QD-RRT: 基于 Q 距离函数和快速扩展随机树的机械臂运动规划方法

吴昊, 丁焱*, 朱向阳

上海交通大学机械与动力工程学院, 上海 200240

* E-mail: y.ding@sjtu.edu.cn

收稿日期: 2020-12-23; 接受日期: 2021-04-15; 网络版发表日期: 2022-04-25

国家自然科学基金(批准号: 51822506, 51935010)资助项目

摘要 在传统运动规划算法快速扩展随机树(rapidly-exploring random trees, RRT)的基础上, 引入了目标偏置策略和一种基于 Q 距离(Q -distance, QD)函数的避障方法. 在随机树生成过程中, 首先通过目标偏置策略引导随机树以一定概率朝目标点生长. 若机械臂在新生成的路径节点所表示的位形处与环境障碍物发生碰撞, 则使用 Q 距离函数快速高效地计算二者的嵌入距离, 并利用 Q 距离函数的可微性, 计算碰撞点的 Q 距离函数关于机械臂各个关节角度的梯度, 对原有的路径点进行修正. 这减少了RRT算法在路径扩展过程中的盲目性、随机性. 使用MATLAB与CoppeliaSim机器人仿真软件对该算法进行仿真实验验证, QD-RRT算法与传统RRT算法相比, 收敛速度更快, 生成的路径质量更优. 当环境空间障碍物较复杂或路径需要穿过狭长通道时, 该算法优势更加明显. 同时, 该算法也可应用于RRT的某些改进算法中, 例如RRT*. 本文也将RRT*与QD-RRT*做出了比较.

关键词 机械臂, 路径规划, Q 距离函数, RRT算法

1 引言

运动规划问题一直是机器人领域的研究热点, 经典的运动规划问题是考虑如何将一架钢琴从一个房间搬运到另外一个房间且不与房间中其他物体发生碰撞, 即钢琴搬运工问题^[1]. 该问题揭示了机器人运动规划问题的本质——为机器人寻找一条无碰撞路径使其从起始点运动到终止点^[2]. 在实际的应用中运动规划算法有路径图法^[3,4]、势场法^[5]、胞分解法^[6-8]等, 这些算法在参数设置的比较好时, 可以保证运动规划的完

备性, 即如果存在可达路径, 在有限时间内, 一定可以找到该路径. 然而这些算法在解决高维空间运动规划问题上存在诸多缺陷, 比如在高维空间中构建完整的路图或进行胞分解计算量过大, 势场法则容易陷入局部最小值^[9]. 这极大限制了以上算法的使用场景, 如多连杆机械臂或超冗余机器人的运动规划问题^[10,11].

为了解决以上问题, 基于采样的运动规划算法被提出. 这一类算法不需要明确地描述障碍物空间, 而是通过某种策略在无碰撞空间中选取采样点, 进而将这些采样点连接来完成运动规划^[12]. 该类算法的代表

引用格式: 吴昊, 丁焱, 朱向阳. QD-RRT: 基于 Q 距离函数和快速扩展随机树的机械臂运动规划方法. 中国科学: 技术科学, 2022, 52: 773–783
Wu H, Ding Y, Zhu X Y. QD-RRT: A motion planning method for manipulators based on RRT and Q -distance function (in Chinese). Sci Sin Tech, 2022, 52: 773–783, doi: [10.1360/SST-2020-0503](https://doi.org/10.1360/SST-2020-0503)

是由LaValle^[13]于1998年提出的快速扩展随机树(rapidly-exploring random trees, RRT)算法. RRT算法实现简单, 具备概率完备性, 在高维空间中也可以较快的收敛, 这些特性使得它被广泛使用. 在RRT的基础上, 诸多学者对它进行了改进, Kuffner与LaValle^[14]在2000年提出了RRT-Connect, 该算法从初始状态点和目标状态点同时扩展两棵随机树从而实现对状态空间的快速搜索. 在采样的基础上, Karaman和Frazzoli^[15]在2011年提出了渐近最优的概念, 并提出RRT*算法. 该算法在原有RRT算法的基础上增加了Rewire步骤, 在每一次迭代产生新的路径结点后, 通过重新调整已形成的随机树, 使规划出来的路径具备渐近最优性^[16]. Nasir等人^[17]在2013年提出了RRT*-SMART算法, 进一步提高了RRT*算法的收敛速度.

针对具体的应用场景^[18,19], 许多学者对RRT算法做出了改进. 谢碧云等人^[20]针对7R机械臂的运动规划问题, 提出了一种改进的RRT算法. 借助7R机械臂的解析逆解, 生成多个目标位形, 在给定的障碍物环境中, 机器人自动选择某一合适的位形作为目标点引导随机树有效生长. 针对双机械臂协同运动规划的特点, 陈波芝等人^[21]对RRT做出了改进, 将主臂视为从臂的障碍, 并利用节点剪枝择优和设置目标区域的方式使算法快速收敛. 刘伟等人^[22]将RRT算法应用于无人机运动规划, 根据飞行特征设计了随机采样、威胁规避、路径可跟踪性以及全局与局部平滑性优化策略, 构建了快速平滑收敛RRT(QS-RRT).

然而基于RRT的算法在遇到复杂障碍物环境, 或者路径需要穿过狭长通道时, 都会因为采样点落在狭长通道中的概率过低使收敛速度大大降低. 为此, 本文在原始RRT算法的基础上, 引入了Q距离(Q-distance, QD)函数^[23]. 不同于传统距离的定义, 该函数不仅可以表示两分离凸多面体之间的最小间隔距离, 而且也可以表示两相交多面体之间的最大嵌入距离. 结果通过求解一个线性规划得到, 因此计算过程非常高效. 在快速扩展随机树的生长过程与环境障碍物发生碰撞时, 利用Q距离函数计算碰撞位形处机械臂与环境障碍物之间的嵌入距离, 并计算该点Q距离函数关于关节角度的梯度, 对碰撞位形做出适当的修正, 使算法可以快速收敛. 值得注意的是, 该方法可以作为单独的一个模块, 不仅适用于RRT算法, 同样适用于它的

一些改进算法, 例如RRT*.

2 Q距离函数

引入Q距离函数定义^[23], 设 $Q \subset \mathbb{R}^n$ 是一个紧凸集, 且 $0 \in \text{int}Q$. 定义Q的度规函数为 $g_Q(x) = \inf\{\sigma | x \in \sigma Q, \sigma > 0\}$. 可以把 g_Q 理解为一种范数, 称为Q范数, 记为 $\|\cdot\|_Q$. 自然地, 称 $S_Q(\rho) = \{x | \|x\|_Q \leq \rho\}$ 为以原点为中心的Q范数球. 易得, 对于任意的 $\rho \geq 0$, $S_Q(\rho) = \rho Q$. 假设A和B是 $\mathbb{R}^n$ 上的两个紧凸集, 则A所表示的凸多面体与B所表示的凸多面体的Q距离定义为

$$d_Q(A, B) = \begin{cases} \min_{\rho Q \cap (B-A) \neq \emptyset, \rho \geq 0} \rho, & A \cap B = \emptyset, \\ \min_{\rho Q \cap A=B, \rho \geq 0} -\rho, & A \cap B \neq \emptyset. \end{cases} \quad (1)$$

由式(1)可知, 使用Q距离函数计算两凸集之间的距离时, 需要首先判断两凸集是否相交, 即进行碰撞检测. 有多种方法可以解决这一问题^[24,25], 本文使用了经典的碰撞检测算法——GJK算法^[26]. 该方法使用了一种高效策略, 在两凸集的闵可夫斯基差中构造单纯形, 根据单纯形是否包含原点来判断两凸集是否相交.

我们将A, B, Q分别用各自顶点的集合来表示, 即 $A = \{a_i | i = 1, \dots, n_A\}$, $B = \{b_j | j = 1, \dots, n_B\}$, 以及 $Q = \{q_k | k = 1, \dots, n_Q\}$. $d_Q(A, B)$ 的求解过程可以转化为求解线性规划的问题.

如果 $A \cap B = \emptyset$:

$$d_Q(A, B) = \min \sum_{k=1}^{n_Q} \rho_k, \quad \text{s.t.} \begin{cases} \sum_{j=1}^{n_B} \beta_j b_j - \sum_{i=1}^{n_A} \alpha_i a_i = \sum_{k=1}^{n_Q} \rho_k q_k, \\ \sum_{i=1}^{n_A} \alpha_i = 1, \\ \sum_{j=1}^{n_B} \beta_j = 1, \\ \alpha_i \geq 0, \beta_j \geq 0, \rho_k \geq 0. \end{cases} \quad (2)$$

如果 $A \cap B \neq \emptyset$:

$$\begin{aligned}
d_Q(A, B) &= \max(d_k \mid k = 1, \dots, n_Q), \\
d_k &= \min -\rho, \\
\text{s.t. } &\begin{cases} \rho q_k = \sum_{i=1}^{n_A} \alpha_i a_i - \sum_{j=1}^{n_B} \beta_j b_j, \\ \sum_{i=1}^{n_A} \alpha_i = 1, \\ \sum_{j=1}^{n_B} \beta_j = 1, \\ \alpha_i \geq 0, \beta_j \geq 0, \rho \geq 0. \end{cases} \quad (3)
\end{aligned}$$

$\theta \in \mathbb{R}^n$ 表示凸集 A 和 B 的位形参数矢量, 记为 $A(\theta) = \{a_i(\theta) \mid i = 1, \dots, n_A\}$, $B(\theta) = \{b_j(\theta) \mid j = 1, \dots, n_B\}$. $a_i(\theta)$, $b_j(\theta)$ 分别表示凸集 $A(\theta)$, $B(\theta)$ 的顶点, 且都是关于 θ 的光滑函数, 根据 Lipschitz 连续性可得 $d_Q(\theta)$ 关于 θ 几乎处处可微^[27]. 若 $A \cap B = \emptyset$, 设

$$\Phi^* = \{\alpha_1^*, \dots, \alpha_{n_A}^*, \beta_1^*, \dots, \beta_{n_B}^*, \rho_1^*, \dots, \rho_{n_Q}^*\}$$

为式(2)的最优解, 其中设 $\mathcal{B}$ 表征最优解 Φ^* 中的正元素, 定义 $\kappa_A = \{i \mid \alpha_i^* \in \mathcal{B}\}$, $\kappa_B = \{j \mid \beta_j^* \in \mathcal{B}\}$, $\kappa_Q = \{k \mid \rho_k^* \in \mathcal{B}\}$. 则 $d_Q(A_i(\theta), B_j(\theta))$ 关于 θ_τ 的导数为

$$\frac{\partial d_Q(\theta)}{\partial \theta_\tau} = E \cdot P(\theta)^{-1} \left(\sum_{j \in \kappa_B} \beta_j^* \frac{\partial b_j}{\partial \theta_\tau} - \sum_{i \in \kappa_A} \alpha_i^* \frac{\partial a_i}{\partial \theta_\tau} \right), \quad (4)$$

其中, $\theta_\tau \in \mathbb{R}$ 是 $\theta \in \mathbb{R}^n$ 中的第 τ 个元素; $E = \sum_{k=1}^{|\kappa_Q|} e_k^n$, $|\kappa_Q|$ 表示 κ_Q 集合中的元素个数, $e_k^n \in \mathbb{R}^{1 \times n}$ 是除第 k 列为 1 外, 其余列都为 0 的行向量; $P(\theta) = (P_Q, P_A, -P_B) \in \mathbb{R}^{n \times n}$. 其中

$$\begin{aligned}
P_Q &= (q_k)_{k \in \kappa_Q}, \\
P_A &= (a_{i \in \kappa_A}) - a_{i_1} \cdot \mathbf{1}_{1 \times |\kappa_A|}, \\
P_B &= (b_{j \in \kappa_B}) - b_{j_1} \cdot \mathbf{1}_{1 \times |\kappa_B|}, \quad (5)
\end{aligned}$$

$i_1 \in \kappa_A, j_1 \in \kappa_B$, $\bar{\kappa}_A = \kappa_A \setminus \{i_1\}$, $\bar{\kappa}_B = \kappa_B \setminus \{j_1\}$; $|\bar{\kappa}_A|$ 表示 $\bar{\kappa}_A$ 集合中元素个数; $|\bar{\kappa}_B|$ 表示 $\bar{\kappa}_B$ 集合中元素的个数; $\mathbf{1}_{1 \times |\bar{\kappa}_A|}$ 表示一个全为 1 的 $1 \times |\bar{\kappa}_A|$ 行向量, $\mathbf{1}_{1 \times |\bar{\kappa}_B|}$ 同理.

类似地, 当 $A \cap B = \emptyset$ 时, $d_Q(A_i(\theta), B_j(\theta))$ 关于 θ_τ 的导数为

$$\frac{\partial d_Q(\theta)}{\partial \theta_\tau} = e_1^n P(\theta)^{-1} \left(\sum_{j \in \kappa_B} \beta_j^* \frac{\partial b_j}{\partial \theta_\tau} - \sum_{i \in \kappa_A} \alpha_i^* \frac{\partial a_i}{\partial \theta_\tau} \right). \quad (6)$$

3 基于 Q 距离的 RRT 算法

3.1 RRT 算法

RRT 算法^[13]通过随机采样的方式构建一棵随机树 $\mathcal{T}$. 设待搜索的位形空间为 X , 障碍物空间为 $X_{\text{obstacles}} \subseteq X$, 无障碍空间为 $X_{\text{free}} \subseteq X$. 随机树从起始点 x_{init} 处开始生长. 每一次迭代中, 在位形空间中随机选择一点 x_{rand} , 在随机树 $\mathcal{T}$ 已有的路径节点中选择一个距离 x_{rand} 最近的节点 x_{nearest} , 然后以 x_{nearest} 为根节点朝 x_{rand} 生长距离 Δt 得到 x_{new} . 使用碰撞检测算法检测 x_{nearest} 到 x_{rand} 之间路径是否可行, 如果可行则将 x_{new} 加入到随机树 $\mathcal{T}$ 中; 如果路径不可行则将 x_{new} 点废弃. 重复整个过程, 直到新节点与目标节点之间的距离小于设定阈值, 则判定已找到一条从 x_{init} 到 x_{goal} 的路径. RRT 算法的伪代码见算法 1 所示.

3.2 对 RRT 算法的改进

3.2.1 引入目标偏置

分析 RRT 的原理可知, 它的一大特点是对空间的随机搜索, 每一次采样点都是在位形空间中随机采样, 这样的特性保证 RRT 算法可以对整个位形空间进行有效探索, 使得到路径解的可能性大大增加. 然而, 路径探索的盲目性使得随机树过多地扩展到了“无用区域”, 降低了收敛速度. 我们希望随机树尽可能向着目标方向搜索, 以加快搜索速度. 一种常用的做法是引入目标偏置策略^[28], 在每次迭代时, 以一定概率选择 x_{goal} 为 x_{rand} , 即 $x_{\text{rand}} = x_{\text{goal}}$. 伪代码如算法 2 所示.

算法 1 RRT 算法

```

1:  $\mathcal{T} \leftarrow \text{initializeTree}()$ ; //初始化随机树
2:  $\mathcal{T}.\text{addNode}(\emptyset, x_{\text{init}})$ ;
3: for  $k = 1$  to  $K$  do
4:    $x_{\text{rand}} \leftarrow \text{randomSample}()$ ; //随机采样
5:    $x_{\text{nearest}} \leftarrow \mathcal{T}.\text{nearest}(x_{\text{rand}})$ ; //在已有随机树中寻找最近节点
6:    $x_{\text{new}} \leftarrow \text{steer}(x_{\text{nearest}}, x_{\text{rand}}, \Delta t)$ ; //获得新节点
7:   if  $\text{obstacleFree}(x_{\text{nearest}}, x_{\text{new}})$  then
8:      $\mathcal{T}.\text{addNode}(x_{\text{nearest}}, x_{\text{new}})$ ; //如果路径无碰撞, 将该节点加入到随机树中
9:   end if
10: end for
11: return  $\mathcal{T}$ 

```

算法2 引入目标偏置的RRT算法

```

1:  $\mathcal{T} \leftarrow \text{initializeTree}()$ ; // 初始化随机树
2:  $\mathcal{T}.\text{addNode}(\emptyset, x_{\text{init}})$ ;
3: for  $k = 1$  to  $K$  do
4:    $x_{\text{rand}} \leftarrow \text{chooseTarget}(x_{\text{rand}}, x_{\text{goal}}, P)$ ; // 以概率 $P$ 选择 $x_{\text{goal}}$ 为 $x_{\text{rand}}$ 
5:    $x_{\text{nearest}} \leftarrow \mathcal{T}.\text{nearest}(x_{\text{rand}})$ ; // 在已有随机树中寻找最近节点
6:    $x_{\text{new}} \leftarrow \text{steer}(x_{\text{nearest}}, x_{\text{rand}}, \Delta t)$ ; // 获得新节点
7:   if  $\text{obstacleFree}(x_{\text{nearest}}, x_{\text{new}})$  then
8:      $\mathcal{T}.\text{addNode}(x_{\text{nearest}}, x_{\text{new}})$ ; // 如果路径无碰撞, 将该节点加入到随机树中
9:   end if
10: end for
11: return  $\mathcal{T}$ 
    
```

3.2.2 引入避障步骤

RRT算法的随机树在扩展过程中, 如果没有遇到障碍物, 将该节点加入到随机树 $\mathcal{T}$ 中, 如图1(a)所示; 如果遇到障碍物则会将该路径点舍弃, 如图1(b)所示。

在引入目标偏置策略的前提下, 本文提出一种基于Q距离函数的随机树扩展策略。如果随机树 $\mathcal{T}$ 在扩展过程中 x_{nearest} 到 x_{new} 之间的路径没有发生碰撞, 则如原算法一样, 将该点加入到随机树 $\mathcal{T}$ 中; 如果 x_{nearest} 到 x_{new} 之间的路径不可行, 且 $x_{\text{rand}} = x_{\text{goal}}$, 则对碰撞点 $x_{\text{collision}}$ 做出修正, 如图2所示。

具体修正过程如下, 由式(1)给出的Q距离函数公式, 计算机臂在该位形下连杆与环境障碍物发生碰撞的Q距离, 将其定义为

$$\varphi(\theta) = \sum_{(i,j) \in I \times J} d_Q(A_i(\theta), B_j). \quad (7)$$

其中, 对于 $\forall i \in I, j \in J, d_Q(A_i(\theta_{\text{collision}}), B_j) \leq 0$. $\theta \in \mathbb{R}^n$ 是机械臂关节空间参数; $\theta_{\text{collision}} \in \mathbb{R}^n$ 是机械臂与环境

障碍物发生碰撞处的位形; I 是发生碰撞的机械臂连杆集合; $A_i(\theta)$ 表示与环境障碍物发生碰撞的第 i 节机械臂连杆, 它是关于 θ 的光滑函数; J 是发生碰撞的环境障碍物集合; B_j 是与机械臂发生碰撞的第 j 个环境障碍物。按下式计算 $\varphi(\theta)$ 的导数:

$$\frac{d\varphi(\theta)}{d\theta} = \sum_{(i,j) \in I \times J} \frac{d_Q(A_i(\theta), B_j)}{d\theta}. \quad (8)$$

对碰撞点做出如下修正:

$$\theta_{\text{new}} = \theta_{\text{collision}} + s \cdot \left\| \frac{d\varphi(\theta)}{d\theta} \right\|^{-1} \frac{d\varphi(\theta)}{d\theta}, \quad (9)$$

其中, $s > 0$ 表示步长。再次判断 θ_{nearest} 到 θ_{new} 之间的路径是否可行, 如果可行则将该节点加入到随机树 $\mathcal{T}$ 中, 否则重复之前的避障步骤 i 次; 如果路径仍不可行, 则将该点废弃。

将修改后的RRT算法称为QD-RRT。综上所述, QD-RRT算法流程如下所示:

Step 1: 初始化随机树 $\mathcal{T}$;

Step 2: 以概率 P 选取目标点 x_{goal} 为 x_{rand} , 否则在位形空间 X 中随机选取一点作为 x_{rand} ;

Step 3: 从随机树 $\mathcal{T}$ 中选取距离 x_{rand} 最近的一点 x_{nearest} , 从 x_{nearest} 出发, 沿一定距离 Δt 得到 x_{new} ;

Step 4: 使用碰撞检测算法判断 x_{nearest} 到 x_{new} 之间路径的可行性; 如果可行则将该点加入到随机树 $\mathcal{T}$ 中; 否则进入Step 5;

Step 5: 如果 $x_{\text{rand}} \neq x_{\text{goal}}$, 则将点 x_{new} 废弃进入Step 2; 如果 $x_{\text{rand}} = x_{\text{goal}}$ 则使用Q距离函数计算碰撞处的梯度, 产生新的 x_{new} ; 如果 x_{nearest} 到 x_{new} 之间的路径仍然不可行,

图 1 (网络版彩图)RRT算法随机树的扩展过程. (a) 扩展路径可达; (b) 扩展路径不可达

Figure 1 (Color online) The process of RRT extension. (a) Extension path is available; (b) extension path is not available.

图 2 (网络版彩图)基于 Q 距离的避障步骤

Figure 2 (Color online) The process of obstacles avoidance method based on the Q distance function.

则重复Step 5 t 次, 如果仍不可行, 将该点废弃进入 Step 2;

Step 6: 如果 $\|x_{\text{goal}} - x_{\text{new}}\| < \text{threshold}$ 则找到了一条从 x_{init} 到 x_{goal} 的可达路径; 否则进入Step 2.

完整的算法流程如图3所示.

4 仿真验证

硬件平台为Intel(R) Core(TM) i5-9400f CPU主频 2.90 GHz, 内存16 GB, 软件平台为MATLAB 2018a, CoppeliaSim Edu Version 4.1.0.

4.1 平面三连杆运动规划仿真

本节以平面三连杆机构为例, 通过仿真实验将 QD-RRT算法与RRT算法进行对比验证. 因为目标偏置、避障环节可以作为独立的模块, 所以我们将相同的改进方式也引入到了RRT*算法中. RRT*在RRT算法的基础上引入了Rewire步骤^[15], 即当产生新的路径节点 x_{new} 之后, 会以 x_{new} 为中心, r 为邻域半径划定一个范围, 在该范围内的所有节点都是潜在父节点, 遍历每一个潜在父节点, 计算将此节点作为父节点时的代价并进行碰撞检测, 如果这条路径是可行的且代价低于原有路径, 则将该点作为新的父节点, 如图4所示. QD-RRT*并不改变RRT*的Rewire步骤, 只是当 x_{nearest} 与 x_{new} 之间路径不可达时会根据 Q 距离函数的梯度产生一个新的路径节点. 在本例中将QD-RRT*算法与RRT*算法进行对比验证.

为了衡量引入基于 Q 距离的避障策略对优化RRT

图 3 QD-RRT算法流程图

Figure 3 QD-RRT algorithm flow chart.

算法的贡献程度, QD-RRT算法也将与只引入目标偏置策略的RRT算法(P-RRT)进行比较.

平面三连杆初始位形为 θ_{init} , 目标位形为 θ_{goal} . 设置4个障碍物, 如图5所示.

五种算法统一设置步长为 $s=0.2$, 其中QD-RRT与

图 4 RRT*算法Rewire步骤

Figure 4 The Rewire step of RRT*.

图 5 平面三连杆运动规划仿真

Figure 5 Planar 3-DOF motion planning simulation.

QD-RRT*的随机采样概率 $p_{\text{rand}}=0.7$, 即有70%的概率随机采样, 有30%的概率以目标点 θ_{goal} 作为采样点; QD-

RRT与QD-RRT*的避障尝试次数 $t=4$; RRT*与QD-RRT*的邻域半径 $r=0.4$ ^[15]. 我们以最终规划的路径长度以及规划时间作为性能指标, 简单起见, 我们将路径长度定义为 $\sum_i \|\theta_{i+1} - \theta_i\|_2$, $\theta_i \in \mathbb{R}^3$ 为路径点. 五种算法各做20组仿真实验, 限制最长运算时间为300 s. 实验数据对比如图6所示; RRT与QD-RRT的平均数据对比见表1, RRT*与QD-RRT*的平均数据对比见表2, P-RRT与QD-RRT的平均数据对比见表3.

分析仿真结果可以发现, 不论是QD-RRT算法还是QD-RRT*算法, 在引入了目标偏置以及避障环节之后收敛速度以及规划所得路径的质量都得到了提高, 这证明了改进措施的可行性. 分析表3数据可知, 在本例中, 因环境障碍物较为简单, QD-RRT与P-RRT相比, 收敛速度反而因为需要额外计算Q距离函数的梯度降低了11.89%, 但平均路径长度减少了12.28%.

除此之外, 保持起始位形与目标位形不变, 在原有路径上增加第5个障碍物, 分别使用RRT算法、P-RRT算法以及QD-RRT算法进行仿真, 同样分别进行20次仿真实验, 限制每次最长仿真时间为300 s. 在规定时间内, RRT算法没有成功找到可达路径; P-RRT只有4次找到可达路径, 平均耗时为6.3165 s, 平均路径长度为6.0318; QD-RRT在20次仿真实验中全部找到可达路径, 其中一次仿真结果如图7所示, 平均耗时为3.3597 s, 平均路径长度为5.7641.

4.2 平面超冗余机器人运动规划仿真

本节以平面6自由度超冗余机器人为例, 通过仿真

图 6 五种算法耗时(a)和规划路径长度(b)对比

Figure 6 Time-consuming comparison (a) and path-length comparison (b) of five algorithms.

表 1 RRT与QD-RRT的仿真数据对比

Table 1 RRT and QD-RRT simulation comparison

方法	平均耗时 (s)	平均路径长度
RRT	8.6615	8.9957
QD-RRT	3.1674	5.6179
效果对比	减少63.43%	减少37.54%

表 2 RRT*与QD-RRT*的仿真数据对比

Table 2 RRT* and QD-RRT* simulation comparison

方法	平均耗时 (s)	平均路径长度
RRT*	48.2470	7.4950
QD-RRT*	6.8324	4.8453
效果对比	减少85.58%	减少35.35%

表 3 P-RRT与QD-RRT的仿真数据对比

Table 3 P-RRT and QD-RRT simulation comparison

方法	平均耗时 (s)	平均路径长度
P-RRT	2.8307	6.4049
QD-RRT	3.1674	5.6179
效果对比	增加11.89%	减少12.28%

实验将QD-RRT算法与RRT算法进行对比验证, 同样各进行20次, 仿真环境同上. 在300 s内, RRT没有找到可达路径, QD-RRT有5次没有在300 s内找到可达路径, 剩余15次找到可达路径的平均耗时为10.1428 s, 平均路径长度为10.1390, 仿真结果如图8所示.

4.3 UR10运动规划仿真及实物验证

该算法同样适用于三维环境下6自由度机械臂的运动规划问题. 在三维环境中设置三个障碍物, 并设置起始位形、中间位形和终止位形, 分别如图9(a)~(c)所示, 分两段进行运动规划. 根据UR10的外形几何尺寸, 分别使用3个长方体对UR10的前三个连杆进行包络; 因为UR10的4, 5, 6连杆体积较小, 活动范围有限, 使用一个固连在关节4坐标系上的长方体将4, 5, 6连杆包络, 如图10所示. 这加快了碰撞检测以及QD-RRT算法中 Q 距离梯度的计算速率.

使用RRT, P-RRT(只引入目标偏置的RRT算法), QD-RRT分别进行20次仿真实验. 仿真环境和算法参数同上. 在规定的300 s时间内, RRT算法均无法找到

图 7 (网络版彩图)增加障碍物后QD-RRT的仿真结果

Figure 7 (Color online) QD-RRT simulation results after adding obstacles.

图 8 (网络版彩图) QD-RRT应用于平面超冗余机器人上的仿真结果

Figure 8 (Color online) Simulation results of QD-RRT applied to planar hyper-redundant robots.

一条可达路径; P-RRT分别在第8次、第10次、第12次、第13次、第15次、第17次仿真中没有在规定时间内找到可达路径; QD-RRT均可找到可达路径. P-RRT与QD-RRT的耗时对比如图11(a)所示; 规划路径长度对比如图11(b)所示; RRT与QD-RRT的平均数据对比见表4所示.

分析仿真结果可以发现, 引入了目标偏置和基于 Q 距离避障策略的QD-RRT算法与只引入目标偏置的P-RRT算法相比, 在规定时间内规划的成功率更高, 收

图 9 (网络版彩图) UR10运动规划仿真. (a) 起始位形; (b) 中间位形; (c) 终止位形

Figure 9 (Color online) UR10 motion planning simulation. (a) Initial configuration; (b) middle configuration; (c) goal configuration.

图 10 (网络版彩图) UR10的包络模型

Figure 10 (Color online) Envelope model of UR10.

敛速度以及规划所得路径质量更佳. 将QD-RRT在MATLAB中规划所得路径导入到CoppeliaSim仿真环境中, 仿真结果如图12所示.

根据仿真环境中障碍物的布置以及机械臂目标位形, 在真实环境中进行实物试验, 效果如图13所示, 机械臂可以顺利完成避障达点任务, 这验证了数值仿真的结果.

5 总结与展望

本文在RRT算法的基础上, 引入了目标偏置策略以及基于Q距离函数的避障措施, 人为地引导随机树以一定概率向目标方向生长, 且当节点与环境障碍物发生碰撞时, 利用Q距离函数的梯度对碰撞点做出修

图 11 (网络版彩图) P-RRT与QD-RRT耗时(a)和规划路径长度(b)对比

Figure 11 (Color online) P-RRT and QD-RRT time-consuming comparison (a) and path-length comparison (b).

表 4 P-RRT与QD-RRT的仿真数据对比

Table 4 P-RRT and QD-RRT simulation comparison

方法	平均耗时 (s)	平均路径长度
P-RRT	31.0851	3.8748
QD-RRT	15.8870	3.5349
效果对比	减少48.89%	减少8.77%

正, 引导随机树绕过环境障碍物, 避免陷入局部最小值。

我们分别在平面三连杆机构、六连杆机构以及UR10上对QD-RRT算法做出了仿真验证。在三连杆仿真实验中, QD-RRT比RRT收敛速度快63.43%, 规划所得路径长度减少37.54%; QD-RRT*比RRT*收敛速度快85.58%, 规划所得路径长度减少35.35%; 在UR10的仿真实验中, QD-RRT与只引入目标偏置策略的P-RRT相比, 在规定时间内路径规划的成功率更高, 且收敛速度快48.89%。改进后的QD-RRT, QD-RRT*算法与RRT, RRT*算法相比, 收敛速度、路径质量都得到了提高, 证实了本文所述方法的可行性和有效性。

本文提出的QD-RRT算法现阶段主要用于解决机

图 12 QD-RRT应用于UR10上的仿真结果

Figure 12 Simulation result of QD-RRT applied to UR10.

械臂离线编程环境下的运动规划问题, 即在已知环境下对机械臂进行运动规划。理论上, 该算法可以配合机器视觉以及图像处理技术实现实时运动规划^[29,30], 使其具有在未知环境下的运动规划以及遇到动态障碍物时重规划^[31]的能力, 拓展该算法的使用场景, 这将是我们的未来研究方向。

图 13 实物试验效果

Figure 13 Experiment results.

参考文献

- 1 Schwartz J T, Sharir M. On the “piano movers” problem I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Comm Pure Appl Math*, 1983, 36: 345–398
- 2 Minguez J, Lamiroux F, Laumond J-P. Motion planning and obstacle avoidance. In: Siciliano B, Khatib O, eds. *Springer Handbook of Robotics*. Cham: Springer, 2016. 1177–1202
- 3 Canny J. Constructing roadmaps of semi-algebraic sets I: Completeness. *Artif Intell*, 1988, 37: 203–222
- 4 Canny J. Computing roadmaps of general semi-algebraic sets. *Comput J*, 1993, 36: 504–514
- 5 Rimon E, Koditschek D E. Exact robot navigation using artificial potential functions. *IEEE Trans Robot Autom*, 1992, 8: 501–518
- 6 Choset H, Pignon P. Coverage path planning: The boustrophedon cellular decomposition. In: Zelinsky A, ed. *Field and Service Robotics*. London: Springer, 1998. 203–209
- 7 Šeda M. Roadmap methods vs. cell decomposition in robot motion planning. In: *Proceedings of the 6th WSEAS International Conference on Signal Processing, Robotics and Automation*. Corfu Island, Greece, 2007. 127–132
- 8 Preparata F P, Shamos M I. *Computational geometry*. In: *Texts and Monographs in Computer Science*. Berlin: Springer-Verlag, 1985
- 9 Choset H M, Hutchinson S, Lynch K M, et al. *Principles of Robot Motion: Theory, Algorithms, and Implementation*. Cambridge, USA: MIT Press, 2005
- 10 Bulut Y, Konkur E S. A real-time path-planning algorithm with extremely tight maneuvering capabilities for hyper-redundant manipulators. *Eng Sci Tech, Int J*, 2021, 24: 247–258
- 11 Shvalb N, Moshe B B, Medina O. A real-time motion planning algorithm for a hyper-redundant set of mechanisms. *Robotica*, 2013, 31: 1327–1335
- 12 LaValle S M. *Planning Algorithms*. Cambridge: Cambridge University Press, 2006
- 13 LaValle S M. Rapidly-exploring random trees: A new tool for path planning. Research Report, 1998
- 14 Kuffner J J, LaValle S M. RRT-Connect: An efficient approach to single-query path planning. In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. San Francisco, 2000. 995–1001
- 15 Karaman S, Frazzoli E. Sampling-based algorithms for optimal motion planning. *Int J Robot Res*, 2011, 30: 846–894
- 16 Gammell J D, Strub M P. A survey of asymptotically optimal sampling-based motion planning methods. arXiv: 2009.10484
- 17 Nasir J, Islam F, Malik U, et al. RRT*-SMART: A rapid convergence implementation of RRT*. *Int J Adv Robotic Syst*, 2013, 10: 299
- 18 Ye C L, Ma S G, Hui L. An omnidirectional mobile robot (in Chinese). *Sci Sin Inform*, 2011, 41: 181–189 [叶长龙, 马书根, 回丽. 一种全方位移动机器人. *中国科学: 信息科学*, 2011, 41: 181–189]
- 19 Xiong P W, Tong X B, Song A G, et al. Robotic cross-modal generative adversarial network based on variational Bayesian Gaussian mixture noise model (in Chinese). *Sci Sin Inform*, 2021, 51: 104–121 [熊鹏文, 童小宝, 宋爱国, 等. 基于变分贝叶斯高斯混合噪声模型的机器人跨模态生成对抗网络. *中国科学: 信息科学*, 2021, 51: 104–121]
- 20 Xie B Y, Zhao J, Liu Y. Motion planning of reaching point movements for 7R robotic manipulators in obstacle environment based on rapidly-exploring random tree algorithm (in Chinese). *J Mech Eng*, 2012, 48: 63–69 [谢碧云, 赵京, 刘宇. 基于快速扩展随机树的7R机械臂避障达点运动规划. *机械工程学报*, 2012, 48: 63–69]
- 21 Chen B Z, Lu L, Lei X Y, et al. Simultaneous obstacle-avoidance motion planning approach for dual arm robots based on improved RRT algorithm (in Chinese). *China Mech Eng*, 2018, 29: 1220–1226 [陈波芝, 陆亮, 雷新宇, 等. 基于改进快速扩展随机树算法的双机械臂协同避障规划方法. *中国机械工程*, 2018, 29: 1220–1226]
- 22 Liu W, Zheng C, Cai K Y, et al. QS-RRT based motion planning for unmanned aerial vehicles using quick and smooth convergence strategies (in Chinese). *Sci Sin Inform*, 2012, 42: 1403–1422 [刘伟, 郑征, 蔡开元, 等. 快速平滑收敛策略下基于QS-RRT的UAV运动规划. *中国科学: 信息科学*, 2012, 42: 1403–1422]
- 23 Zhu X Y, Wang J. Synthesis of force-closure grasps on 3-D objects based on the Q distance. *IEEE Trans Robot Automat*, 2003, 19: 669–679
- 24 Xu J, Liu Z, Yang C, et al. A pseudo-distance algorithm for collision detection of manipulators using convex-plane-polygons-based representation. *Robot Comput-Integrated Manuf*, 2020, 66: 101993
- 25 Haddadin S, De Luca A, Albu-Schaffer A. Robot collisions: A survey on detection, isolation, and identification. *IEEE Trans Robot*, 2017, 33:

1292–1312

- 26 Gilbert E G, Johnson D W, Keerthi S S. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE J Robot Automat*, 1988, 4: 193–203
- 27 Zhu X Y, Ding H, Xiong Y L. A pseudo minimum translational distance of convex polyhedra: I. Definition and properties (in Chinese). *Sci Sin Tech*, 2001, 31: 128–136 [朱向阳, 丁汉, 熊有伦. 凸多面体之间的伪最小平移距离——I. 定义及其性质. 中国科学E辑: 技术科学, 2001, 31: 128–136]
- 28 LaValle S M, Kuffner J J, Donald B. Rapidly-exploring random trees: Progress and prospects. In: Donald B, Lynch K, Rus D, eds. *Algorithmic and Computational Robotics: New Directions 2000 WAFR*. Boca Raton: CRC Press, 2001. 293–308
- 29 Ren B Y, Wei K, Wu Z Q. Research progress of path planning for visual servoing of robotic manipulator (in Chinese). *J Harbin Inst Tech*, 2018, 50: 1–10 [任秉银, 魏坤, 吴卓琦. 机械臂视觉伺服路径规划研究进展. 哈尔滨工业大学学报, 2018, 50: 1–10]
- 30 Zhang Y F, Xiong R. Real-time vision system for a Ping-Pang robot (in Chinese). *Sci Sin Inform*, 2012, 42: 1115–1129 [章逸丰, 熊蓉. 乒乓球机器人的视觉伺服系统. 中国科学: 信息科学, 2012, 42: 1115–1129]
- 31 Zhang F, Tan D L. Mobile robot real-time motion planning based on the relative coordinates in dynamic and unknown environments (in Chinese). *Robot*, 2004, 26: 434–438 [张凤, 谈大龙. 动态未知环境下基于相对坐标系的移动机器人实时运动规划. 机器人, 2004, 26: 434–438]

QD-RRT: A motion planning method for manipulators based on RRT and Q -distance function

WU Hao, DING Ye & ZHU XiangYang

School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

This paper introduces a goal-biased growing strategy and an obstacle avoidance method based on the Q -distance (QD) function to the rapidly-exploring random trees (RRT) algorithm. In the process of random tree generation, the goal-biased strategy guides the random tree to grow toward the target point with a certain probability. The penetration depth between a robot and environmental obstacles is calculated using the QD function when the robot collides with environment obstacles under some configurations, which are represented by the newly generated path nodes. The gradient of the QD function with respect to each joint angle of the robot is calculated using the differentiability of the QD function. The algorithm, QD-RRT, reduces the blindness and randomness of the RRT algorithm. Verified by MATLAB and CoppeliaSim, QD-RRT has faster computational speed and could generate a shorter path than RRT. The advantages of the improved algorithm are more obvious when the obstacles in the environment are more complicated or the path should pass through a long and narrow channel. This algorithm can be applied to some variants of RRT, such as RRT*.

manipulator, motion planning, Q -distance function, RRT algorithm

doi: [10.1360/SST-2020-0503](https://doi.org/10.1360/SST-2020-0503)