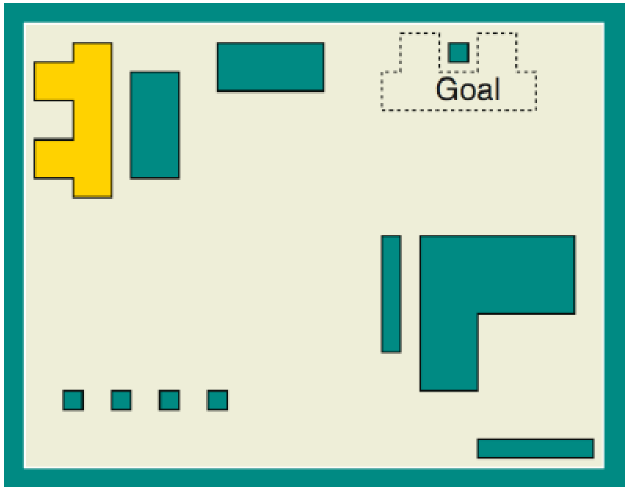
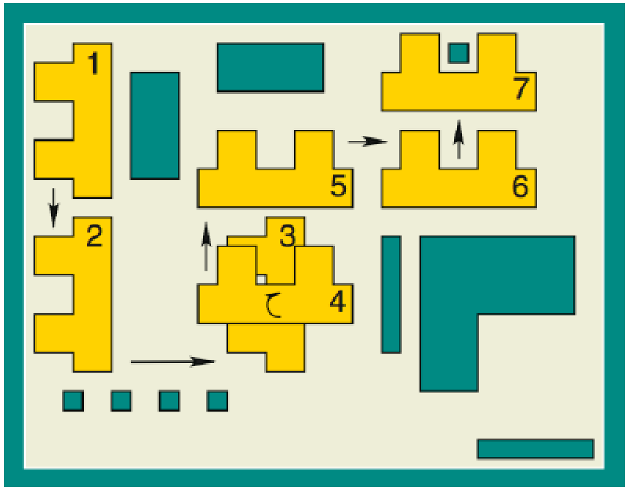
Обзор методов **Motion Planning**.

Введение





Формулировка задачи

Начальные данные:

-Исходная позиция робота

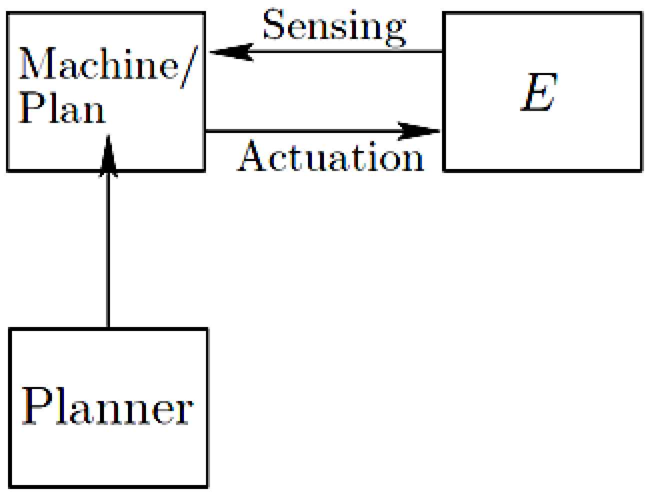
-Конечная позиция робота

-Описание робота и возможных манипуляций

-Описание области, по которой перемещается робот

Построить путь из начальной позиции в конечную, который обеспечивает обход препятствий.

Введение



Machine, interacts with the environment, E, through sensing and actuation.

Введение

Критерии при планировании

1. Feasibility (осуществимость)

Find a plan that causes arrival at a goal state, regardless of it’s efficiency.

2. Optimality (оптимальность)

Find a feasible plan that optimizes performance in some care-fully specified manner, in addition to arriving in a goal state.

(*Время, Энергия…)*

Обзор методов

1. Дискретное планирование

2. Sampling-based планирование (probabilistic planning, randomized planning)

3. Combinatorial planning (Комбинаторное планирование)

4. Методы основанные на использовании потенциальных полей

Дискретное планирование

1. Набор состояний ***x*** $x \in X$

2. Набор действий ***u*** $u \in U$

3. Transition function:

$$x' = f(x, u)$$

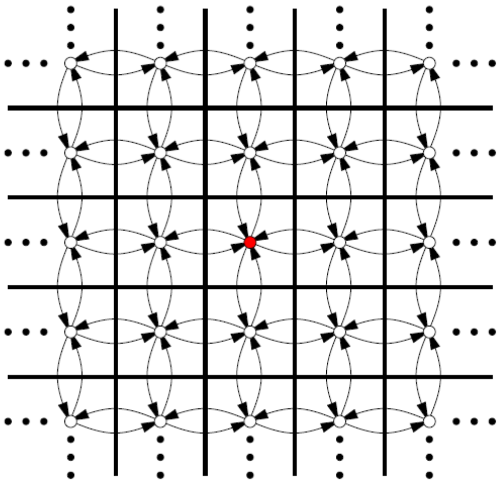
4. Начальное состояние x_{init}

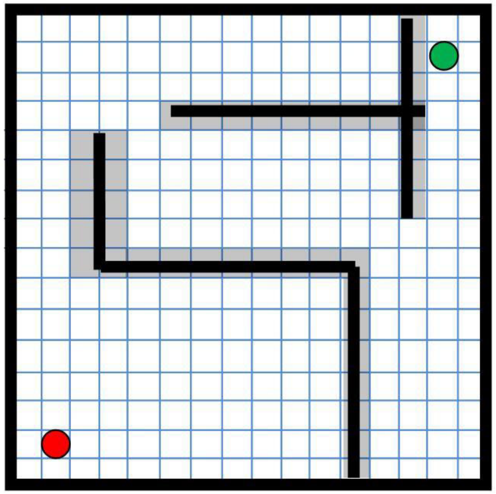
5. Конечное состояние x_G

$$u_1, u_1, \ldots, u_n : \; x_{init} \rightarrow x_G$$

Examples of Discrete Planning

Moving on a 2D Grid



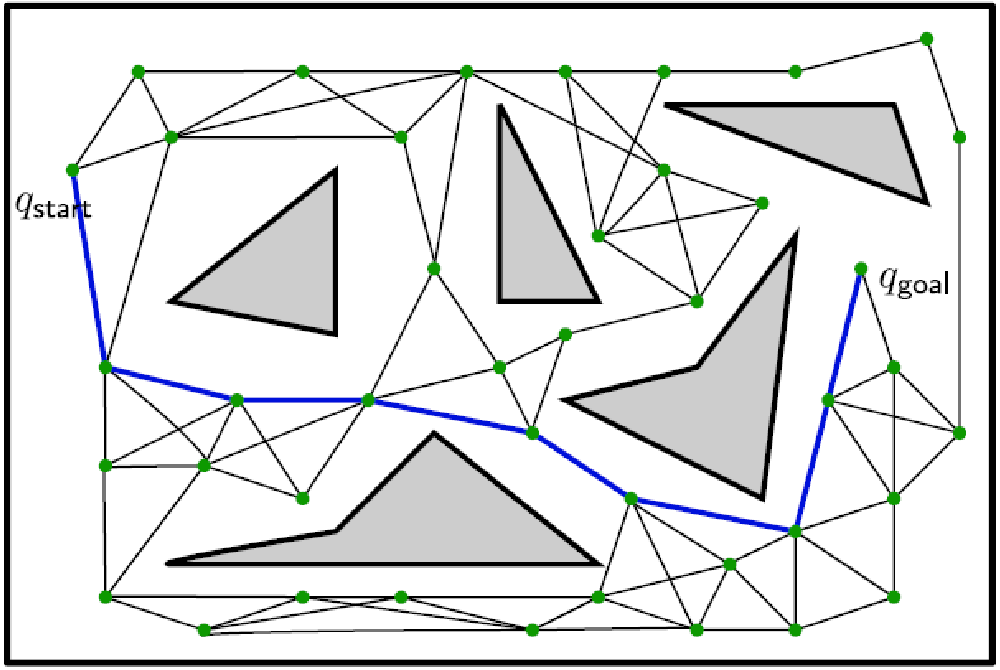


$$U = \{(0,1),(0,-1),(1,0),(-1,0)\}$$

$$f(x,u) = x + u$$

Examples of Discrete Planning

RoadMap



Использовать алгоритм поиска на графе для построения пути от начальной точки до конечной.

Алгоритмы поиска на графе

Неинформированные алгоритмы поиска:

1. **Обход в ширину** (Breadth-first search)

2. **Обход в глубину** (Depth-first search)

3. **Алгоритм Дейкстры** (Dijkstra's algorithm)

Информированные алгоритмы поиска:

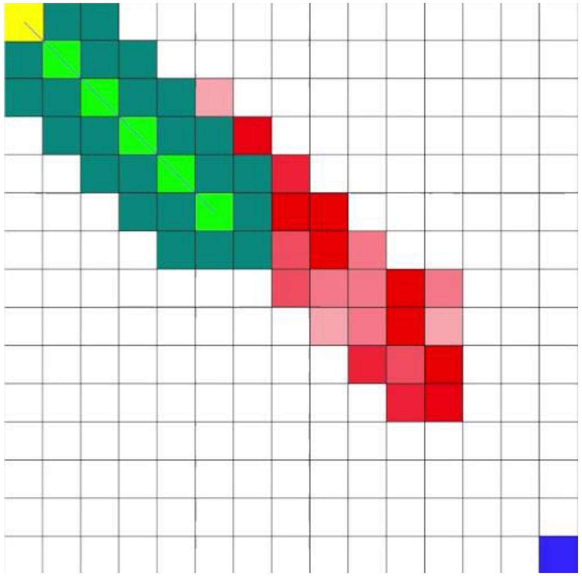
4. **Best-first search**

5. **Алгоритм A*** (A star)

Best-first search

Поиск «лучший — первый» — алгоритм поиска, который исследует граф путём расширения наиболее перспективных узлов, выбираемых в соответствии с указанным правилом.

```
Best-First-Search(Grah g, Node start)
    1) Create an empty PriorityQueue
       PriorityQueue pq;
    2) Insert "start" in pq.
       pq.insert(start)
    3) Until PriorityQueue is empty
          u = PriorityQueue.DeleteMin
          If u is the goal
             Exit
          Else
             Foreach neighbor v of u
                If v "Unvisited"
                    Mark v "Visited"
                    pq.insert(v)
             Mark v "Examined"
End procedure
```



$$f(v) = dist(v, v_{Goal})$$

Алгоритм A*

Алгоритм A* (A star**)** - алгоритм поиска на графе, который находит маршрут **<u>с наименьшей стоимостью</u>** от начальной вершины к конечной.

Алгоритм поиска A* является примером оптимального поиска «лучший — первый».

Функция для оценки узла в очереди:

$$f(v) = g(v) + h(v)$$

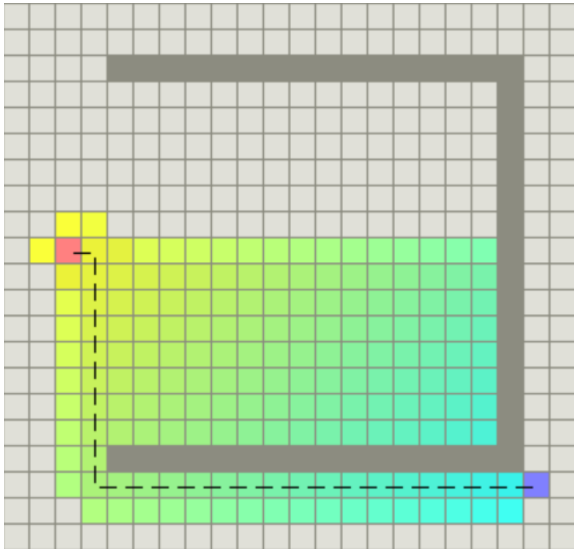
g(v) - стоимость достижения вершины **v** из начальной вершины.
h(v) - эвристическое приближение стоимости пути от вершины **v** до цели.

Алгоритм A*

Функция h(v) должна быть **допустимой эвристической оценкой**.

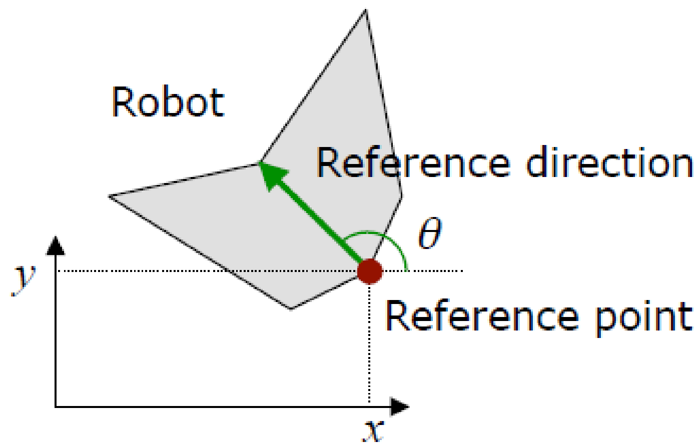
Говорят, что **эвристическая оценка** h(v) **допустима**, если для любой вершины v значение h(v) меньше или равно весу кратчайшего пути от v до цели.

Например, для задачи маршрутизации h(v) может представлять собой расстояние до цели по прямой линии.



Configuration Space

Rigid-body robot example



3-parameter representation:

$$q = (x, y, \theta)$$

In 3D, q would be of the form *(x,y,z,α,β,γ)*

Configuration Space

If the robot has n degrees of freedom, the set of transformations is usually a manifold of dimension n.
This manifold is called the **configuration space** of the robot, and it's name is often shortened to ***C-space***.

Введем обозначения: World W, Robot A, Obstacle O

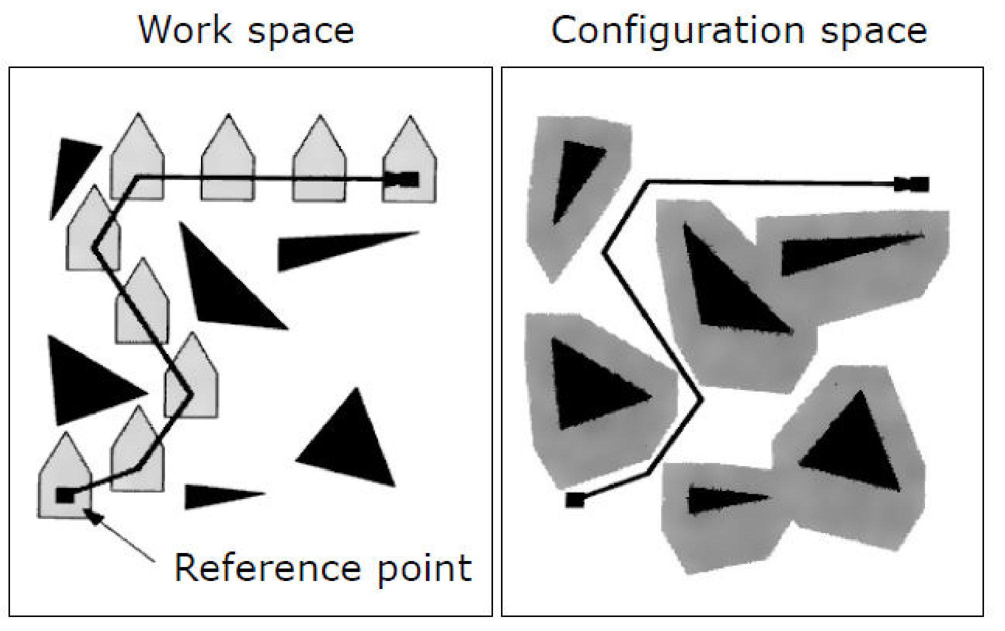
Obstacle region $C_{obs} \subseteq C$ **определяется как**

$$C_{obs} = \{q \in C \mid A(q) \cap O \neq \varnothing\}$$

Free space: $C_{free} = C \setminus C_{obs}$

Configuration Space

Example: polygonal robot, translation only



C_{free} is obtained by sliding the robot along the edge of the obstacle regions

Definition of basic motion planning

1. A world **W**

2. Obstacles **O**

3. Robot **A**

4. The **configuration space C** determined by specifying the set of all possible transformations that may be applied to the robot.

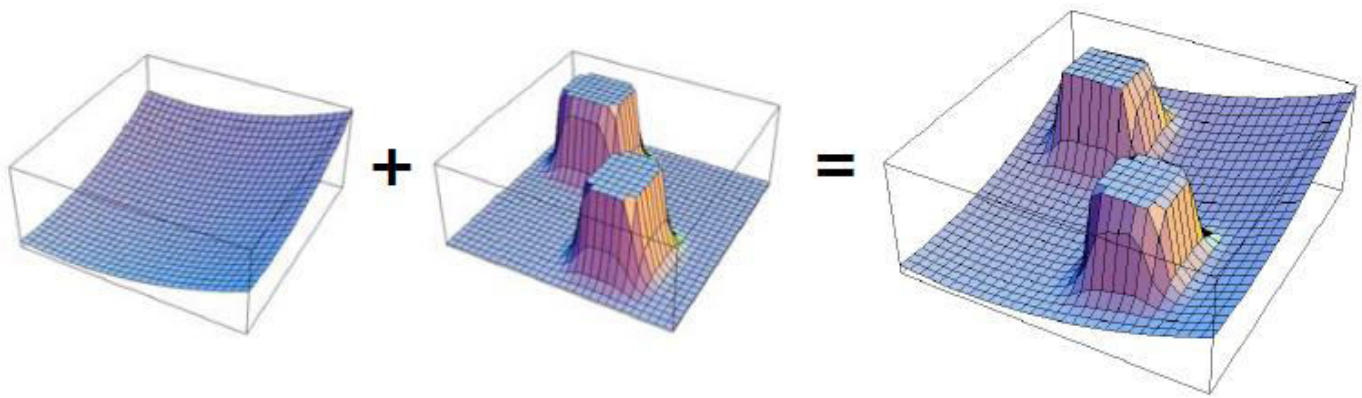
4. Начальная и конечная позиция

5. Построить непрерывный путь в $C_{free} = C \setminus C_{obs}$ соединяющий начальную и конечную позицию.

Potential Fields Methods

Potential Field:

$$U(\boldsymbol{q}) = U_{\mathrm{att}}(\boldsymbol{q}) + U_{\mathrm{rep}}(\boldsymbol{q})$$



Force:

$$F(\boldsymbol{q}) = -\nabla U(\boldsymbol{q}) = -\nabla U_{\mathrm{att}}(\boldsymbol{q}) - \nabla U_{\mathrm{rep}}(\boldsymbol{q})$$

Potential Fields Methods

Distance to goal:

$$\rho_f(\boldsymbol{q}) = ||\boldsymbol{q} - \boldsymbol{q}_{\text{final}}||$$

Field that grows quadratically with the distance to goal:

$$U_{\text{att}}(\boldsymbol{q}) = \frac{1}{2}\zeta\rho_f^2(\boldsymbol{q})$$

Potential Fields Methods

Combine the quadratic and conic potentials

$$U_{\text{att}}(\boldsymbol{q}) = \begin{cases} \frac{1}{2}\zeta\rho_f^2(\boldsymbol{q}) & : \quad \rho_f(\boldsymbol{q}) \leq d \\ d\zeta\rho_f(\boldsymbol{q}) - \frac{1}{2}\zeta d^2 & : \quad \rho_f(\boldsymbol{q}) > d \end{cases}$$

Force:

$$F_{\text{att}}(q) = -\nabla U_{\text{att}}(\boldsymbol{q}) = \begin{cases} -\zeta(\boldsymbol{q} - \boldsymbol{q}_{\text{final}}) & : \quad \rho_f(\boldsymbol{q}) \leq d \\ -\dfrac{d\zeta(\boldsymbol{q} - \boldsymbol{q}_{\text{final}})}{\rho_f(\boldsymbol{q})} & : \quad \rho_f(\boldsymbol{q}) > d \end{cases}$$

Potential Fields Methods

The Repulsive field

One way to achieve this is to define a potential that goes to infinity at obstacle boundaries, and drops to zero at a certain distance from the obstacle.

$$U_{\text{rep}}(\boldsymbol{q}) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho(\boldsymbol{q})} - \frac{1}{\rho_0}\right)^2 & : \quad \rho(\boldsymbol{q}) \leq \rho_0 \\ 0 & : \quad \rho(\boldsymbol{q}) > \rho_0 \end{cases}$$

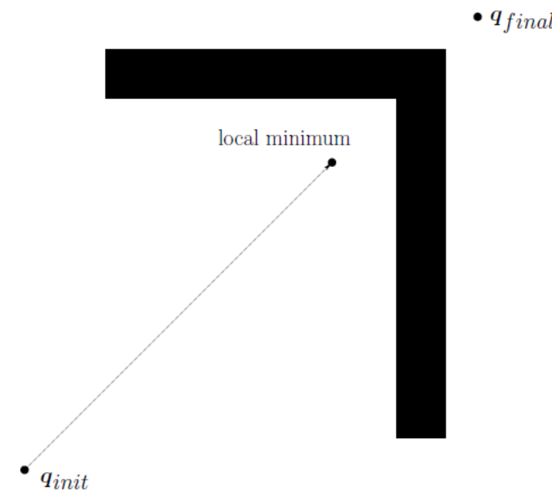
ρ(q) - shortest distance from q to obstacle

Force:

$$F_{\text{rep}}(\boldsymbol{q}) = \begin{cases} \eta\left(\frac{1}{\rho(\boldsymbol{q})} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2(\boldsymbol{q})}\nabla\rho(\boldsymbol{q}) & : \quad \rho(\boldsymbol{q}) \leq \rho_0 \\ 0 & : \quad \rho(\boldsymbol{q}) > \rho_0 \end{cases}$$

Potential Fields Methods

Main problems: robot gets stuck in **local minima.**



Using Random Motions to Escape Local Minima:

1. $\boldsymbol{q}^0 \leftarrow \boldsymbol{q}_{\text{init}}, \; i \leftarrow 0$
2. **IF** $\boldsymbol{q}^i \neq \boldsymbol{q}_{\text{final}}$

   $$\boldsymbol{q}^{i+1} \leftarrow \boldsymbol{q}^i + \alpha^i \frac{F(\boldsymbol{q}^i)}{||F(\boldsymbol{q}^i)||}$$

   $$i \leftarrow i + 1$$

   ELSE return $< \boldsymbol{q}^0, \boldsymbol{q}^1 \cdots \boldsymbol{q}^i >$
3. **IF** stuck in a local minimum

   execute a random walk, ending at $\boldsymbol{q}'$

   $$\boldsymbol{q}^{i+1} \leftarrow \boldsymbol{q}'$$
4. **GO TO** 2

Trajectory Planning

Условие для задания координат:

$$q(t_i) = q_i$$

Условие для задания скоростей:

$$\dot{q}(t_i) = v_i$$

Условие для задания ускорения:

$$\ddot{q}(t_i) = a_i$$

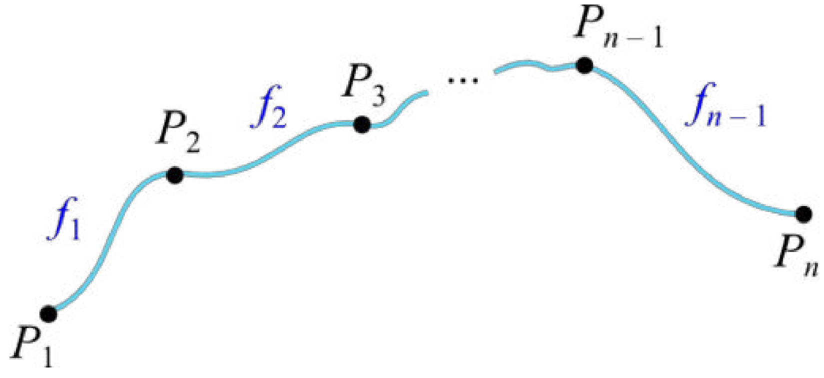
Интерполяция

1. Интерполяционный полином

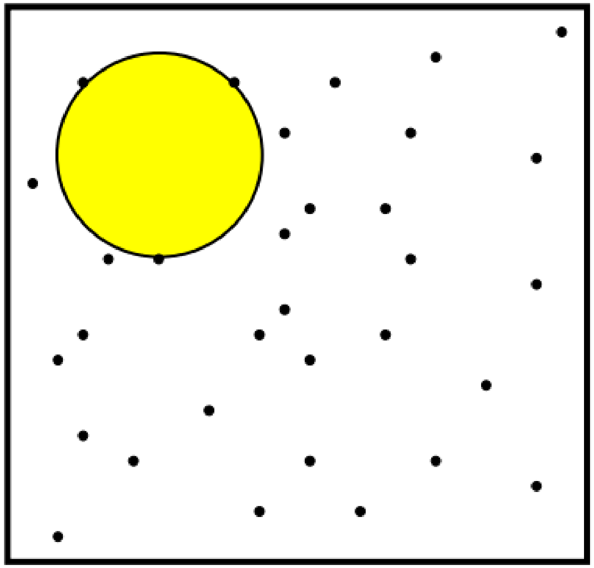
2. Сплайн

(обеспечение непрерывности скорости)

$$f_i(t) = a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0,\ t_i \le t < t_{i+1}$$



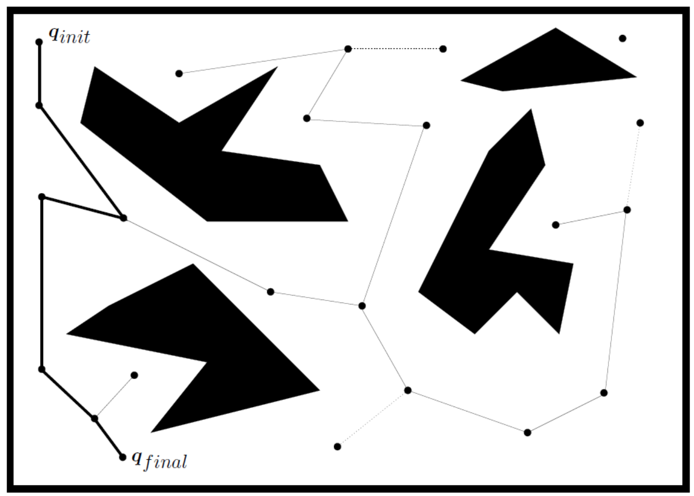
Sampling-Based Motion Planning



Random Sampling: *Построение множества из N случайных точек в* $\boldsymbol{C_{free}}$

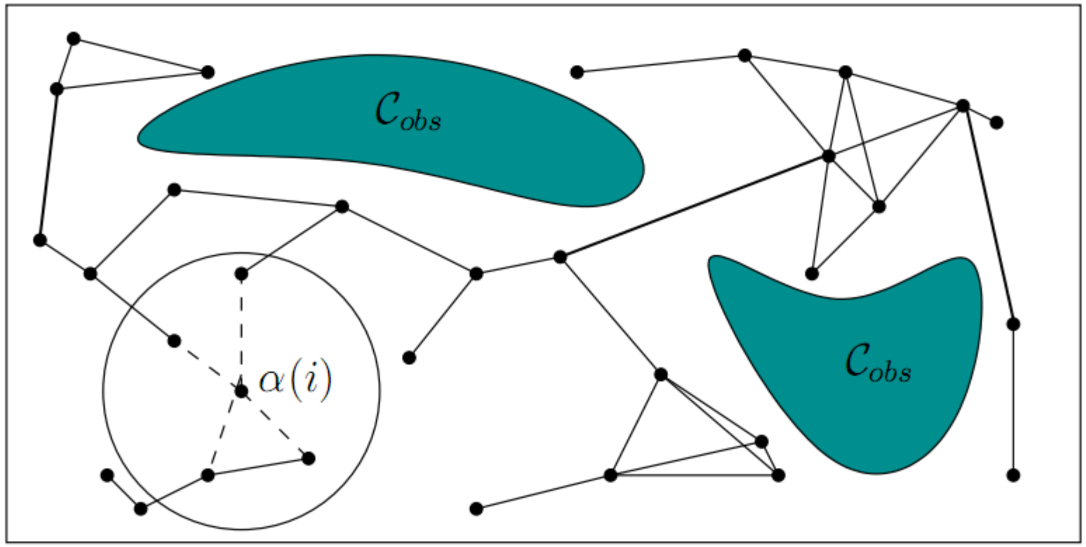
Roadmap Methods

Preprocessing Phase: During the preprocessing phase, substantial effort is invested to build G in a way that is useful for quickly answering future queries.



Query Phase: During the query phase, a pair, q_I and q_G, is given. Each configuration must be connected easily to G using a local planner. Following this, a discrete search is performed using any of the algorithms to obtain a sequence of edges that forms a path from q_I to q_G.

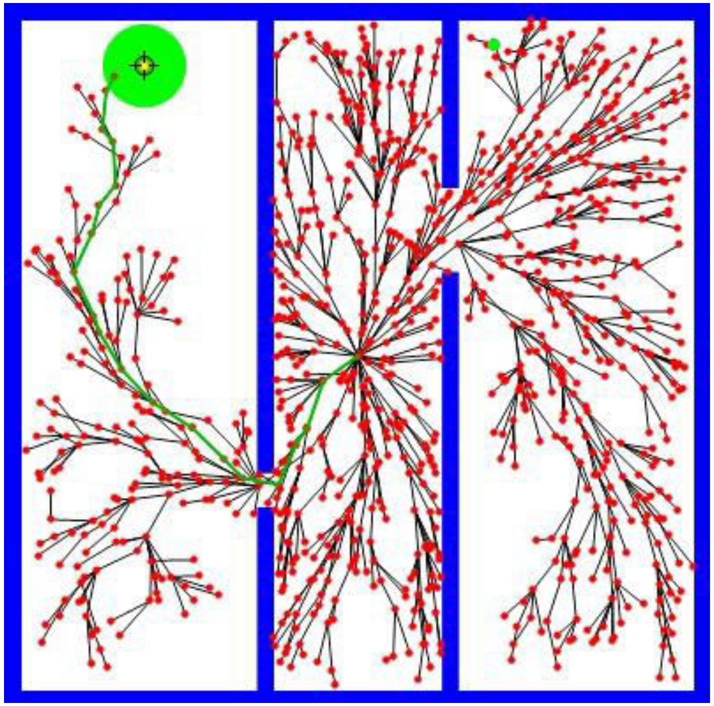
Probabilistic roadmaps



BUILD_ROADMAP

$\mathcal{G}$.init(); $i \leftarrow 0$;
while $i < N$
  if $\alpha(i) \in \mathcal{C}_{free}$ **then**
    $\mathcal{G}$.add_vertex($\alpha(i)$); $i \leftarrow i + 1$;
    for each $q \in$ NEIGHBORHOOD($\alpha(i)$,$\mathcal{G}$)
      if $\mathcal{G}$.vertex_degree(q) $< K$
        $\mathcal{G}$.add_edge($\alpha(i)$, q);

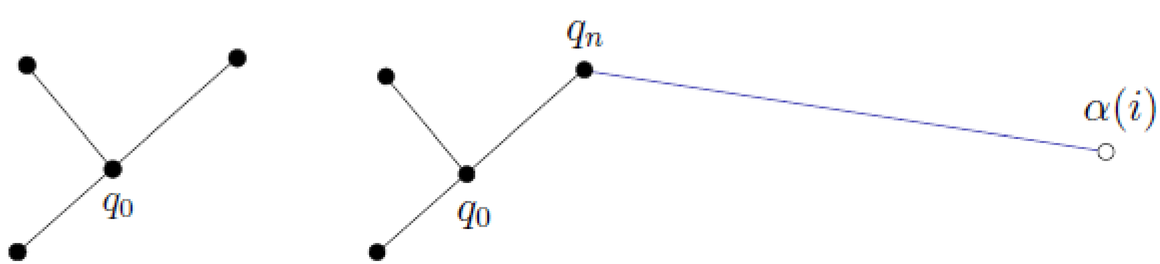
Rapidly Exploring Dense Trees



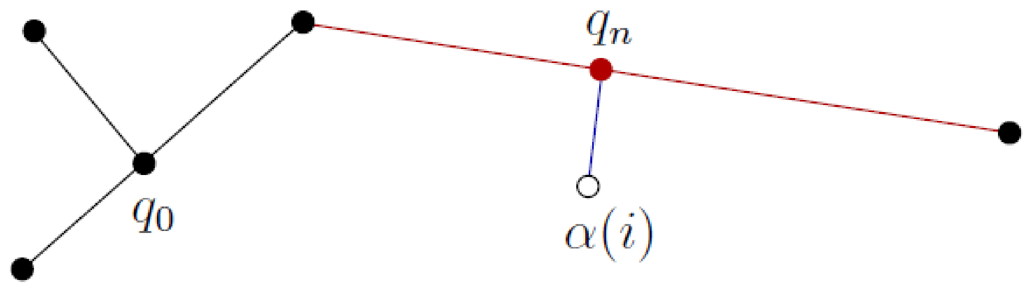
Использование дерева, построенного на основе случайной выборки точек.

Rapidly Exploring Dense Trees

SIMPLE_RDT(q_0)
$\mathcal{G}$.init(q_0);
for $i = 1$ **to** k **do**
    $\mathcal{G}$.add_vertex($\alpha(i)$);
    $q_n \leftarrow$ NEAREST($S(\mathcal{G}), \alpha(i)$);
    $\mathcal{G}$.add_edge($q_n, \alpha(i)$);



Rapidly Exploring Dense Trees

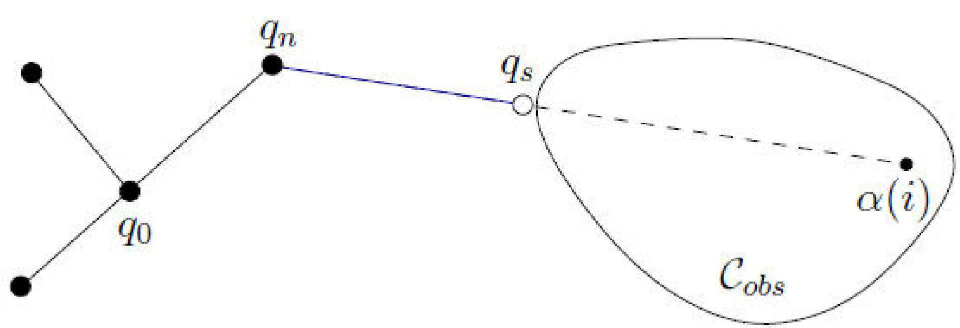


If the nearest point in S lies in an edge, then the edge is split into two, and a new vertex is inserted into graph G.

Rapidly Exploring Dense Trees

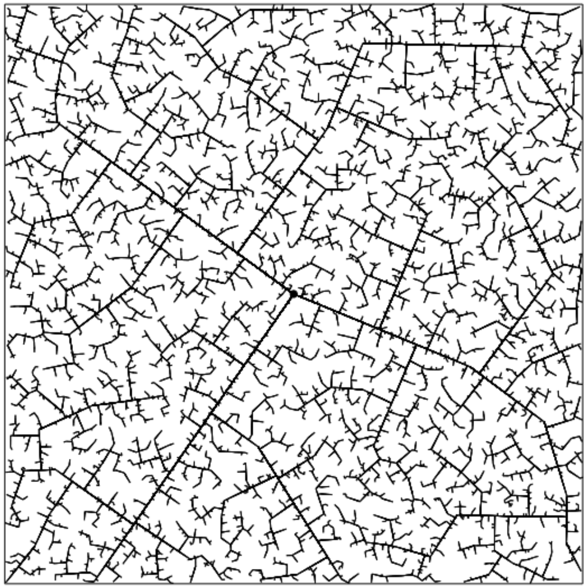
RDT(q_0)

$\mathcal{G}$.init(q_0);
for $i = 1$ **to** k **do**
    $q_n \leftarrow$ NEAREST($S, \alpha(i)$);
    $q_s \leftarrow$ STOPPING-CONFIGURATION(q_n,$\alpha(i)$);
    if $q_s \neq q_n$ **then**
        $\mathcal{G}$.add_vertex(q_s);
        $\mathcal{G}$.add_edge(q_n, q_s);



Rapidly Exploring Dense Trees

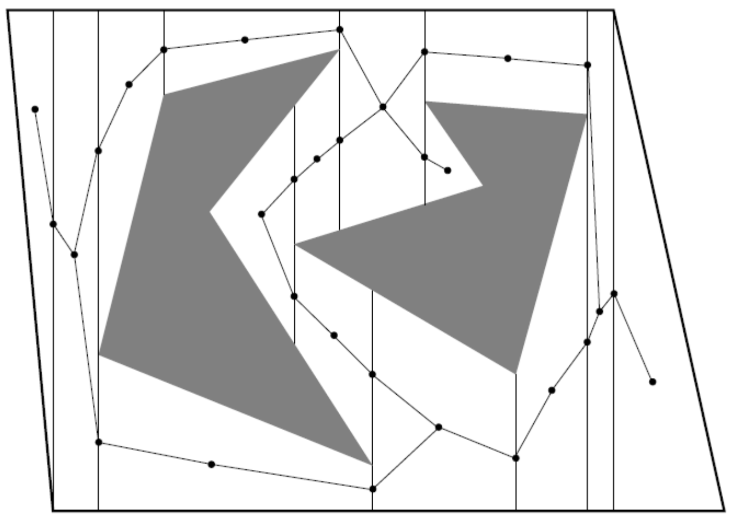
45 iterations

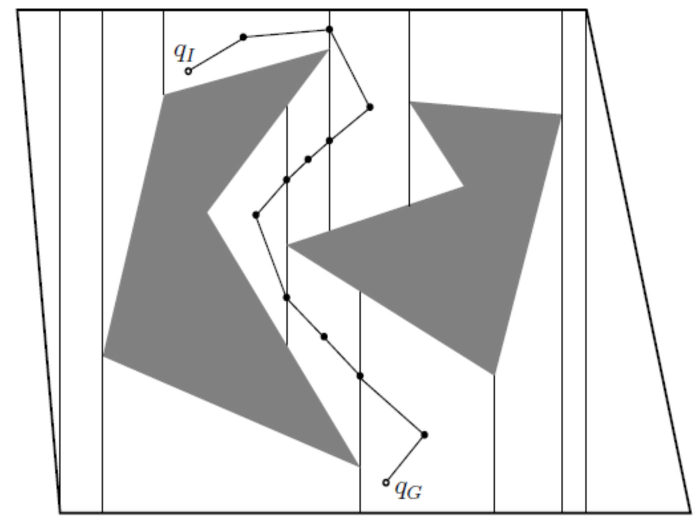


2345 iterations

Planner can be made by directly using the algorithm to grow a tree from q_I and periodically check whether it is possible to connect the RDT to q_G.

Trapezoidal decomposition

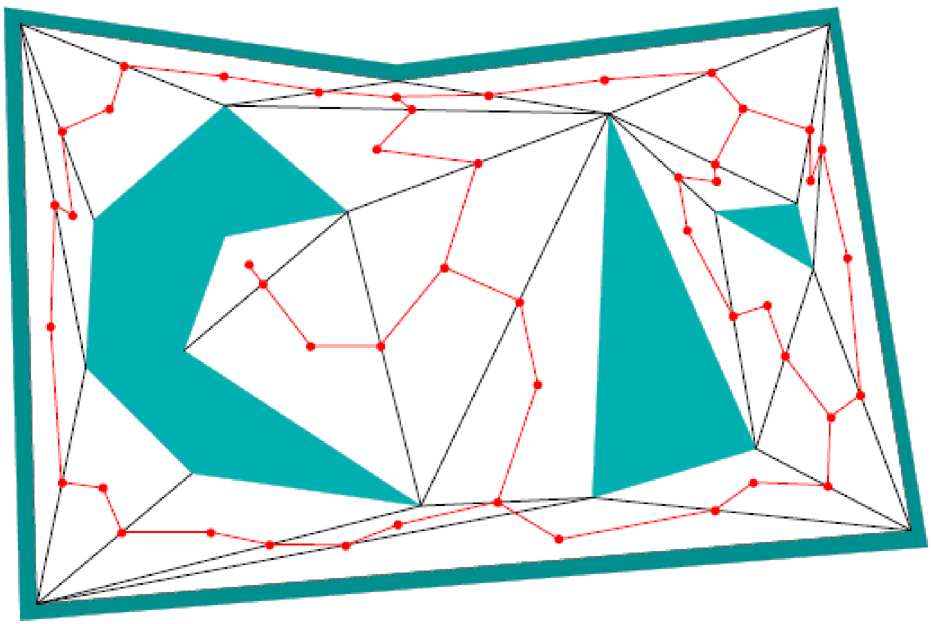




C_{free} делится на многоугольники, имеющие форму трапеций и треугольников.
Центры многоугольников соединяются с серединами их сторон.
Полученный граф используется в качестве **RoadMap**.

Построение: **Метод заметающий прямой** (Plane-sweep principle)

A roadmap obtained from the triangulation.



Center of triangles and midpoints of triangle edges are the nodes of the roadmap.