Motion planning problem formulation

SIDRA Summer School,
Bertinoro 2023

Paolo Falcone

Department of Electrical Engineering,
Chalmers University of Technology,
Gothenburg, Sweden

Dipartimento di Ingegneria “Enzo Ferrari”,
Università di Modena e Reggio Emilia,
Modena

Lecture objectives

- Motion planning problem statement

Lecture objectives

- Motion planning problem statement
- Trajectory and path planning problems

Lecture objectives

- Motion planning problem statement
- Trajectory and path planning problems
- Randomized methods

Lecture objectives

- Motion planning problem statement
- Trajectory and path planning problems
- Randomized methods
- Artificial potential fields

Problem statement

Find the trajectory to track (path to follow) as the solution of the problem of minimizing a desired cost, while satisfying physical and design constraints.

Problem statement

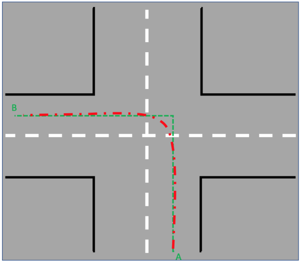
Find the trajectory to track (path to follow) as the solution of the problem of minimizing a desired cost, while satisfying physical and design constraints.

Designing the cost.

Problem statement

Find the trajectory to track (path to follow) as the solution of the problem of minimizing a desired cost, while satisfying physical and design constraints.

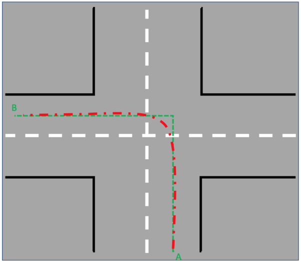
Designing the cost. In road transportation applications a “reference” path is likely to be available. E.g., the lane centerline of the *desired route*,

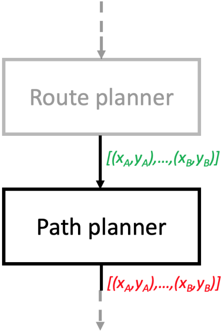


Problem statement

Find the trajectory to track (path to follow) as the solution of the problem of minimizing a desired cost, while satisfying physical and design constraints.

Designing the cost. In road transportation applications a "reference" path is likely to be available. E.g., the lane centerline of the *desired route*, which can be assumed to be given.

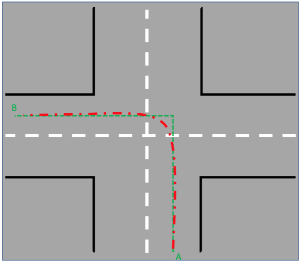


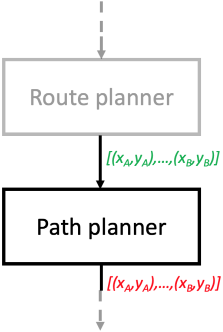


Problem statement

Find the trajectory to track (path to follow) as the solution of the problem of minimizing a desired cost, while satisfying physical and design constraints.

Designing the cost. In road transportation applications a "reference" path is likely to be available. E.g., the lane centerline of the *desired route*, which can be assumed to be given.





The cost should then be designed such that the *planned path* minimizes, in some sense, the deviation from the *reference path*.

Path or trajectory planning?

A ***path*** in $\in \mathbb{R}^2$ is a sequence of pairs $[(x_1, y_1), \ldots, (x_n, y_n)]$.

Path or trajectory planning?

A ***path*** in $\in \mathbb{R}^2$ is a sequence of pairs $[(x_1, y_1), \ldots, (x_n, y_n)]$.

A ***trajectory*** in $\in \mathbb{R}^2$ is a *timed* sequence of pairs $[(x_1(t), y_1(t)), \ldots, (x_n(t), y_n(t))]$.

Path or trajectory planning?

A ***path*** in $\in \mathbb{R}^2$ is a sequence of pairs $[(x_1, y_1), \ldots, (x_n, y_n)]$.

A ***trajectory*** in $\in \mathbb{R}^2$ is a *timed* sequence of pairs $[(x_1(t), y_1(t)), \ldots, (x_n(t), y_n(t))]$.

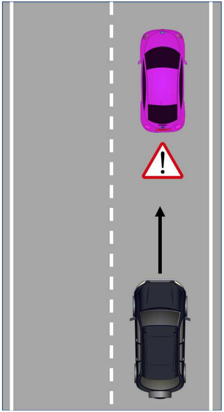
A path or trajectory is sought to plan the vehicle motion in static and dynamic environments, respectively.

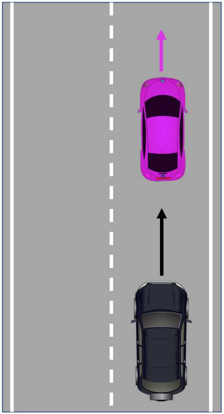
Path or trajectory planning?

A ***path*** in $\in \mathbb{R}^2$ is a sequence of pairs $[(x_1, y_1), \ldots, (x_n, y_n)]$.

A ***trajectory*** in $\in \mathbb{R}^2$ is a *timed* sequence of pairs $[(x_1(t), y_1(t)), \ldots, (x_n(t), y_n(t))]$.

A path or trajectory is sought to plan the vehicle motion in static and dynamic environments, respectively.





Randomized methods[a]

Problem formulation. Given a *configuration workspace* $C = C_{free} \bigcup C_{obs}$ and the points $x_{start},\ x_{goal} \in C_{free}$, find a function $f : [0, 1] \to C_{free}$, such that $f(0) = x_{start}$, $f(1) = x_{goal}$.

[a] *Randomized motion planning - A tutorial.* Stefano Carpin.

Randomized methods[a]

Problem formulation. Given a *configuration workspace* $C = C_{free} \bigcup C_{obs}$ and the points x_{start}, $x_{goal} \in C_{free}$, find a function $f : [0, 1] \rightarrow C_{free}$, such that $f(0) = x_{start}$, $f(1) = x_{goal}$.

Main Idea.

1. Explore the environment by (random) sampling.

[a] *Randomized motion planning - A tutorial.* Stefano Carpin.

Randomized methods[a]

Problem formulation. Given a *configuration workspace* $C = C_{free} \bigcup C_{obs}$ and the points x_{start}, $x_{goal} \in C_{free}$, find a function $f : [0, 1] \to C_{free}$, such that $f(0) = x_{start}$, $f(1) = x_{goal}$.

Main Idea.

1. Explore the environment by (random) sampling.
2. Add legit (e.g., safe) points/regions to the set of explored points/regions.

[a] *Randomized motion planning - A tutorial.* Stefano Carpin.

Randomized methods[a]

Problem formulation. Given a *configuration workspace* $C = C_{free} \bigcup C_{obs}$ and the points $x_{start},\ x_{goal} \in C_{free}$, find a function $f : [0, 1] \rightarrow C_{free}$, such that $f(0) = x_{start},\ f(1) = x_{goal}$.

Main Idea.

1. Explore the environment by (random) sampling.
2. Add legit (e.g., safe) points/regions to the set of explored points/regions.
3. Connect (find a path between) the new point/region with the existing ones.

[a] *Randomized motion planning - A tutorial.* Stefano Carpin.

Randomized methods[a]

Problem formulation. Given a *configuration workspace* $C = C_{free} \bigcup C_{obs}$ and the points $x_{start},\ x_{goal} \in C_{free}$, find a function $f : [0, 1] \rightarrow C_{free}$, such that $f(0) = x_{start},\ f(1) = x_{goal}$.

Main Idea.

1. Explore the environment by (random) sampling.
2. Add legit (e.g., safe) points/regions to the set of explored points/regions.
3. Connect (find a path between) the new point/region with the existing ones.
4. ***[Online]*** Given an origin x_{start} and a destination x_{goal}, online extract a path connecting them.

[a] *Randomized motion planning - A tutorial.* Stefano Carpin.

Randomized methods[a]

Problem formulation. Given a *configuration workspace* $C = C_{free} \bigcup C_{obs}$ and the points x_{start}, $x_{goal} \in C_{free}$, find a function $f : [0, 1] \to C_{free}$, such that $f(0) = x_{start}$, $f(1) = x_{goal}$.

Main Idea.

1. Explore the environment by (random) sampling.
2. Add legit (e.g., safe) points/regions to the set of explored points/regions.
3. Connect (find a path between) the new point/region with the existing ones.
4. ***[Online]*** Given an origin x_{start} and a destination x_{goal}, online extract a path connecting them.

Two main approaches

[a] *Randomized motion planning - A tutorial.* Stefano Carpin.

Randomized methods[a]

Problem formulation. Given a *configuration workspace* $C = C_{free} \bigcup C_{obs}$ and the points x_{start}, $x_{goal} \in C_{free}$, find a function $f : [0, 1] \to C_{free}$, such that $f(0) = x_{start}$, $f(1) = x_{goal}$.

Main Idea.

1. Explore the environment by (random) sampling.
2. Add legit (e.g., safe) points/regions to the set of explored points/regions.
3. Connect (find a path between) the new point/region with the existing ones.
4. ***[Online]*** Given an origin x_{start} and a destination x_{goal}, online extract a path connecting them.

Two main approaches

- *probabilistic roadmaps*. Data organized in a *graph*.

[a] *Randomized motion planning - A tutorial.* Stefano Carpin.

Randomized methods[a]

Problem formulation. Given a *configuration workspace* $C = C_{free} \bigcup C_{obs}$ and the points $x_{start}, x_{goal} \in C_{free}$, find a function $f : [0,1] \rightarrow C_{free}$, such that $f(0) = x_{start}$, $f(1) = x_{goal}$.

Main Idea.

1. Explore the environment by (random) sampling.
2. Add legit (e.g., safe) points/regions to the set of explored points/regions.
3. Connect (find a path between) the new point/region with the existing ones.
4. ***[Online]*** Given an origin x_{start} and a destination x_{goal}, online extract a path connecting them.

Two main approaches

- *probabilistic roadmaps*. Data organized in a *graph*.
- *rapidly exploring random trees (RRT)*. Data organized in a *tree*.

[a] *Randomized motion planning - A tutorial.* Stefano Carpin.

Randomized methods. Probabilistic roadmaps

1. ***Learning stage.*** An *undirected graph* $G = (V, E)$ is built by exploration.

2. ***Query stage.*** Extract a path from $G = (V, E)$ connecting x_{start} to x_{goal}.

Randomized methods. Probabilistic roadmaps

1. ***Learning stage.*** An *undirected graph* $G = (V, E)$ is built by exploration.
    1. Randomly pick a point c in C,
2. ***Query stage.*** Extract a path from $G = (V, E)$ connecting x_{start} to x_{goal}.

Randomized methods. Probabilistic roadmaps

1. ***Learning stage.*** An *undirected graph* $G = (V, E)$ is built by exploration.
   1. Randomly pick a point c in C,
   2. if $c \in C_{free}$ then add it to V,

2. ***Query stage.*** Extract a path from $G = (V, E)$ connecting x_{start} to x_{goal}.

Randomized methods. Probabilistic roadmaps

1. ***Learning stage.*** An *undirected graph* $G = (V, E)$ is built by exploration.
   1. Randomly pick a point c in C,
   2. if $c \in C_{free}$ then add it to V,
   3. select the neighbors of c in V as those nodes in V within a distance M from c.

2. ***Query stage.*** Extract a path from $G = (V, E)$ connecting x_{start} to x_{goal}.

Randomized methods. Probabilistic roadmaps

1. ***Learning stage.*** An *undirected graph* $G = (V, E)$ is built by exploration.
    1. Randomly pick a point c in C,
    2. if $c \in C_{free}$ then add it to V,
    3. select the neighbors of c in V as those nodes in V within a distance M from c.
    4. connect c to the neighbors in V with a "simple" path planner (straight segment within C_{free}).

2. ***Query stage.*** Extract a path from $G = (V, E)$ connecting x_{start} to x_{goal}.

Randomized methods. Probabilistic roadmaps

1. ***Learning stage.*** An *undirected graph* $G = (V, E)$ is built by exploration.
    1. Randomly pick a point c in C,
    2. if $c \in C_{free}$ then add it to V,
    3. select the neighbors of c in V as those nodes in V within a distance M from c.
    4. connect c to the neighbors in V with a "simple" path planner (straight segment within C_{free}).
    5. a refinement step might be performed to more densely sample critical regions.
2. ***Query stage.*** Extract a path from $G = (V, E)$ connecting x_{start} to x_{goal}.

Randomized methods. Probabilistic roadmaps

1. ***Learning stage.*** An *undirected graph* $G = (V, E)$ is built by exploration.
    1. Randomly pick a point c in C,
    2. if $c \in C_{free}$ then add it to V,
    3. select the neighbors of c in V as those nodes in V within a distance M from c.
    4. connect c to the neighbors in V with a "simple" path planner (straight segment within C_{free}).
    5. a refinement step might be performed to more densely sample critical regions.
2. ***Query stage.*** Extract a path from $G = (V, E)$ connecting x_{start} to x_{goal}.
    1. Connect x_{start} and x_{goal} to the nodes in V. Report failure if it is not possible,

Randomized methods. Probabilistic roadmaps

1. ***Learning stage.*** An *undirected graph* $G = (V, E)$ is built by exploration.
    1. Randomly pick a point c in C,
    2. if $c \in C_{free}$ then add it to V,
    3. select the neighbors of c in V as those nodes in V within a distance M from c.
    4. connect c to the neighbors in V with a "simple" path planner (straight segment within C_{free}).
    5. a refinement step might be performed to more densely sample critical regions.
2. ***Query stage.*** Extract a path from $G = (V, E)$ connecting x_{start} to x_{goal}.
    1. Connect x_{start} and x_{goal} to the nodes in V. Report failure if it is not possible,
    2. be v_s and v_g the nodes connected to x_{start} and x_{goal}, respectively. If a path connecting v_s and v_g return it, otherwise return failure.

Randomized methods. Probabilistic roadmaps

1. ***Learning stage.*** An *undirected graph* $G = (V, E)$ is built by exploration.
    1. Randomly pick a point c in C,
    2. if $c \in C_{free}$ then add it to V,
    3. select the neighbors of c in V as those nodes in V within a distance M from c.
    4. connect c to the neighbors in V with a "simple" path planner (straight segment within C_{free}).
    5. a refinement step might be performed to more densely sample critical regions.
2. ***Query stage.*** Extract a path from $G = (V, E)$ connecting x_{start} to x_{goal}.
    1. Connect x_{start} and x_{goal} to the nodes in V. Report failure if it is not possible,
    2. be v_s and v_g the nodes connected to x_{start} and x_{goal}, respectively. If a path connecting v_s and v_g return it, otherwise return failure.

Remarks.

Randomized methods. Probabilistic roadmaps

1. ***Learning stage.*** An *undirected graph* $G = (V, E)$ is built by exploration.
   1. Randomly pick a point c in C,
   2. if $c \in C_{free}$ then add it to V,
   3. select the neighbors of c in V as those nodes in V within a distance M from c.
   4. connect c to the neighbors in V with a "simple" path planner (straight segment within C_{free}).
   5. a refinement step might be performed to more densely sample critical regions.
2. ***Query stage.*** Extract a path from $G = (V, E)$ connecting x_{start} to x_{goal}.
   1. Connect x_{start} and x_{goal} to the nodes in V. Report failure if it is not possible,
   2. be v_s and v_g the nodes connected to x_{start} and x_{goal}, respectively. If a path connecting v_s and v_g return it, otherwise return failure.

Remarks.

1. Learning stage can be expensive. Worth if it supports multiple queries,

Randomized methods. Probabilistic roadmaps

1. ***Learning stage.*** An *undirected graph* $G = (V, E)$ is built by exploration.
   1. Randomly pick a point c in C,
   2. if $c \in C_{free}$ then add it to V,
   3. select the neighbors of c in V as those nodes in V within a distance M from c.
   4. connect c to the neighbors in V with a "simple" path planner (straight segment within C_{free}).
   5. a refinement step might be performed to more densely sample critical regions.
2. ***Query stage.*** Extract a path from $G = (V, E)$ connecting x_{start} to x_{goal}.
   1. Connect x_{start} and x_{goal} to the nodes in V. Report failure if it is not possible,
   2. be v_s and v_g the nodes connected to x_{start} and x_{goal}, respectively. If a path connecting v_s and v_g return it, otherwise return failure.

Remarks.

1. Learning stage can be expensive. Worth if it supports multiple queries,
2. not suitable for dynamic environments,

Randomized methods. Probabilistic roadmaps

1. ***Learning stage.*** An *undirected graph* $G = (V, E)$ is built by exploration.
    1. Randomly pick a point c in C,
    2. if $c \in C_{free}$ then add it to V,
    3. select the neighbors of c in V as those nodes in V within a distance M from c.
    4. connect c to the neighbors in V with a "simple" path planner (straight segment within C_{free}).
    5. a refinement step might be performed to more densely sample critical regions.
2. ***Query stage.*** Extract a path from $G = (V, E)$ connecting x_{start} to x_{goal}.
    1. Connect x_{start} and x_{goal} to the nodes in V. Report failure if it is not possible,
    2. be v_s and v_g the nodes connected to x_{start} and x_{goal}, respectively. If a path connecting v_s and v_g return it, otherwise return failure.

Remarks.

1. Learning stage can be expensive. Worth if it supports multiple queries,
2. not suitable for dynamic environments,
3. simple (uniform) sampling policies may leave narrow regions poorly explored.

Randomized methods. RRTs

C_{free} is randomly explored and a tree is built.

Randomized methods. RRTs

C_{free} is randomly explored and a tree is built.

1. Starting form the current tree, pick a random configuration x_{rand} in C_{free},

Randomized methods. RRTs

C_{free} is randomly explored and a tree is built.

1. Starting form the current tree, pick a random configuration x_{rand} in C_{free},
2. find the nearest node x_{near} in the tree,

Randomized methods. RRTs

C_{free} is randomly explored and a tree is built.

1. Starting form the current tree, pick a random configuration x_{rand} in C_{free},
2. find the nearest node x_{near} in the tree,
3. from the nearest node find the control input to the robot (planner) that leads the state x as close as possible to x_{rand},

Randomized methods. RRTs

C_{free} is randomly explored and a tree is built.

1. Starting form the current tree, pick a random configuration x_{rand} in C_{free},
2. find the nearest node x_{near} in the tree,
3. from the nearest node find the control input to the robot (planner) that leads the state x as close as possible to x_{rand},
4. if such a control input exists and $x_{rand} = x$, the state x_{rand} is added to the tree,

Randomized methods. RRTs

C_{free} is randomly explored and a tree is built.

1. Starting form the current tree, pick a random configuration x_{rand} in C_{free},
2. find the nearest node x_{near} in the tree,
3. from the nearest node find the control input to the robot (planner) that leads the state x as close as possible to x_{rand},
4. if such a control input exists and $x_{rand} = x$, the state x_{rand} is added to the tree,
5. if $x_{rand} \neq x$, then x is added to the tree, which grows,

Randomized methods. RRTs

C_{free} is randomly explored and a tree is built.

1. Starting form the current tree, pick a random configuration x_{rand} in C_{free},
2. find the nearest node x_{near} in the tree,
3. from the nearest node find the control input to the robot (planner) that leads the state x as close as possible to x_{rand},
4. if such a control input exists and $x_{rand} = x$, the state x_{rand} is added to the tree,
5. if $x_{rand} \neq x$, then x is added to the tree, which grows,
6. otherwise, the algorithm is trapped.

Randomized methods. RRTs

C_{free} is randomly explored and a tree is built.

1. Starting form the current tree, pick a random configuration x_{rand} in C_{free},
2. find the nearest node x_{near} in the tree,
3. from the nearest node find the control input to the robot (planner) that leads the state x as close as possible to x_{rand},
4. if such a control input exists and $x_{rand} = x$, the state x_{rand} is added to the tree,
5. if $x_{rand} \neq x$, then x is added to the tree, which grows,
6. otherwise, the algorithm is trapped.

The tree can grow faster if the tree grows from both x_{start} to x_{goal}.

Artificial potential fields

Idea. Build a function to be minimized, which decreases toward the goal position and increases close to the obstacles.

Artificial potential fields

Idea. Build a function to be minimized, which decreases toward the goal position and increases close to the obstacles.

The potential function is $U = U_g + U_o$, where

Artificial potential fields

Idea. Build a function to be minimized, which decreases toward the goal position and increases close to the obstacles.

The potential function is $U = U_g + U_o$, where the term U_g *attracts* the robot toward the goal position x_{goal},

$$U_g(x) = \frac{1}{2}k(x - x_{goal})^T(x - x_{goal}).$$

Artificial potential fields

Idea. Build a function to be minimized, which decreases toward the goal position and increases close to the obstacles.

The potential function is $U = U_g + U_o$, where the term U_g *attracts* the robot toward the goal position x_{goal},

$$U_g(x) = \frac{1}{2}k(x - x_{goal})^T(x - x_{goal}).$$

The minimization of $U_g(x)$ results in an attractive force

$$f_g(x) = -\nabla U_g(x) = k(x - x_{goal}).$$

Artificial potential fields

Idea. Build a function to be minimized, which decreases toward the goal position and increases close to the obstacles.

The potential function is $U = U_g + U_o$, where the term U_g *attracts* the robot toward the goal position x_{goal},

$$U_g(x) = \frac{1}{2}k(x - x_{goal})^T(x - x_{goal}).$$

The minimization of $U_g(x)$ results in an attractive force

$$f_g(x) = -\nabla U_g(x) = k(x - x_{goal}).$$

Similarly, a repulsive force $f_o(x)$ is generate by minimizing U_o defined as

$$U_o(x) = \begin{cases} \frac{k}{\alpha}\left(\frac{1}{d(x)} - \frac{1}{d_0}\right)^{\alpha}, & \text{if } d(x) \leq d_0, \\ 0, & \text{if } d(x) > d_0, \end{cases}$$

Artificial potential fields

Idea. Build a function to be minimized, which decreases toward the goal position and increases close to the obstacles.

The potential function is $U = U_g + U_o$, where the term U_g *attracts* the robot toward the goal position x_{goal},

$$U_g(x) = \frac{1}{2}k(x - x_{goal})^T(x - x_{goal}).$$

The minimization of $U_g(x)$ results in an attractive force

$$f_g(x) = -\nabla U_g(x) = k(x - x_{goal}).$$

Similarly, a repulsive force $f_o(x)$ is generate by minimizing U_o defined as

$$U_o(x) = \begin{cases} \frac{k}{\alpha}\left(\frac{1}{d(x)} - \frac{1}{d_0}\right)^{\alpha}, & \text{if } d(x) \leq d_0, \\ 0, & \text{if } d(x) > d_0, \end{cases}$$

where $d(x) = \min_{\bar{x} \in C_{obs}} \|x - \bar{x}\|$.

Artificial potential fields

The repulsive force generated by the obstacles is

$$f_o(x) = -\nabla U_o(x) = \begin{cases} \frac{k}{d^2(x)}\left(\frac{1}{d(x)} - \frac{1}{d_0}\right)^{\alpha-1}\nabla d(x), & \text{if } d(x) \leq d_0, \\ 0, & \text{if } d(x) > d_0. \end{cases}$$

Artificial potential fields

The repulsive force generated by the obstacles is

$$f_o(x) = -\nabla U_o(x) = \begin{cases} \frac{k}{d^2(x)} \left(\frac{1}{d(x)} - \frac{1}{d_0} \right)^{\alpha-1} \nabla d(x), & \text{if } d(x) \leq d_0, \\ 0, & \text{if } d(x) > d_0. \end{cases}$$

Main issue. *Local minima*. The robot may get stuck in a local minimum introduced by U_o.

Artificial potential fields

The repulsive force generated by the obstacles is

$$f_o(x) = -\nabla U_o(x) = \begin{cases} \frac{k}{d^2(x)} \left(\frac{1}{d(x)} - \frac{1}{d_0} \right)^{\alpha-1} \nabla d(x), & \text{if } d(x) \leq d_0, \\ 0, & \text{if } d(x) > d_0. \end{cases}$$

Main issue. *Local minima*. The robot may get stuck in a local minimum introduced by U_o. The potential function could be modified to remove the local minima, provided that the environment is known.

Actuator limitations and design constraints are not included in the force generation. The forces f_g, f_o could be limited.