planning scenario $\Sigma \triangleq (\mathcal{S}_{free}, \mathbf{w}_s, \mathbf{w}_g)$. The path planning problem is

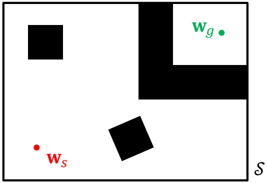
- to find a path $\boldsymbol{\sigma}$ in $\mathcal{S}_{free}$, if at least one exists or
- to report a failure to find a path in $\mathcal{S}_{free}$, if no one exists.



Definitions

Comments

- If $\mathcal{S}_{free}$ is not connected and $\mathbf{w}_s$ and $\mathbf{w}_g$ belong to different components of $\mathcal{S}_{free}$, then (it is clear that) the function $\boldsymbol{\sigma}$ does not exist.



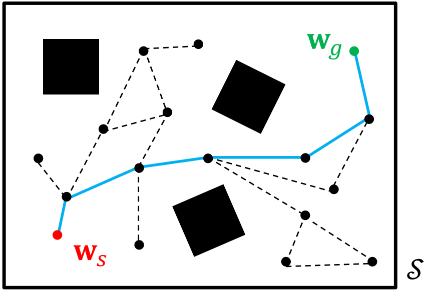
- The path planning problem is hard from the computational point of view and, therefore, its exact solution is impractical.
- The popular alternative is to use a sample to approximately (rather than exactly) represent $\mathcal{S}_{free}$.
- The most influential sampling-based approaches are the probabilistic roadmap (PRM) and the rapidly-exploring random tree (RRT).

Probabilistic Roadmap . . .

Probabilistic Roadmap

The probabilistic roadmap (PRM) methods are organized in two steps:

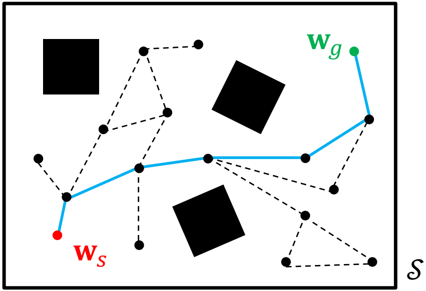
- Learning: $\mathcal{S}_{free}$ is randomly sampled and the resulting samples are used as nodes (or vertices) in a geometric graph, which is built so as to avoid obstacles.
- Search: $\mathbf{w}_s$ and $\mathbf{w}_g$ are inserted into the graph and the shortest path is found.



Probabilistic Roadmap

Probabilistic Roadmap

The aforementioned graph is called probabilistic roadmap and will be denoted by $\mathcal{G}(V, E)$, where V is the set of nodes (random samples) and E is the set of edges (local paths).



The learning and search steps are detailed in the sequel.

Probabilistic Roadmap

Learning Step

Consider first the following primitive procedures:

- $\mathbf{w} \leftarrow \textit{Sampling}(\mathcal{S}')$ takes a sample $\mathbf{w} \in \mathcal{S}'$ from a uniform probability distribution over $\mathcal{S}'$.
- $V_{near} \leftarrow \textit{Near}(V, \mathbf{w}, \mu)$ provides the set

$$V_{near} \triangleq \left\{ \bar{\mathbf{w}} \in V : \bar{\mathbf{w}} \in \mathcal{B}_{\mu}(\mathbf{w}), \bar{\mathbf{w}} \neq \mathbf{w} \right\}$$

- $Y \leftarrow \textit{NoCollision}(\mathbf{w}_1, \mathbf{w}_2)$ returns $Y = \texttt{true}$ if (the line segment) $(\mathbf{w}_1, \mathbf{w}_2) \in \mathcal{S}_{free}$ and $Y = \texttt{false}$ otherwise.
- $V_{nearest} \leftarrow \textit{Nearest}(V, \mathbf{w}, k)$ provides the set $V_{nearest}$, which contains the k nearest points of V from $\mathbf{w}$.

Probabilistic Roadmap

Algorithm 1. Learning Step.

Data: $\mathcal{S}_{free}$
Result: $\mathcal{G}(V, E)$

$V \leftarrow \emptyset, E \leftarrow \emptyset$
for $i = 1 : n$ **do**
  $\mathbf{w} \leftarrow Sampling(\mathcal{S}_{free})$
  $V(i) \leftarrow \{\mathbf{w}\}$
end
for $i = 1 : n$ **do**
  $V_{near} \leftarrow Near(V, V(i), \mu)$
  for *each* $\bar{\mathbf{w}} \in V_{near}$ **do**
    if $(V(i), \bar{\mathbf{w}}) \notin E$ and $NoCollision(V(i), \bar{\mathbf{w}})$ **then**
      $E \leftarrow E \cup \{(V(i), \bar{\mathbf{w}})\}$
    end
  end
end

Probabilistic Roadmap

Other Versions

There are many versions of Algorithm 1 trying to reduce its computational burden. Two of them are obtained by

1. replacing $V_{near} \leftarrow Near(V, V(i), \mu)$ by $V_{nearest} \leftarrow Nearest(V, V(i), k)$.
2. checking the points $\bar{\mathbf{w}} \in V_{near}$ in order of increasing distances to $V(i)$ and avoiding to include in E the edges that connect V(i) to the same component of $\mathcal{G}$.

Probabilistic Roadmap

Comments

- If the resulting roadmap has more than one (disconnected) components, it must be improved before proceeding to the search step.
- The disconnection occurs because of the existence of areas with low sampling probability (*e.g.*, narrow corridors). In fact, there are many different versions of the PRM aiming at improving sampling in such regions.

Probabilistic Roadmap

Search Step

Consider first the following primitive procedures:

- $\bar{\mathbf{w}} \leftarrow$ *NearestCollisionFree*$(V, \mathbf{w})$ return the nearest point $\bar{\mathbf{w}} \in V$ from $\mathbf{w}$ such that $(\bar{\mathbf{w}}, \mathbf{w}) \in \mathcal{S}_{free}$.
- $P \leftarrow$ *ShortestPath*$(\mathcal{G}(V, E), \mathbf{w}_s, \mathbf{w}_g)$ returns the shortest path $P \triangleq \{\mathbf{w}_s, \mathbf{w}_1, \ldots, \mathbf{w}_{n_w}, \mathbf{w}_g\}$ of $\mathcal{G}$ that connects $\mathbf{w}_s$ to $\mathbf{w}_g$.[1]

[1] For example, this procedure can be based on the Dijkstra's or A* algorithm.

Probabilistic Roadmap

Algorithm 2. Search Step.

Data: $\mathcal{G}(V, E)$, $\mathbf{w}_s$, $\mathbf{w}_g$
Result: P

$\mathbf{w}_1 \leftarrow \mathit{NearestCollisionFree}(V, \mathbf{w}_s)$
$\mathbf{w}_2 \leftarrow \mathit{NearestCollisionFree}(V, \mathbf{w}_g)$

$\bar{V} \leftarrow V \cup \left\{\mathbf{w}_s, \mathbf{w}_g\right\}$
$\bar{E} \leftarrow E \cup \left\{(\mathbf{w}_1, \mathbf{w}_s), (\mathbf{w}_2, \mathbf{w}_g)\right\}$

$P \leftarrow \mathit{ShortestPath}(\mathcal{G}(\bar{V}, \bar{E}), \mathbf{w}_s, \mathbf{w}_g)$

Comments

- Usually, the learning step takes much more time than the search one. Therefore, the PRM is more suitable for multiple query problems. For a single shot problem, the next method is a better choice.
- After running Algorithm 2, the resulting path P can be smoothed for eliminating useless motions. A simple post-processing algorithm is (to try) to replace parts of P containing more than two nodes by a straight segment.
- Assume that $\mathcal{S}_{free}$ is connected. One can show that if $n \to \infty$ the probability that Algorithm 1 together with Algorithm 2 will return a solution is one. One can also show that the probability of failure to find a path when one exists exponentially decays as $n \to \infty$.

Rapidly-Exploring Random Tree ...

Rapidly-Exploring Random Tree

New Scenario

Consider that the path planning problem now has the following scenario:

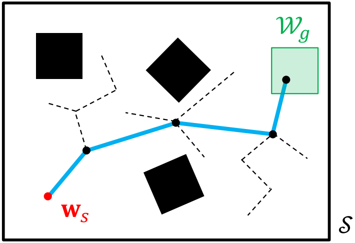
$$\Sigma = (\mathcal{S}_{free}, \mathbf{w}_s, \mathcal{W}_g)$$

where $\mathcal{W}_g \subset \mathcal{S}_{free}$ is a set, rather than a single pose. Therefore, the new problem is to find a path in $\mathcal{S}_{free}$ that starts at $\mathbf{w}_s$ and ends at any point in $\mathcal{W}_g$.

Rapidly-Exploring Random Tree

The idea of the Method

In its basic version, the rapidly-exploring random tree (RRT) algorithm incrementally builds a tree[2] of feasible paths starting from $\mathbf{w}_s$. As soon as the tree enters into $\mathcal{W}_g$, a path is found and the algorithm stops.



[2] In graph theory, a tree is a connected graph that has no cycles.

Rapidly-Exploring Random Tree

Algorithm 3. RRT (Original Version)

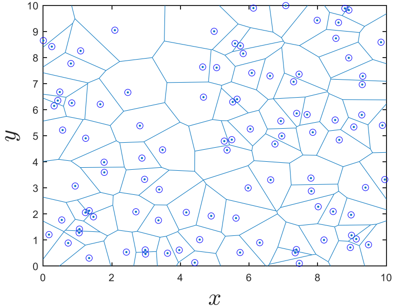
Data: $\mathbf{w}_s$, $\mathcal{W}_g$
Result: $\mathcal{G}(V, E)$

$V \leftarrow \{\mathbf{w}_s\}$, $E \leftarrow \emptyset$
for $i = 1 : n$ **do**
  $\mathbf{w} \leftarrow Sampling(\mathcal{S}_{free})$
  $\mathbf{w}_{nearest} \leftarrow Nearest(V, \mathbf{w}, 1)$
  $\mathbf{w}_{new} \leftarrow \mathbf{w}_{nearest} + \eta(\mathbf{w} - \mathbf{w}_{nearest})/\|\mathbf{w} - \mathbf{w}_{nearest}\|$ (or $\mathbf{w}_{new} \leftarrow \mathbf{w}$)
  if $NoObstacle(\mathbf{w}_{nearest}, \mathbf{w}_{new})$ **then**
    $V \leftarrow V \cup \{\mathbf{w}_{new}\}$
    $E \leftarrow E \cup \{(\mathbf{w}_{new}, \mathbf{w}_{nearest})\}$
    if $\mathbf{w}_{new} \in \mathcal{W}_g$ **then**
      return $\mathcal{G}(V, E)$
    end
  end
end

Rapidly-Exploring Random Tree

Comments

- The RRT methods are primarily aimed at single-query applications.
- The growth of the random tree is biased toward the unexplored areas of $\mathcal{S}_{free}$, as verified in the Voronoi diagram:



- The RRT can also be shown to be probabilistically complete and to have a probability of failure that dacays exponentially as $n \to \infty$ [3].

References . . .

References

- S. M. LaValle. Planning Algorithm. Cambridge University Press, 2006.
- H. Choset. Principles of Robot Motion: Theory, Algorithms, and Implementation. MIT Press, 2005.
- S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. The International Journal of Robotics Research, 30(7), 846–894, 2011.