运动规划

清华大学自动化系

授课教师：李翔

清華大學
Tsinghua University

引言

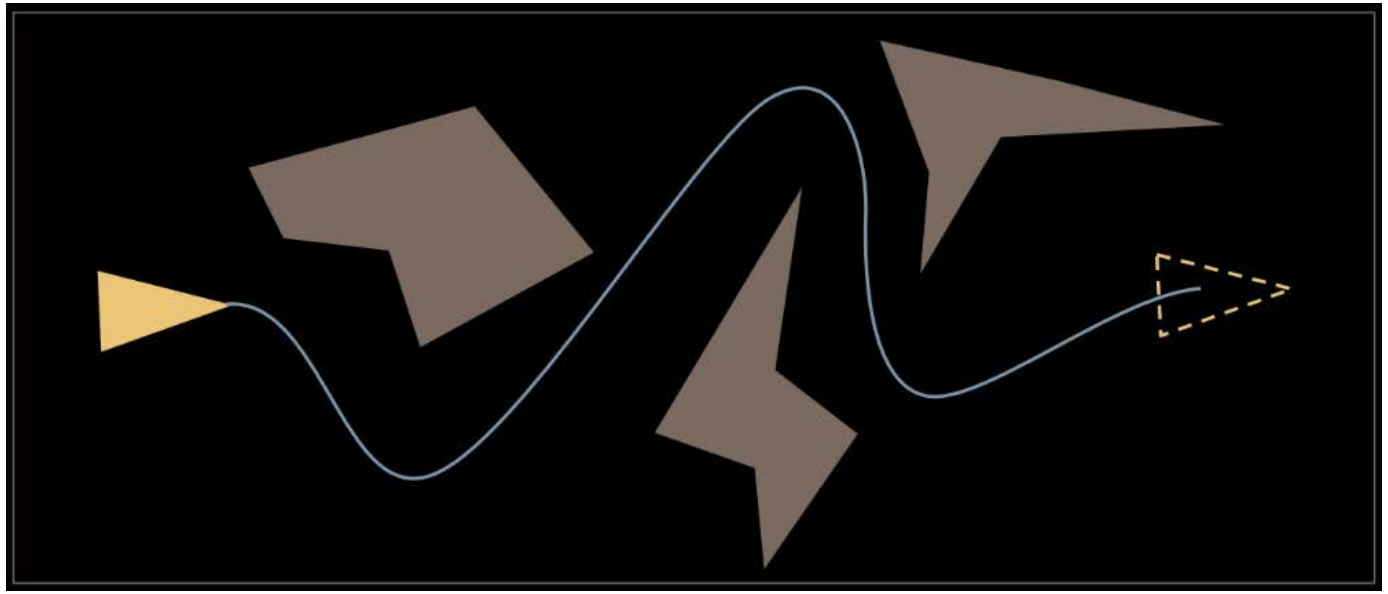
- **运动规划（Motion Planning）**
  - 给定机器人的初始状态 q_s 和目标状态 q_g ，规划出适当的机器人的运动序列，使机器人能够从初始状态 q_s 运动到目标状态q_g。
  - 根据任务不同，运动过程会具有不同的约束，例如无碰撞约束。

无人机在障碍环境中飞行

人形机器人在障碍环境中进行操作

引言

- 运动规划（motion Planning）
  - **路径规划（path planning）**
    - 生成一条从初始状态到目标状态的连续路径，而不考虑和时间的关系及机器人本身的动力学。
    - 路径规划是几何问题（geometric problem）
  - **轨迹规划（trajectory planning）**
    - 每个轨迹点包含时间，需要考虑机器人的动力学约束。



路径规划一般是完整运动规划的基础，受课时限制，本课程主要介绍路径规划。

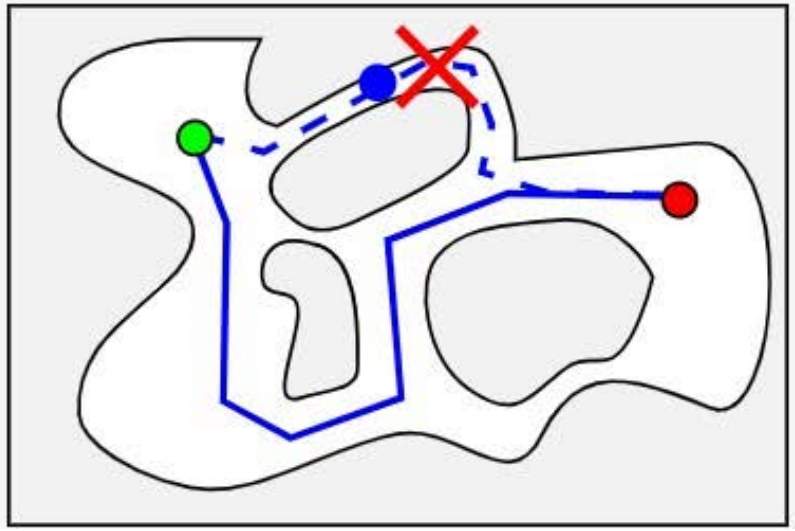
引言

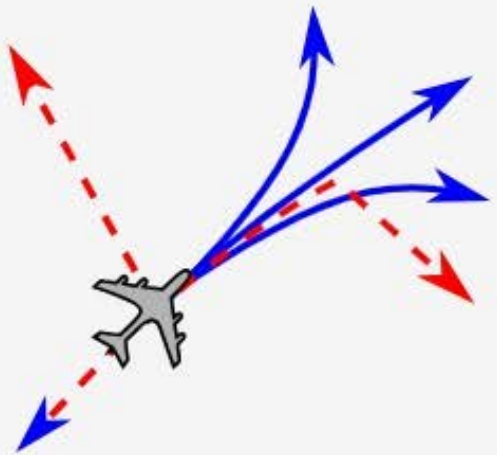
清華大學 Tsinghua University

- **路径规划问题的定义**
  - 定义机器人的**构型空间**（configuration space, C-space）为 $\mathcal{C}$。
  - 一个路径规划问题需要有以下**输入**：
    - 机器人的几何和运动学描述；
    - 环境的几何描述；
    - 机器人运动的约束；
    - 机器人的初始状态；
    - 机器人的目标状态（可以是不同的描述形式）；
    - 对于最优规划问题，还需要有最优化的目标：例如使路径最短、消耗能量最少等。
  - **可行路径**（feasible path）：C-space 中的一条以初始状态为起点、最终到达目标状态的不违反任何约束的连续曲线 $y(s): [0,1] \rightarrow \mathcal{C}$。
  - 如果制定了**最优化**的目标，那么得到的路径应该是最优的或者近似最优的（optimal path or near-optimal path）。

引言

- **约束**：
  - 局部约束：每个路径点自身需要满足的约束，例如无碰撞约束等。
  - 微分约束：对于路径的导数（变化）的约束，例如喷气式飞机的转向约束。
  - 全局约束：对于路径整体的约束，例如无人机需要在电量耗尽前返回基地。
- **Kinodynamic planning**：一个规划问题中既有局部约束，也有微分约束。





引言

清華大學
Tsinghua University

- **规划算法的性能概念**：
  - 完备性（completeness）：完备性指对于可解的问题，算法能保证在有限时间内找到可行解；对于无解的问题，算法能够返回“无解”。
  - 最优性（optimality）：算法是能找到定义指标下的最优路径。
  - 效率（efficiency）：在实际问题中，算法需要多长时间来找到可行/最优路径?
  - 泛化性（generality）：算法能够求解哪类问题？不能求解哪类问题?
- 对于不同的任务，以上指标的重要性和必要性是不同的，需要具体问题具体分析。

机器人构型空间

清華大學 Tsinghua University

- **构型空间**:
  - 定义机器人的**构型空间**（configuration space, C-space）为 $\mathcal{C}$。
  - 机器人的一个状态 q 对应 C-space 中的一个点，即 $q \in \mathcal{C}$。
  - 定义**工作空间** $\mathcal{W}$（workspace）为机器人所在的二维（$\mathbb{R}^2$）或三维空间（$\mathbb{R}^3$）。

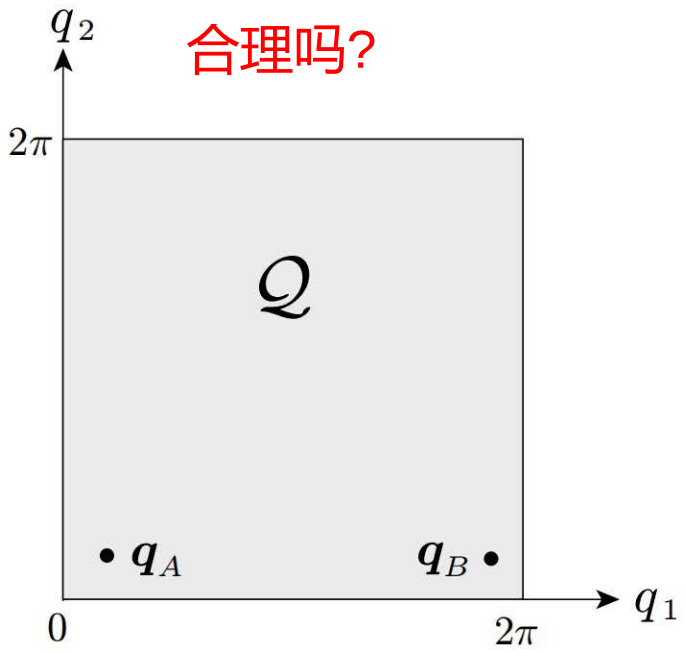
- **例子**:
  - 平面上的质点移动机器人：$\mathcal{C} = \mathbb{R}^2$，即二维坐标。
  - 平面上的多边形移动机器人（考虑朝向）：$\mathcal{C} = SE(2) = \mathbb{R}^2 \times SO(2)$，即二维坐标+一维朝向，共三维。
  - 三维空间中的多面体移动机器人（考虑姿态）：$\mathcal{C} = SE(3) = \mathbb{R}^3 \times SO(3)$，即三维坐标+三维朝向，共六维。
  - 基座固定的具有n个单自由度关节的机械臂：$\mathcal{C}$ 是 $SE(3)$ 的一个子集。因为每个关节只有一个自由度（即有五个约束），所以总的 $\mathcal{C}$ 的维度为 $6n - 5n = n$ 维。

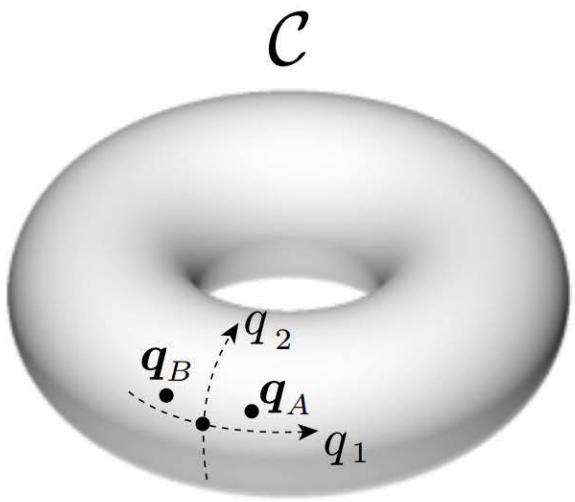
机器人构型空间

- 例子：考虑一个具有两个旋转关节（可无限旋转）的平面机械臂

$$\mathcal{Q} = \{\boldsymbol{q} = (q_1, q_2) : q_1 \in [0, 2\pi), q_2 \in [0, 2\pi)\}$$

$SO(2) \times SO(2)$





平面二旋转关节机械臂的 C-space

- 若是关节最大旋转角度范围只有2π的机械臂呢?

机器人构型空间

- **障碍物**：
  - 定义在工作空间$\mathcal{W}$中。假设有p个障碍物，记它们为$\mathcal{O}_1, \ldots, \mathcal{O}_p$。记机器人在状态$q$时对应的在$\mathcal{W}$中的几何形状为$\mathcal{B}(q)$。
  - 在$\mathcal{W}$中定义的障碍物$\mathcal{O}_i$在构型空间$\mathcal{C}$中的映像被称为$\mathcal{C}-obstacle$，数学描述为：

$$\mathcal{CO}_i = \{\boldsymbol{q} \in \mathcal{C} : \mathcal{B}(\boldsymbol{q}) \cap \mathcal{O}_i \neq \emptyset\}$$

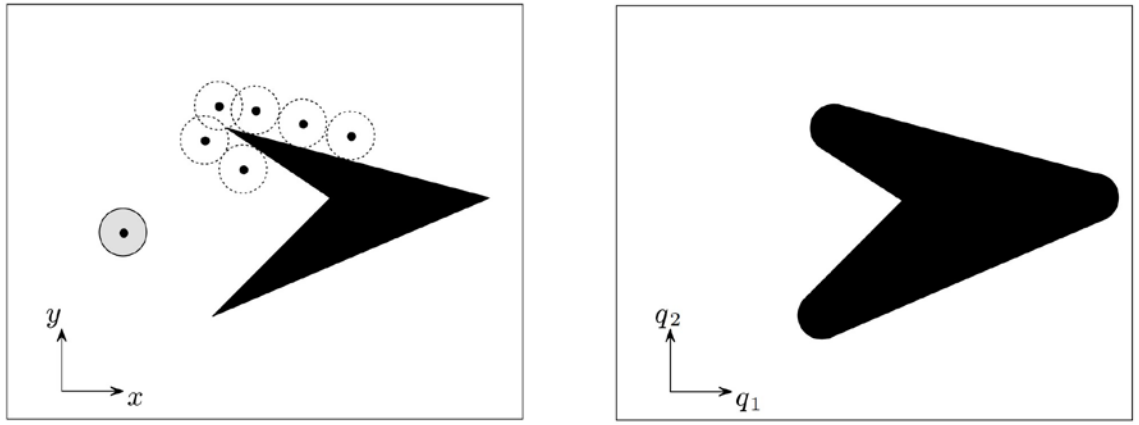
  - 考虑所有障碍物，则$\mathcal{C}$-obstacles空间被定义为：

$$\mathcal{CO} = \bigcup_{i=1}^{p} \mathcal{CO}_i$$

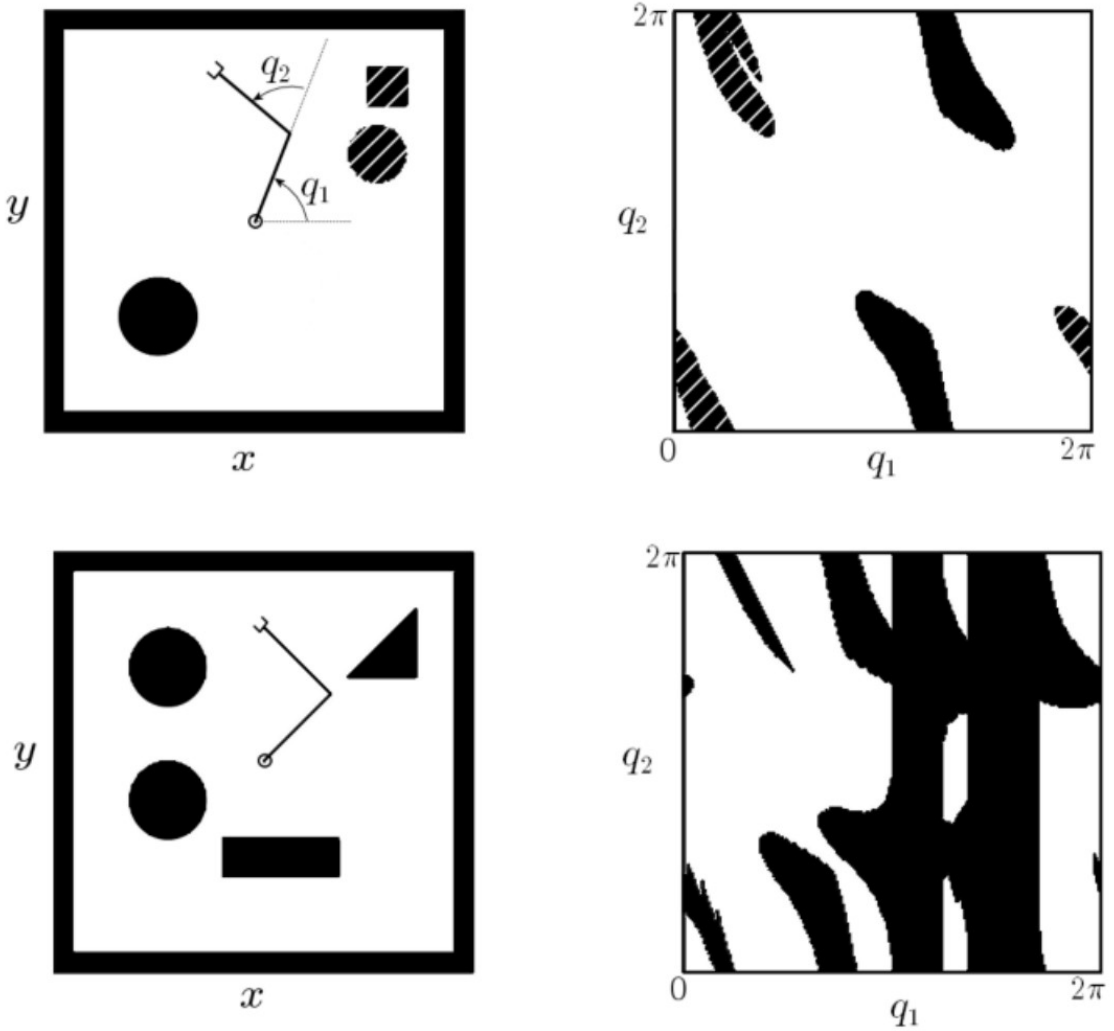
  - 其补集被称为自由构型空间$\mathcal{C}_{\text{free}}$。

机器人构型空间

- **障碍物**:



一个平面圆形移动机器人的$\mathcal{C}$-obstacles

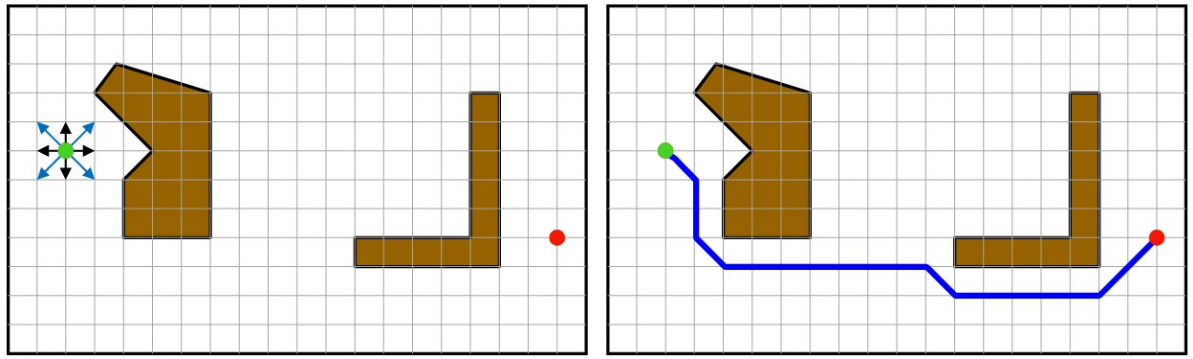


一个平面二自由度机械臂的$\mathcal{C}$-obstacles

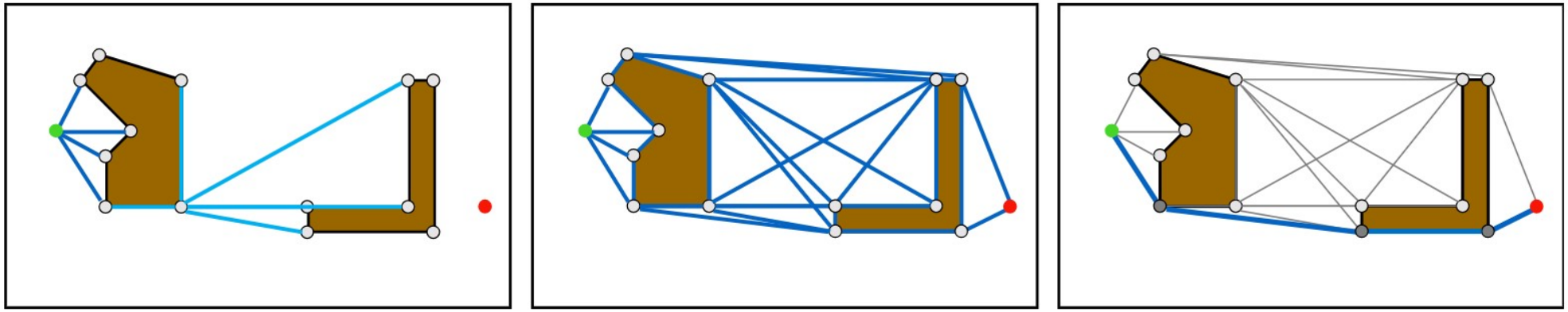
复杂

构型空间中的路径规划算法

- **基于搜索的算法**
  - 基本思想：将 $\mathcal{C}_{\text{free}}$ 离散化为一个图（graph），之后在图上搜索得到路径。
  - **建图方法**：
    - 网格搜索（grid search）
      - 分辨率影响较大。
      - 分辨率完备（resolution-complete）
    - 可视图（visibility graph）
      - 若$\mathcal{CO}_i$均为多边形，则是完备且最优的。



网格搜索



可视图

构型空间中的路径规划算法

- **基于搜索的算法**
  - **图搜索方法**:
    - Dijkstra算法:
      - 可以得到初始节点到图中任意节点的最短路径。
        - Maintain a container to store all the nodes to be visited
        - The container is initialized with the start state X_S
        - Loop
          - Remove a node from the container according to some pre-defined score function
            - Visit a node
          - Expansion: Obtain all neighbors of the node
            - Discover all its neighbors
          - Push them (neighbors) into the container
        - End Loop

$v_i.g$: accumulated cost
$v_m.prior$: 前路径点

Algorithm 1 Dijkstra 算法

```
1:  Initialize priority queue Q ← {v_s}; v_s.g ← 0; v_i.g ← ∞, v_i ∈ V \ v_s;
2:  while Q ≠ ∅ do
3:      v_n ← Q.pop(); // v_n = arg min_{v_i} v_i.g (v_i ∈ Q)
4:      v_n.expanded = true;
5:      if v_n = v_g then
6:          return true;
7:      end if
8:      for v_m ∈ Neighbor(v_n) do
9:          if v_m.expanded = false and v_m.g > v_n.g + c_nm then
10:             if v_m.g = ∞ then
11:                 Q.push(v_m);
12:             end if
13:             v_m.g ← v_n.g + c_nm;
14:             v_m.prior ← v_n;
15:         end if
16:     end for
17: end while
18: return false;
```

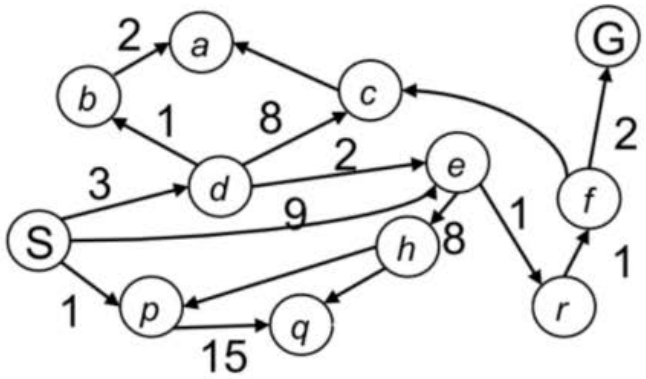
- 完备且最优。
- 没有使用目标的信息。

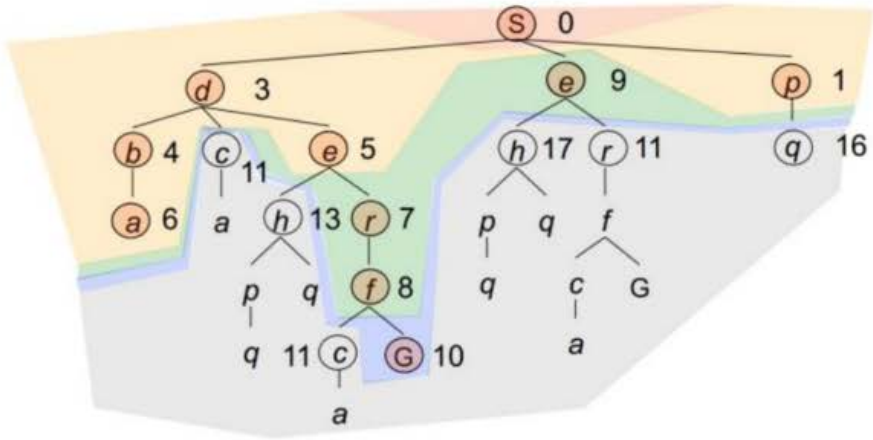
构型空间中的路径规划算法

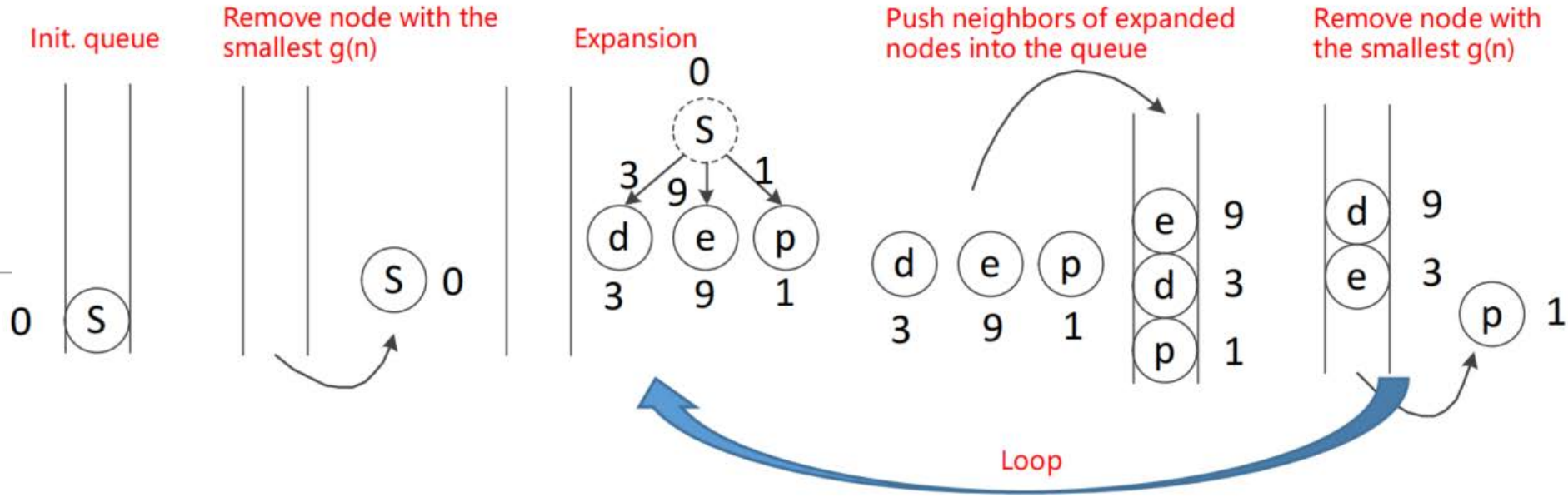
- **基于搜索的算法**
  - **图搜索方法**:
    - Dijkstra算法:

Algorithm 1 Dijkstra 算法

```
1: Initialize priority queue Q ← {v_s}; v_s.g ← 0; v_i.g ← ∞, v_i ∈ V \ v_s;
2: while Q ≠ ∅ do
3:     v_n ← Q.pop(); // v_n = arg min_{v_i} v_i.g (v_i ∈ Q)
4:     v_n.expanded = true;
5:     if v_n = v_g then
6:         return true;
7:     end if
8:     for v_m ∈ Neighbor(v_n) do
9:         if v_m.expanded = false and v_m.g > v_n.g + c_nm then
10:            if v_m.g = ∞ then
11:                Q.push(v_m);
12:            end if
13:            v_m.g ← v_n.g + c_nm;
14:            v_m.prior ← v_n;
15:        end if
16:    end for
17: end while
18: return false;
```







构型空间中的路径规划算法

清華大學 Tsinghua University

- **基于搜索的算法**
  - **图搜索方法**:
    - A*算法:
      - Dijkstra算法在探索过程中不利用目标点的信息，A*算法使用启发式的策略来更好地利用目标信息来引导搜索。
      - 对于每个节点v_i，除了从v_s到v_i的累积代价$v_i.g$外，还维护一个从v_s途径v_i到达v_g的最小估计代价 $v_i.f = v_i.g + h(v_i)$，其中$h(v_i)$指从v_i到v_g的**估计**最小代价。
      - A*算法在搜索时以$v_i.f$为指标，而Dijkstra算法中是以$v_i.g$为指标。

构型空间中的路径规划算法

- **基于搜索的算法**
  - **图搜索方法**:
    - A*算法:

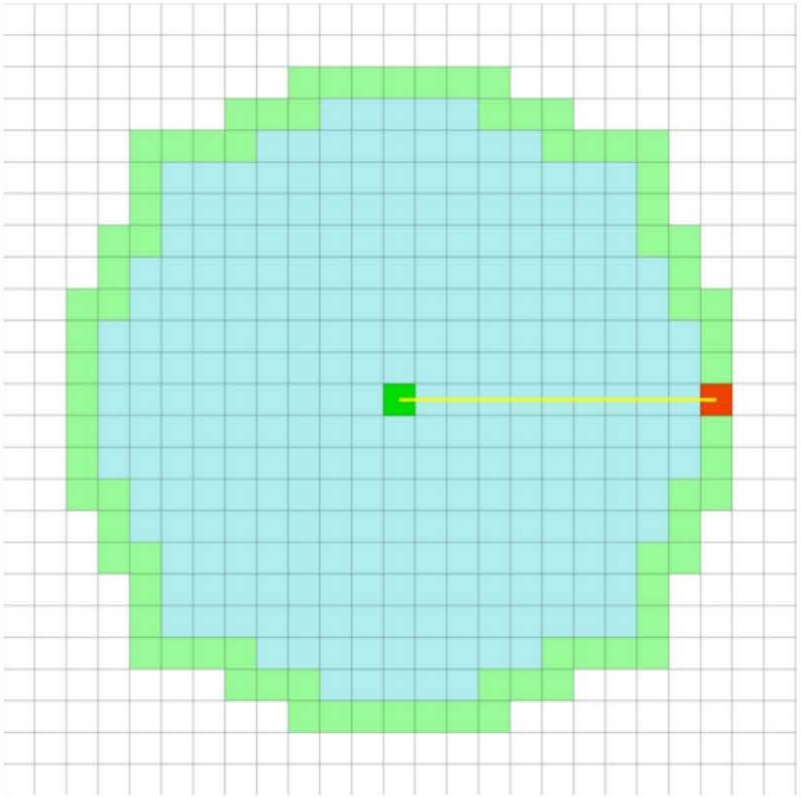
Algorithm 2 A* 算法

Initialize priority queue $\mathcal{Q} \leftarrow \{v_s\}$; $v_s.g \leftarrow 0$; $v_i.g \leftarrow \infty, v_i \in \mathcal{V} \setminus v_s$; Define $h(v_i)$;
while $\mathcal{Q} \neq \emptyset$ **do**
    $v_n \leftarrow \mathcal{Q}$.pop(); // $v_n = \arg\min_{v_i} v_i.f \ (v_i \in \mathcal{Q})$
    v_n.expanded = true;
    if $v_n = v_g$ **then**
        return true;
    end if
    for $v_m \in$ Neighbor(v_n) **do**
        if v_m.expanded = false **and** $v_m.g > v_n.g + c_{nm}$ **then**
            if $v_m.g = \infty$ **then**
                $\mathcal{Q}$.push(v_m);
            end if
            $v_m.g \leftarrow v_n.g + c_{nm}$;
            v_m.prior $\leftarrow v_n$;
        end if
    end for
end while
return false;

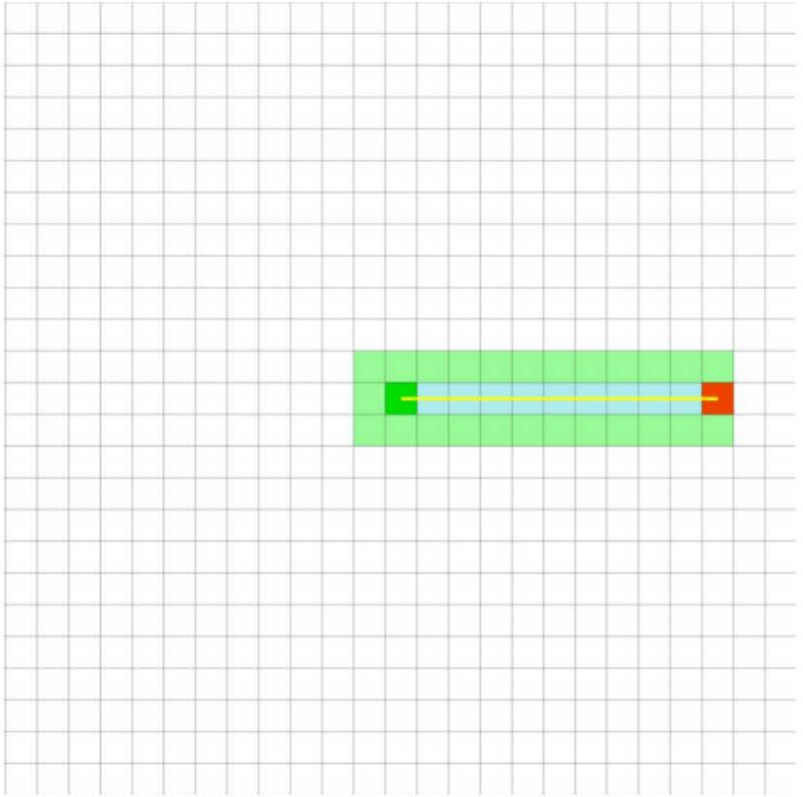
构型空间中的路径规划算法

- **基于搜索的算法**
  - Dijkstra算法 v.s. A*算法

http://qiao.github.io/PathFinding.js/visual/



Dijkstra



A*

- **基于搜索的算法**
  - **小结：**
    - 需要 $\mathcal{C}_{\text{free}}$ 能被显式地表达出来。
    - 适合于低维空间，例如二维或三维空间。算法复杂度随空间维度指数增长。
    - 在移动机器人上应用较多，较少应用于高自由度机械臂。

构型空间中的路径规划算法

清華大學
Tsinghua University

- **基于采样的算法**
  - 不需要显式的$\mathcal{CO}$表达。
  - 只需可行性查询（feasibility queries）：
    - 一个机器人状态 q 是否是可行的（feasible），即 q 是否属于$\mathcal{CO}$。
    - 两个状态之间的运动是否是可行的（visible）（一般是两个状态之间的直线运动）。
  - 可以处理高维空间的规划问题，例如机械臂规划。
  - 最常见的两类算法：
    - 概率路标图（probabilistic roadmaps, PRMs）
    - 快速探索随机树（rapidly-exploring random trees, RRTs）

构型空间中的路径规划算法

清華大學 Tsinghua University

- **基于采样的算法**
  - **Probabilistic roadmaps （PRM）**
    - 基本思想：
      - 使用随机采样的方式，建立 $\mathcal{C}_{\text{free}}$ 的近似路标图，之后再使用图搜索方法得到可行路径。
    - 流程：
      - 在$\mathcal{C}$中随机采样 N 个点；
      - 对采样得到的点进行 feasibility 检测；用其中 feasible 的点以及初始状态和目标状态建立一个路标图（roadmap），这些点即为 roadmap 的顶点（又称 milestones）；
      - 对每个顶点，选择它与它附近的顶点进行 visibility 检测；将其中 visible 的边添加进图中。
      - 在图上使用图搜索方法（例如 Dijkstra 或 A* 算法）得到可行路径。

构型空间中的路径规划算法

- **基于采样的算法**
  - **PRM**

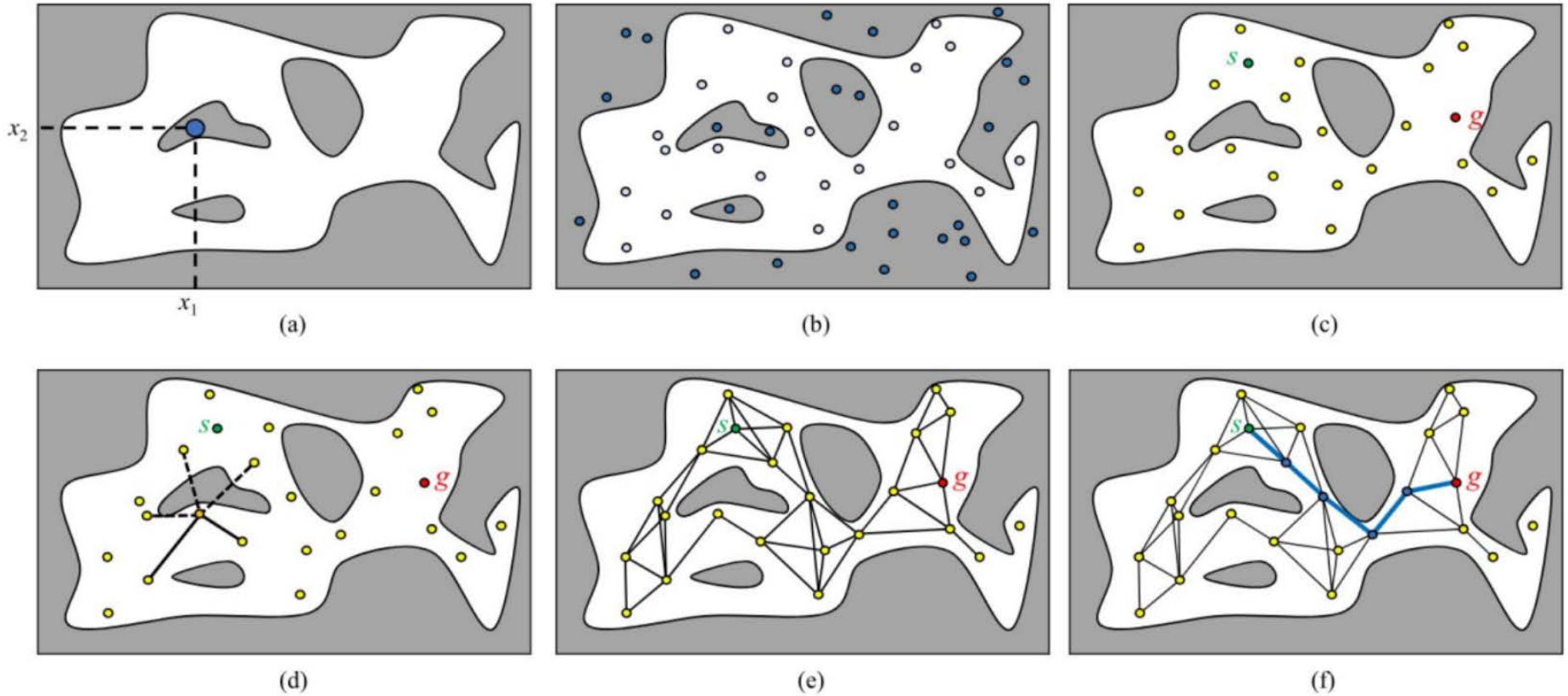
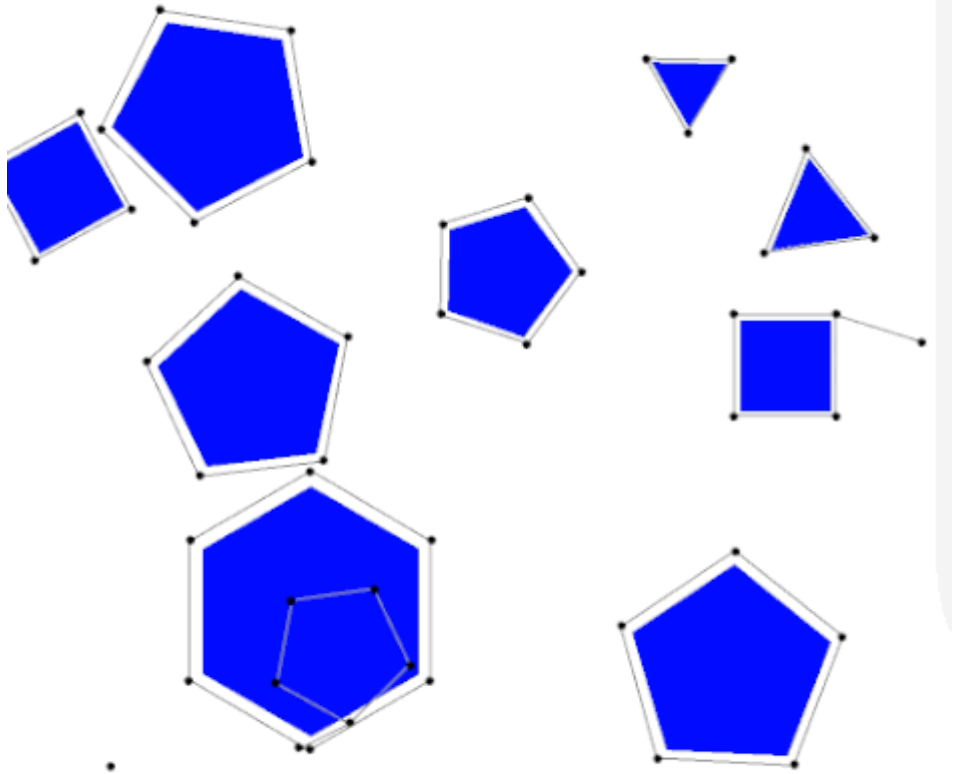


图 10: PRM 算法流程示意图 [4]。(a) 随机采样；(b)feasiblity 检测；(c) 保留 feasible 的点，添加初始和目标点；(d)visibility 检测；(d) 形成最终的路标图；(e) 进行图搜索。

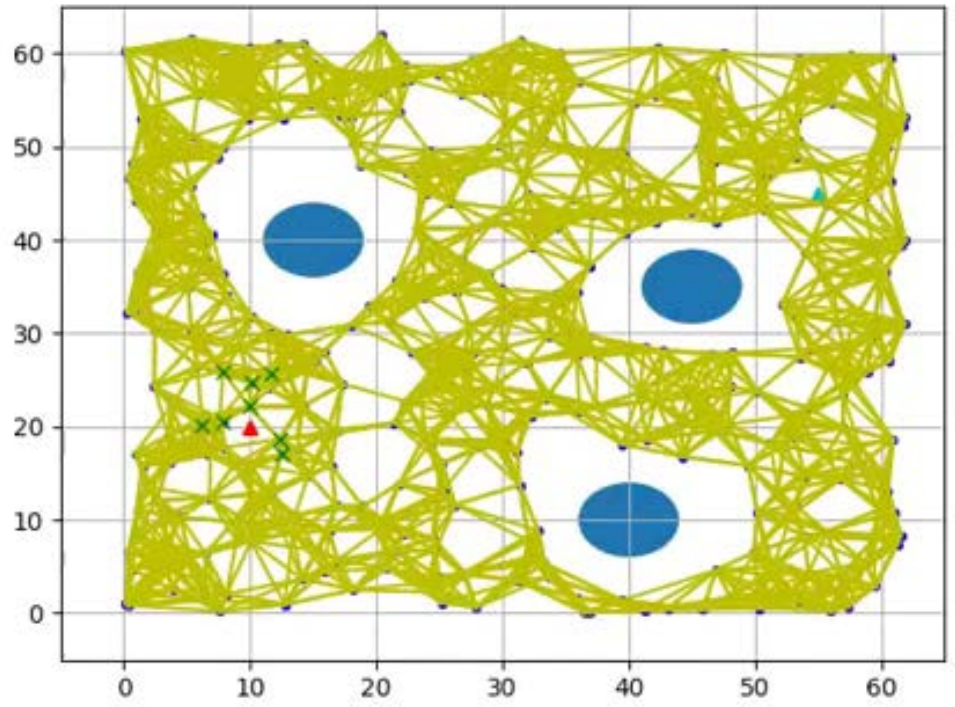
构型空间中的路径规划算法

- **基于采样的算法**
  - **PRM**

建图



图搜索



构型空间中的路径规划算法

清華大學
Tsinghua University

- **基于采样的算法**
  - **PRM**
    - PRM中**建图**的过程是主要的时间开销来源，当建好图后，可以很高效地得到图中**任意**两个节点之间的可行路径，所以其属于**multi-query**方法。
    - PRM是概率性算法，是**概率完备**的（probabilistically complete）：如果问题有解，则随着采样点的增加，算法找到一个可行路径的概率逐渐趋近于 1。
    - **影响算法性能的因素**:
      - 如何进行随机采样?
        - 可修改 PRM 中的采样分布以提升规划效率。
      - 如何找到一个路标点（顶点）附近的路标点?
        - 最近邻搜索问题（k-d tree …）。

构型空间中的路径规划算法

清華大學
Tsinghua University

- **基于采样的算法**
  - **Rapidly-exploring random trees（RRT）**
    - single-query，即只需要得到初始状态到目标状态的路径。
    - 基本思想：从初始状态开始 q_s 扩展一棵树 $\mathcal{T}$ ，逐渐（随机）探索整个空间，直到扩展到达目标节点 q_g 。

- **基于采样的算法**
  - **Rapidly-exploring random trees（RRT）**
    - Goal bias：以一定概率 p_{goalbias} 选择 q_g 替代 q_{rand} 来引导树扩展。根据经验，一般 p_{goalbias} 选择 0.05 至 0.1 是比较合适的。
    - 与 PRM 算法不同，RRT 算法并不会将随机采样得到的状态用在路径中，而只是用随机采样的状态来引导树扩展的方向。
    - RRT 算法也是概率完备的，而且其算法实现简便，非常易于添加其它约束，所以应用十分广泛。
    - 缺点：
      - 在复杂场景中效率不够高。
      - 得到的路径常常十分曲折（非最优）。

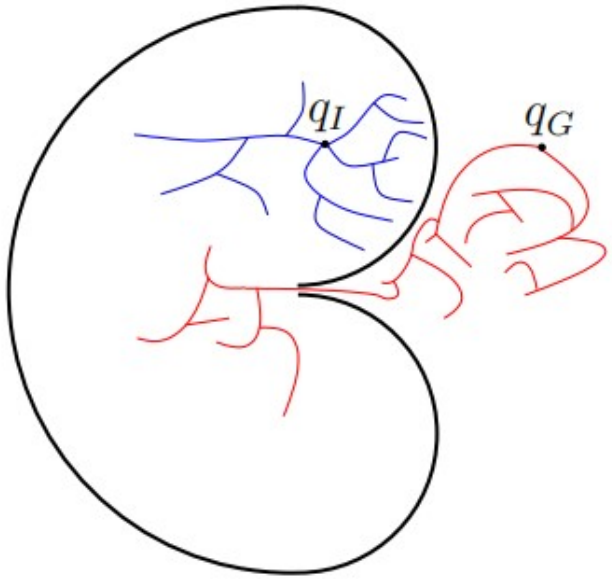
构型空间中的路径规划算法

- **基于采样的算法**
  - **Rapidly-exploring random trees（RRT）**

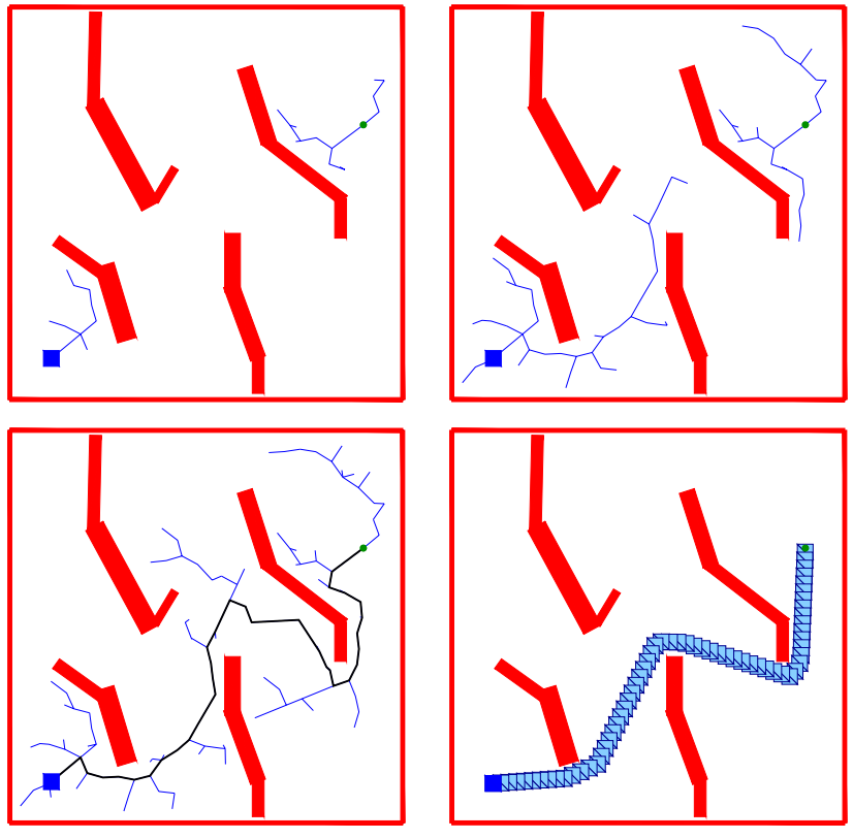
RRT过程示意视频 (youtube)

- **基于采样的算法**
  - **Bidirectional RRT （双向RRT）**
    - 目的：提高规划效率。
    - 基本思想：其从 q_s 和 q_g 同时长出两棵树，相向探索，从而提高扩展效率。

      当两棵树连起来时，即为找到可行路径。



bug trap



- **基于采样的算法**
  - **小结：**
    - 既可以用于低维空间，也可以用于高维空间。应用广泛。
    - 可以非常方便地加入其它局部约束，例如实现末端位姿约束的机械臂规划。
    - 还可以加入微分约束来实现 kinodynamic planning。
    - 实现简便，但其性能往往受到很多细节的影响，实际应用时往往需要调整超参数来达到最优的性能。
    - 主要用于找到可行路径，而非严格最优路径。

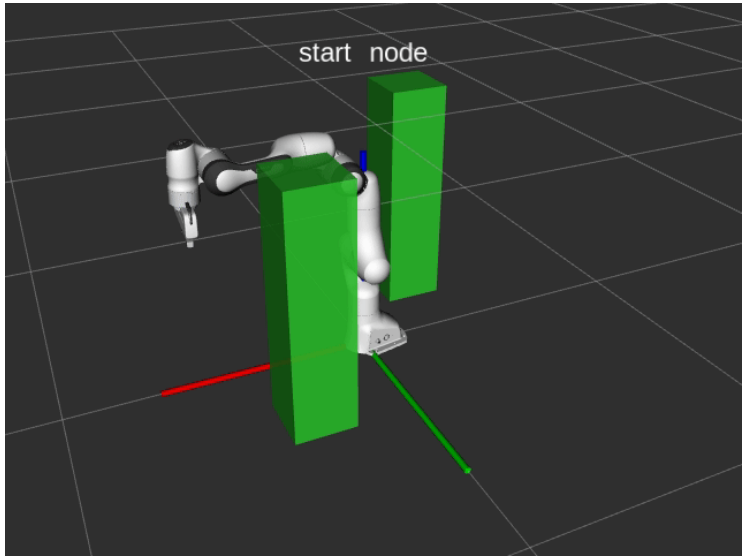
对于多自由度机械臂的路径规划

- **机械臂路径规划（path planning for manipulators）的特点**
  - 维度高。
  - free configuration space （C_{free}） 难以描述。
  - 考虑 configuration space (joint space) 与 task space (cartesian space)。

- **常用的方法：**
  - 复杂任务/障碍物：sampling-based planning.
  - 简单任务/障碍物：potential field methods / optimization-based planning.

对于多自由度机械臂的路径规划

- **关节空间目标的路径规划**
  - 机械臂路径规划的目标是定义在关节空间，即指定了 q_g ，则问题和上一节中讨论的构型空间中的路径规划是一致的。
  - 常用的方法是 PRM、RRT 及其多种变种。一般来说，Bidirectional RRT 的时间开销会比普通 RRT 要小很多。
  - 适当的距离度量：对不同关节加权。
  - 机械臂几何模型复杂，碰撞检测较为复杂，碰撞检测的时间开销在规划的总开销中的占比是很大的。常用 FCL（Flexible Collision Library）库。



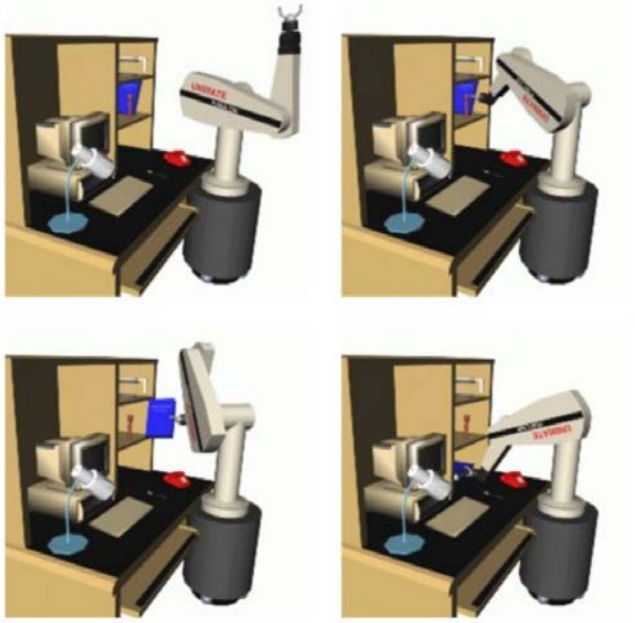
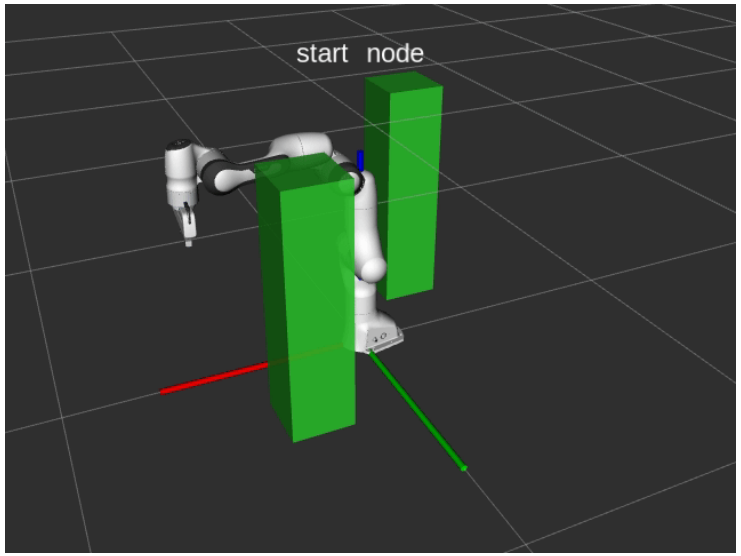


图 16: 基于 Bidirectional RRT 的机械臂路径规划及操作 [9]。

对于多自由度机械臂的路径规划

- **任务空间目标的路径规划**
  - 只指定了机械臂末端的目标位姿。
    - 最简单的方法：先对 task-space goal 求 IK，得到 joint-space goal，再使用前文方法。
      - Problem：IK多解，选择哪一个解？不合适的解可能是不可达的。



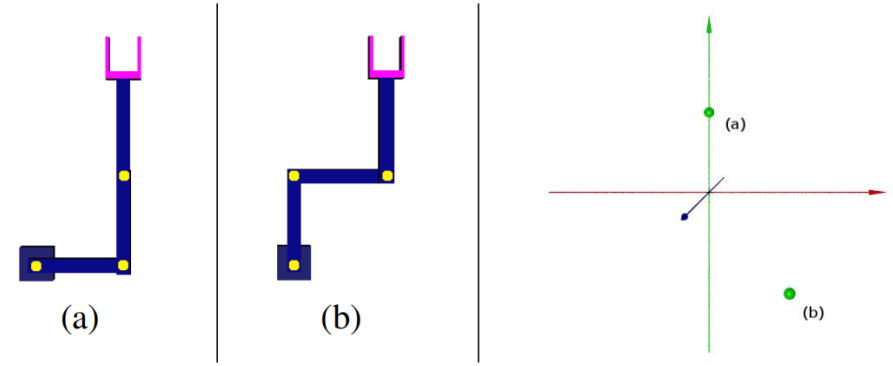


Fig. 2. Two inverse kinematics solutions for the same end-effector posture.

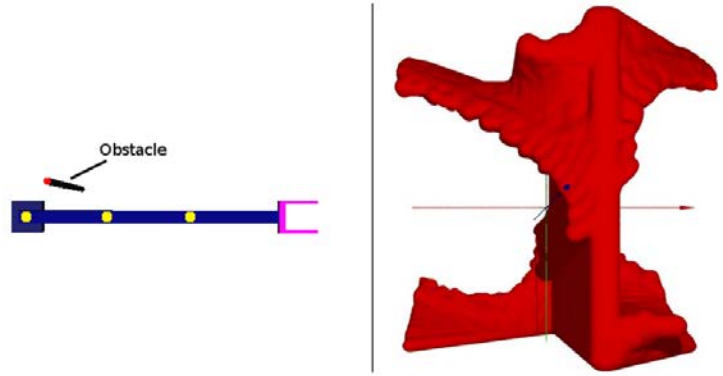


Fig. 3. 3-DOF robot and configuration space with a C-obstacle creating two disconnected components of free space.

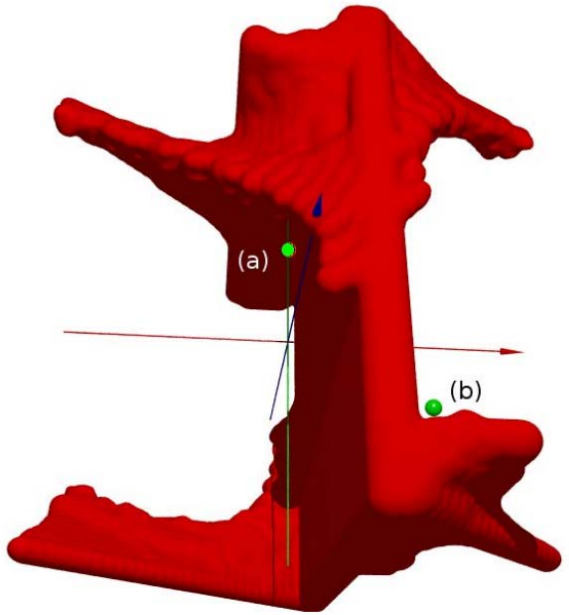


Fig. 4. Two IK solutions (a) and (b) lying in disconnected components of the configuration space.

对于多自由度机械臂的路径规划

- **任务空间目标的路径规划**
  - **第一类方法**：
    - 无需显式 IK，使用 task-space goal 引导 RRT 探索。
  - 基本思想：
    - 在每次 RRT 迭代中，以一定概率按RRT方式扩展（no goal bias）。
    - 以一定概率进行 ExtendTowardsGoal()：
      - 选择离 goal pose 最近的节点，基于 Jacobian 矩阵的转置，向 goal pose 扩展（optimal local controller）。
  - 优劣：
    - 不需要显式的逆运动学求解。

```
GrowRRT()
2  Q_new = {q_start};
3  while (DistanceToGoal(Q_new) > distanceThreshold)
4    p = RandomReal([0.0, 1.0]);
5    if (p < p_g)
6      Q_new = ExtendTowardsGoal();
7    else
8      Q_new = ExtendRandomly();
9    if (Q_new ≠ ∅)
10     AddNodes(Q_new)
ExtendTowardsGoal()
11   q_old =ClosestNodeToGoal();
12   repeat
13     J^T =JacobianTranspose(q_old);
14     δ_x =WorkspaceDelta(q_old, x_goal);
15     δ_q = J^T · δ_x;
16     q_new = q_old + δ_q;
17     if (CollisionFree(q_old, q_new))
18       Q_new = Q_new ∪ q_new;
19     else
20       return Q_new;
21     q_old = q_new;
22   while (DistanceToGoal(q_new) > distanceTrheshold)
23   return Q_new;
```

对于多自由度机械臂的路径规划

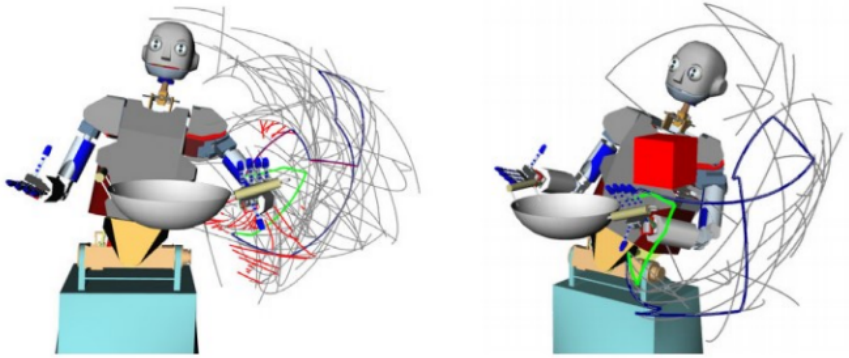
- **任务空间目标的路径规划**
  - **第二类方法**：
    - 在 RRT 过程中不断采样 task-space goal 对应的 IK 解。
  - 基本思想：
    - 每次迭代中，以一定概率采样 goal pose 及其对应的 IK，添加进 goal tree 中。
  - 优劣：
    - 可以使用 bidirectional RRT，提高效率。
    - 需要非常高效的 IK 求解器。

Algorithm 3: IK-RRT(q_{start},p_{obj},gc)

```
1  RRT1.AddConfiguration(q_start);
2  RRT2.Clear();
3  while (!TimeOut()) do
4      if (#IKSolutions == 0 || rand() < p_IK) then
5          grasp ← GetRandomGrasp(gc);
6          p_target ← ComputeTargetPose(p_obj, grasp);
7          q_IK ← ComputeIK(p_target);
8          if (!Collision(q_IK)) then
9              RRT2.AddConfiguration(q_IK);
10     else
11         q_r ← GetRandomConfiguration();
12         if
           (RRT1.Connect(q_r) & RRT2.Connect.(q_r))
           then
13             Solution ← BuildSolutionPath(q_r);
14             return PrunePath(Solution);
15         end
16     end
17 end
```

对于多自由度机械臂的路径规划

- **任务空间目标的路径规划**
  - **两类方法对比**：
    - 第一类方法更适合简单的环境或机械臂逆运动学求解困难的情况，因为其不需要显式的逆运动学求解，而且其扩展方式是局部最优的。
    - 第二类方法更适合于环境复杂、但可以高效机械臂逆运动学求解的情况，因为其可以使用 Bidirectional RRT，从而极大提高复杂环境中的规划效率。



(a)

	Without Obstacle Avg Runtime	With Obstacle Avg Runtime
J^{+}-RRT	2 032 ms	18 390 ms
IK-RRT	140 ms	480 ms

(b)

图 18: 两类任务空间目标的路径规划方法的对比 [14]。其中 $J^{\dagger}$-RRT 属于第一类不需要显式逆运动学求解的方法，IKRRT 属于第二类随机采样逆运动学解的方法。(a) 对于同一任务的 RRT 树扩展过程对比；(b) 规划时间开销对比。

- **本章没有介绍的内容：**
  - 涉及微分约束的路径（轨迹）规划，即 kinodynamic planning。例如对于固定翼飞机、阿克曼转向的车辆（如小汽车）等的规划。
  - 涉及动态障碍物的路径（轨迹）规划。
  - 最优规划。

- **相关课程资源**：
  - **书籍：**
    - Siciliano, “Robotics Modeling, Planning and Control”, 2009. 该书是机器人领域的经典教材，包括机器人从建模、规划、控制全方位的基础知识。其中第 12 章为 Motion Planning，篇幅较短，可用于初步了解机器人运动规划的概念和方法。
    - Lavalle, “ Planning Algorithms”, 2006. 该书是 Planning 领域最全面的教材，作者Lavalle 是 RRT 及其多种变种的提出者。该书理论扎实、内容全面，适合用于极为深入地学习运动规划理论；但本书成书较早，无法包含近期的研究成果。该书可在线获取：http://lavalle.pl/planning/。
  - **课程：**
    - ”Robotic Systems”, UIUC, Keris Hauser, https://motion.cs.illinois.edu/RoboticSystems/. 其中第三章为 Motion Planning。该部分提供了详细的讲义，架构清晰，内容适中，易于阅读（非常推荐）。
    - ”Motion Planning”, University of Michigan, Dmitry Berenson, https://berenson.robotics.umich.edu/courses/winter2022motionplanning/index.html. 该课程提供 slides，内容全面，链接了很多其它相关学习资料，且涉及一些较新的研究成果。