DOI：10.13973/j.cnki.robot.190592

基于引力自适应步长 RRT 的双臂机器人协同路径规划

李 洋，徐 达
（陆军装甲兵学院兵器与控制系，北京 100072）

摘 要：快速扩展随机树（RRT）方法的步长确定过分依赖于程序调试，而且固定的步长会导致碰撞检测失效问题．针对此问题，本文提出一种适用于双臂机器人协同路径规划的引力自适应步长 RRT．首先，通过建立构型空间与工作空间的步长范数不等式，对双臂机器人在工作空间中所产生的步长进行约束，进而确保实现有效的碰撞检测；然后，提出随机树被动生长方法，在保证双臂机器人协同运动的基础上，降低规划空间的维度．最后，在随机树的节点处引入引力函数，加快算法的融合速度．仿真结果表明，引力自适应步长 RRT 方法可对工作空间中的步长进行有效约束，确保算法碰撞检测的有效性．在无碰撞的前提下，引力自适应步长 RRT 方法相比于其他算法减少了迭代次数，降低了运行时间并缩短了路径长度．将所提算法应用于双臂机器人的样机实验，结果表明双臂机器人可在保持位置协同的前提下，完成避障运动，验证了算法的可行性．

关键词：快速随机扩展树；双臂机器人；路径规划；自适应步长；碰撞检测；引力函数
中图分类号：TP242.6 **文献标识码**：A **文章编号**：1002-0446(2020)-05-0606-11

Cooperative Path Planning of Dual-arm Robot Based on Attractive Force Self-adaptive Step Size RRT

LI Yang，XU Da
(*Department of Arms and Control*, *Academy of Army Armored Force*, *Beijing* 100072, *China*)

Abstract: In the RRT (rapidly-exploring random tree) method, the determination of the step size depends too much on program debugging, and collision detection failure may occur due to the fixed step size. For this problem, an attractive adaptive step size RRT for cooperative path planning of the dual-arm robot is proposed. Firstly, the step size norm inequality between the configuration space and the workspace is established to constrain the step size generated by the dual-arm robot in the workspace, and thus the effective collision detection is guaranteed. Then, a passive growth method of random tree is proposed to reduce the dimension of planning space while ensuring cooperative motion of the dual-arm robot. Finally, the attractive force function is introduced at the nodes of the random tree to speed up the fusion of the algorithm. The simulation results show that the attractive force self-adaptive step size RRT method can constrain the step size in the workspace effectively to ensure the effectiveness of collision detection. On the premise of no collision, the attractive adaptive step size RRT method reduces the number of iterations, the running time and the path length compared with other algorithms. The proposed algorithm is applied to the prototype experiment of the dual-arm robot. The experiment results show that the dual-arm robot can complete the obstacle avoidance motion on the premise of maintaining the position coordination, which verifies the effectiveness of the algorithm.

Keywords: rapidly-exploring random tree (RRT); dual-arm robot; path planning; self-adaptive step size; collision detection; attractive force function

1 引言（Introduction）

随着机器人在工业中的广泛应用，单机器人已经无法完成某些任务，例如，协调搬运、协同焊接和救援行动等[1-2]．多机器人系统由此诞生，相比于单机器人，多机器人系统具有更大的负载能力、更高的灵巧度等优势[3-4]．

双臂机器人协同完成某个任务时的一项重要工作是路径规划．目前，国内外学者提出了多种双臂机器人路径规划方法．Andreas 等提出了基于闭链运动学的双机械臂路径规划，以满足机械臂的运动约束[5]．Latombe 等将 PRM（probabilistic road map）应用于多机器人的路径规划，然而自由度和障碍物的增加会降低该方法的运行速度[6]．Lim 等

基金项目：武器装备预先研究项目（41404060201）．
通信作者：徐达，zxyxd@sina.com 收稿/录用/修回：2019-11-05/2019-12-25/2020-02-28

提出了一种面向操作空间的双臂机器人路径规划方法，加快了路径生成速度，但该方法没有考虑双机械臂的碰撞问题[7]．Steven 等提出了基于 APF（artificial potential field）的路径规划算法，提升了双机械臂的避障性能，但该方法易陷入局部最优解[8]．LaValle 提出了快速扩展随机树（RRT）方法，该方法可避免陷入局部最优解，并在节点处进行碰撞检测，已成为处理双臂机器人路径规划问题的主要方法之一．近年来，国内外学者在 RRT 基础上提出了多种改进方法以提高算法的各项性能．Kuffner 等提出了 RRT-connect 方法，该方法利用贪婪算法提高了随机树的生长速度[11]．王坤等在 Kuffner 的基础上在 RRT 中加入了目标偏置函数，进一步提高了算法的搜索速度[12-13]．文 [14-15] 在 RRT 的步长和随机树搜索模式方面做出了不同的改进，以加快双臂机器人路径生成速度．Kim 等提出了一种降维 RRT 方法，该方法根据双臂机器人的任务需求对高维路径规划空间进行降维，使 RRT 算法具有更高的运行效率[16-17]．刘成菊等将势函数加入到 RRT 中，减小了算法的随机性，在局部搜索过程中具有良好的避障效果[18]．然而上述方法只适用于单机械臂或双机械臂非耦合路径规划，对于双臂机器人的协同路径规划，RRT 还需解决以下关键技术．首先，目前针对 RRT 的相关改进研究主要集中在改变步长或者控制随机树的生长方向上，缺乏对工作空间中步长的控制和约束，其结果会导致碰撞检测的失效[19]．其次，对于双机械臂，需要对构型空间中步长所引起的末端执行器和连杆的位移进行约束，以实现机械臂的全局性避障[20]．最后，在随机树生长的过程中如何保证 2 个机械臂协同运动是进行协同路径规划的关键环节[21]．

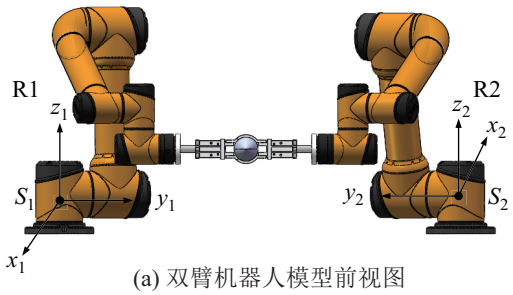
针对上述问题，在文 [22] 的基础上，提出了一种引力自适应步长 RRT 方法，通过自适应步长方法，把随机树每一次生长所引起的机械臂末端执行器与连杆的位移约束在给定范围内，保证碰撞检测的有效性；通过随机树被动生长方法，在机械臂各自的构型空间中控制随机树的生长，实现双机械臂的协调运动；最后，在 RRT 算法的节点处加入引力函数，降低无用节点的产生，提高主动/被动随机树的融合速度．

2　双臂机器人的运动学与约束分析（Kinematics and constraints analysis of the dual-arm robot）

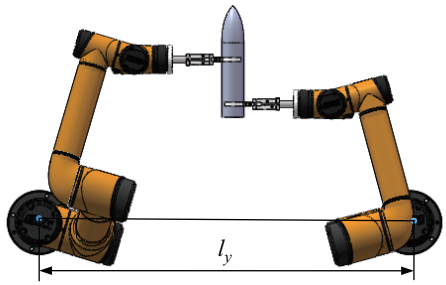
2.1　运动约束

双臂机器人系统由 2 个机器人 R1 和 R2 组成，当双臂机器人协同操作同一个物体时，机器人和物体形成一个闭链系统，机器人之间的位置关系如图 1 所示，S_1、S_2 分别为机器人 R1 和 R2 的基座坐标系，S_1 到 S_2 的位姿变换矩阵为

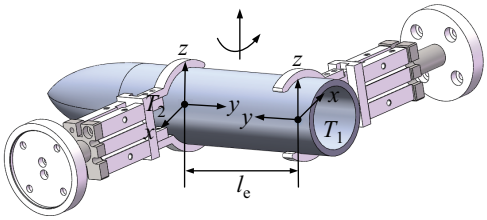
$$ {}^{1}\boldsymbol{S}_2 = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & l_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$



(a) 双臂机器人模型前视图



(b) 双臂机器人模型俯视图



(c) 末端执行器位置关系

图 1　双臂机器人模型

Fig.1　The model of the dual-arm robot

图 1(c) 中 T_1、T_2 分别为机器人 R1 和 R2 的工具坐标系，T_1 到 T_2 的位姿变换矩阵为

$$ {}^{1}\boldsymbol{T}_2 = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & l_e \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2} $$

由式 (1) 和式 (2) 可得到机器人末端执行器之间的约束：

$$ {}^{S_1}\boldsymbol{T}_2 = {}^{S_1}\boldsymbol{T}_1\,{}^{1}\boldsymbol{T}_2 \tag{3} $$

其中 ${}^{S_1}\boldsymbol{T}_1$ 为坐标系 S_1 下机器人 R1 末端执行器的位姿，对于机器人 R2 同理可得其约束为

$$ {}^{S_2}\boldsymbol{T}_2 = {}^{1}\boldsymbol{S}_2\,{}^{S_1}\boldsymbol{T}_2 \tag{4}$$

式 (3) 和式 (4) 为机器人 R1 和 R2 之间的运动约束，同时联系着 R1 和 R2 的运动学，若当已知机器人 R1 的构型或者末端执行器的位置，就可以通过运动约束求解机器人 R2 的正向运动学与逆向运动学，无需再对机器人 R2 进行运动学建模求解.

2.2 正向运动学

建立机器人 R1 的运动旋量模型，如图 2 所示，R1 的惯性坐标系 S 与基座坐标系重合，$\boldsymbol{T}$ 为工具坐标系的位姿，关节 J_1 和 J_5 绕 z 轴方向旋转，其余关节绕 y 轴方向旋转，$\boldsymbol{r}_i$ 为各关节轴线所在位置，建立关节的运动旋量坐标：

$$ \boldsymbol{\xi}_i = [\boldsymbol{\omega}_i \quad \boldsymbol{r}_i \times \boldsymbol{\omega}_i] \tag{5}$$

式中 $\boldsymbol{\omega}_i$ 为机器人第 i 个关节的轴线方向，将关节旋量坐标映射为指数形式并通过指数积求得机器人 R1 的正向运动学：

$$ \boldsymbol{F}_1(\boldsymbol{\theta}) = \mathrm{e}^{\hat{\boldsymbol{\xi}}_1\boldsymbol{\theta}_1}\mathrm{e}^{\hat{\boldsymbol{\xi}}_2\boldsymbol{\theta}_2}\cdots\mathrm{e}^{\hat{\boldsymbol{\xi}}_6\boldsymbol{\theta}_6}\boldsymbol{T} \tag{6}$$

式中，$\boldsymbol{\theta}$ 为机器人关节位置，$\mathrm{e}^{\hat{\boldsymbol{\xi}}\boldsymbol{\theta}}$ 为

$$ \mathrm{e}^{\hat{\boldsymbol{\xi}}\boldsymbol{\theta}} = \begin{bmatrix} \mathrm{e}^{\boldsymbol{\theta}\hat{\boldsymbol{\omega}}} & (\boldsymbol{I}-\mathrm{e}^{\boldsymbol{\theta}\hat{\boldsymbol{\omega}}})(\boldsymbol{\omega}\times\boldsymbol{v}) + \boldsymbol{\theta}\boldsymbol{\omega}\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{v} \\ 0_{1\times3} & 1 \end{bmatrix} \tag{7}$$

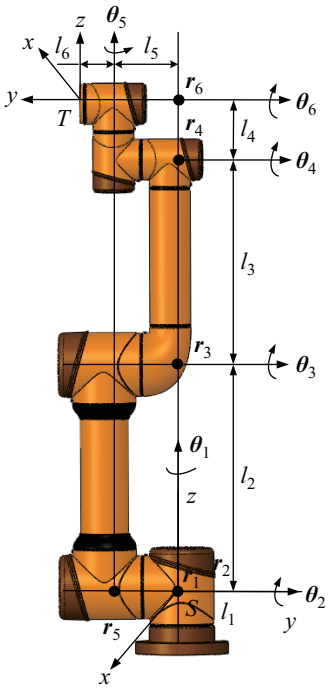


图 2 机器人 R1 的运动旋量模型

Fig.2 The twist model of robot R1

对于机器人 R2 的正向运动学 $\boldsymbol{F}_2(\boldsymbol{\theta})$，无需再次利用指数积公式进行求解，可利用双臂机器人的运动约束进行求解，只需将 $\boldsymbol{F}_1(\boldsymbol{\theta})$ 代入式 (3) 求出 ${}^{S_1}\boldsymbol{T}_2$，再将 ${}^{S_1}\boldsymbol{T}_2$ 代入式 (4) 即可求出 $\boldsymbol{F}_2(\boldsymbol{\theta})$.

2.3 逆向运动学

当机器人 R1 运动到任意位置时，其逆解通常有 8 组，当确定其中的一组解 $\boldsymbol{\theta}_1$ 为机器人 R1 所求的逆解，为了避免机器人 R2 的关节产生过大的位移，其逆解 $\boldsymbol{\theta}_2$ 在满足运动约束的同时还需满足：

$$ \boldsymbol{\theta}_2 = \arg\min_{\boldsymbol{\theta}_2^i} \|\boldsymbol{\theta}_2^i - \boldsymbol{\theta}_2^0\|, \quad i = 1,2,\cdots,8 \tag{8}$$

其中 $\boldsymbol{\theta}_2^0$ 为机器人 R2 运动之前的位置.

3 引力自适应步长 RRT（Attractive adaptive step size RRT）

3.1 步长范数不等式

对于高维运动规划，由于很难在构型空间中建立精确的障碍物模型，RRT 算法的碰撞检测通常只能在工作空间中的节点处进行. 虽然 RRT 的步长在构型空间中是固定的，但通过正向运动学将其映射到工作空间中的步长却不是固定的. 如图 3 所示，随机树以固定的步长 $\Delta\theta$ 在构型空间中生长，当随机树由节点 node1 生长到节点 node2 时，所产生的步长 p_{12} 大于障碍物的尺寸，由于在节点 node1 和 node2 处并未发生碰撞，算法会通过此次碰撞检测，然而机器人在由节点 p_1 向节点 p_2 的运动过程中发生了碰撞，因此对于高维运动规划，固定步长 RRT

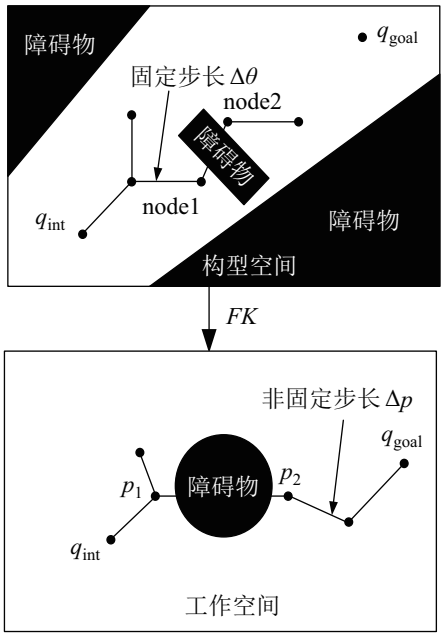


图 3 构型空间和工作空间中的步长关系

Fig.3 Relationship of the step size between the configuration space and the workspace

方法无法确保生成无碰撞路径.

为确保碰撞检测的有效性，需把 RRT 算法在工作空间中的步长限制在最小障碍物的尺寸范围内，引入范数不等式，当给定 2 个向量范数 $\|\cdot\|_{\mathrm{a}}\in \mathrm{R}^{m}$、$\|\cdot\|_{\mathrm{b}}\in \mathrm{R}^{n}$ 和一个矩阵 $\boldsymbol{A}\in \mathrm{R}^{m\times n}$，从属于 2 个向量范数的矩阵算子范数为

$$\|\boldsymbol{A}\|_{\mathrm{a,b}}=\max_{x\neq 0}\frac{\|\boldsymbol{Ax}\|_{\mathrm{a}}}{\|\boldsymbol{x}\|_{\mathrm{b}}} \tag{9}$$

进而可得到范数不等式

$$\|\boldsymbol{Ax}\|_{\mathrm{a}}\leqslant\|\boldsymbol{A}\|_{\mathrm{a,b}}\cdot\|\boldsymbol{x}\|_{\mathrm{b}} \tag{10}$$

机器人在工作空间中的步长为 $\Delta\boldsymbol{p}\in \mathrm{R}^{3\times 1}$，在构型空间中步长为 $\Delta\boldsymbol{\theta}\in \mathrm{R}^{6\times 1}$，根据式 (10)，只需建立从属于 $\Delta\boldsymbol{\theta}$ 和 $\Delta\boldsymbol{p}$ 的矩阵算子范数就可构造构型空间和工作空间的范数不等式，即找出一个矩阵 $\boldsymbol{A}\in \mathrm{R}^{3\times 6}$，满足 $\Delta\boldsymbol{p}=\boldsymbol{A}\Delta\boldsymbol{\theta}$ 即可通过式 (10) 约束 $\Delta\boldsymbol{p}$ 的最大值，达到控制工作空间中步长的目的. 矩阵 $\boldsymbol{A}$ 的构造过程如下：设机器人末端执行器的位置为 $\boldsymbol{P}$，其速度 $\dot{\boldsymbol{P}}$ 为

$$\dot{\boldsymbol{P}}=\boldsymbol{\omega}_{\mathrm{s}}\times\boldsymbol{P}+\boldsymbol{v}_{\mathrm{s}} \tag{11}$$

式中：

$$\boldsymbol{v}_{\mathrm{s}}=[\boldsymbol{r}_1'\times\boldsymbol{\omega}_1'\quad \boldsymbol{r}_2'\times\boldsymbol{\omega}_2'\quad\cdots\quad \boldsymbol{r}_6'\times\boldsymbol{\omega}_6']\dot{\boldsymbol{\theta}}_i \tag{12}$$

$$\boldsymbol{\omega}_{\mathrm{s}}=[\boldsymbol{\omega}_1'\quad\boldsymbol{\omega}_2'\quad\cdots\quad\boldsymbol{\omega}_6']\dot{\boldsymbol{\theta}}_i \tag{13}$$

其中 $\boldsymbol{\omega}_i'$ 和 $\boldsymbol{r}_i'$ 为机器人当前位置的轴线方向和轴线位置，根据式 (12) 和式 (13)，式 (11) 式可改写为

$$\begin{aligned}\dot{\boldsymbol{P}}&=\sum_{i=1}^{6}(\boldsymbol{\omega}_i'\times\boldsymbol{P}+\boldsymbol{v}_i')\dot{\boldsymbol{\theta}}_i\\&=[\hat{\boldsymbol{\omega}}_1'\boldsymbol{p}+\boldsymbol{v}_1'\quad\cdots\quad\hat{\boldsymbol{\omega}}_6'\boldsymbol{p}+\boldsymbol{v}_6']\dot{\boldsymbol{\theta}}\end{aligned} \tag{14}$$

其中 $\hat{\boldsymbol{\omega}}_i'$ 为 $\boldsymbol{\omega}_i'$ 的反对称矩阵，上式两边同乘 Δt 可得：

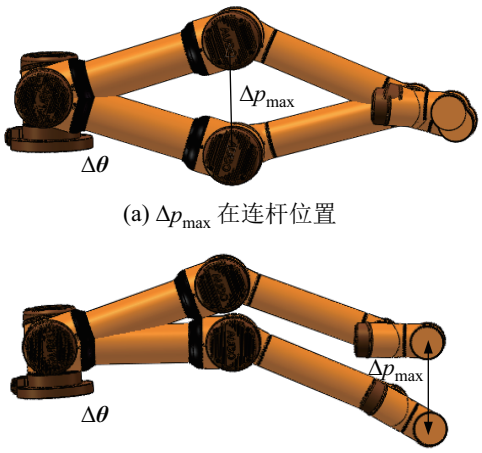
$$\Delta\boldsymbol{p}=[\hat{\boldsymbol{\omega}}_1'\boldsymbol{p}+\boldsymbol{v}_1'\quad\cdots\quad\hat{\boldsymbol{\omega}}_6'\boldsymbol{p}+\boldsymbol{v}_6']\Delta\boldsymbol{\theta} \tag{15}$$

则矩阵 $\boldsymbol{A}=[\hat{\boldsymbol{\omega}}_1'\boldsymbol{p}+\boldsymbol{v}_1'\quad\cdots\quad\hat{\boldsymbol{\omega}}_6'\boldsymbol{p}+\boldsymbol{v}_6']$，可得到构型空间和工作空间范数不等式：

$$\|\Delta\boldsymbol{p}\|_W\leqslant\|\boldsymbol{A}\|_{W,C}\cdot\|\Delta\boldsymbol{\theta}\|_C \tag{16}$$

式中，$\|\Delta\boldsymbol{p}\|_W$ 为工作空间上的范数，$\|\Delta\boldsymbol{\theta}\|_C$ 为构型空间上的范数，取 $W=2$，则 $\|\Delta\boldsymbol{p}\|_2=\sqrt[2]{\Delta p_x^2+\Delta p_y^2+\Delta p_z^2}$ 为工作空间中的位移大小，由于机器人 R1 和 R2 的构型空间为 6 维，$\|\Delta\boldsymbol{\theta}\|_C$ 没有明确的物理意义，为了方便计算 $\|\boldsymbol{A}\|_{W,C}$，令 $C=2$，则 $\|\boldsymbol{A}\|_{W,C}=\|\boldsymbol{A}\|_{2,1}$. 若在工作空间中设定了步长最大值，则在构型空间中每一个步长 $\Delta\boldsymbol{\theta}$ 都会自动调整以满足式 (16)，所引起的位移 $\|\Delta\boldsymbol{p}\|_2$ 总是小于步长最大值 $\Delta p_{\max}$.

在式 (16) 中 $\Delta\boldsymbol{p}$ 特指双臂机器人末端执行器的位移，当在构型空间中给定一个步长 $\Delta\boldsymbol{\theta}$ 时，机器人发生位移的最大位置不一定为末端执行器，也有可能发生在关节处，如图 4 所示.



(a) $\Delta p_{\max}$ 在连杆位置

(b) $\Delta p_{\max}$ 在末端执行器位置

图 4　机器人最大位移的位置

Fig.4　The maximum displacement position of robot

为了保证机器人连杆得到有效的避障，对随机树的每一步生长所引起的机器人连杆最大位移进行约束，使其小于最小障碍物的尺寸 $\|\Delta\boldsymbol{p}_{\max}\|_W$，利用长方体等效包围盒对连杆进行模型等效，设第 i 个连杆的顶点位置集合 $L_i=\{\boldsymbol{p}_{i1},\boldsymbol{p}_{i2},\cdots,\boldsymbol{p}_{i8}\}$，$i=1,2,\cdots,6$，则产生最大位移的位置必然在集合 L_i 中（认为机器人的末端位置在集合 L_6 中），故：

$$\|\Delta\boldsymbol{p}_{\max}\|_W=\max\|\Delta\boldsymbol{p}_{ij}\|_W \tag{17}$$

将式 (17) 代入式 (16) 可得：

$$\max\|\Delta\boldsymbol{p}_{ij}\|_W\leqslant\max\|\boldsymbol{A}_{ij}\|_{W,C}\|\Delta\boldsymbol{\theta}\|_C \tag{18}$$

式中，$\boldsymbol{A}=[\hat{\boldsymbol{\omega}}_1'\boldsymbol{p}_{ij}+\boldsymbol{v}_1'\quad\cdots\quad\hat{\boldsymbol{\omega}}_6'\boldsymbol{p}_{ij}+\boldsymbol{v}_6']\in \mathrm{R}^{24\times 6}$. 进而可得双臂机器人构型空间和工作空间范数不等式：

$$\|\Delta\boldsymbol{p}_{ij}\|_W\leqslant\max\|\boldsymbol{A}_{ij}\|_{W,C}\|\Delta\boldsymbol{\theta}\|_C \tag{19}$$

通过式 (19) 可以在构型空间中控制机器人 R1 和 R2 的最大位移 $\max\|\Delta\boldsymbol{p}_{ij}\|$，完成双机械臂的全局性避障.

3.2 随机树被动生长方法

双臂机器人协同路径规划空间的维数为 12，由于维数的增加导致算法的计算复杂度陡增，算法无法在规定的时间内生成路径，若在机器人各自的构型空间中进行规划，虽然可保证算法的运行速度，但无法完成双机器人的运动协调．为了在满足双臂机器人协同运动的前提下，降低路径规划空间的维度，在 RRT 算法中引入随机树被动生长方法，其原理如图 5 所示．

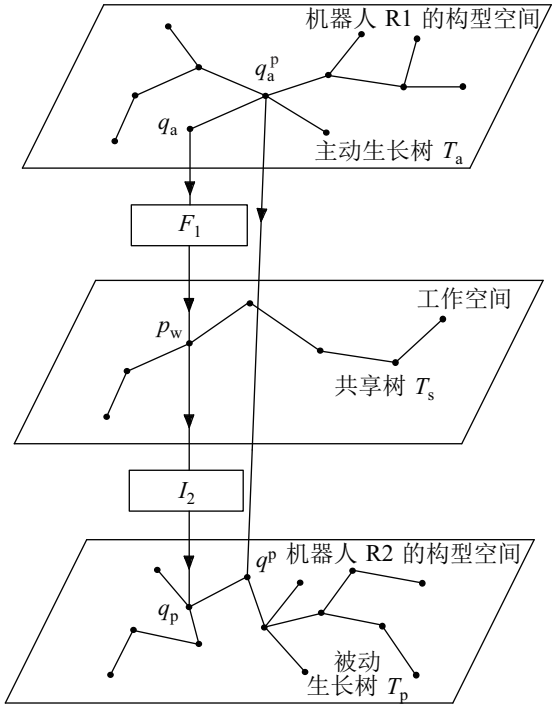


图 5 随机树被动生长方法

Fig.5 The passive-growing method of the random tree

令机器人 R1 的随机树为主动生长随机树 T_a，机器人 R2 的随机树为被动生长随机树 T_p，T_w 为机器人 R1 和 R2 在工作空间中的共享随机树．该方法首先进行主动随机树的生长，当 T_a 在机器人 R1 的构型空间中产生新的节点 q_a 时，通过正向运动学算法将 q_a 映射到工作空间中并生成 T_w 的新节点 p_w：

$$\boldsymbol{F}_1(\boldsymbol{q}_a)=\boldsymbol{p}_w \tag{20}$$

式中 $\boldsymbol{F}_1$ 是机器人 R1 的正向运动学，根据式 (4) 可知，机器人 R2 的随机树 T_p 的新节点 q_p 必须满足运动协同关系：

$${}^1\boldsymbol{T}_2\,{}^1\boldsymbol{S}_2\boldsymbol{F}_1(\boldsymbol{q}_a)=\boldsymbol{F}_2(\boldsymbol{p}_p) \tag{21}$$

式中 $\boldsymbol{F}_2$ 是机器人 R2 的正向运动学，进而可以得到：

$$\boldsymbol{q}_p=\boldsymbol{I}_2({}^1\boldsymbol{T}_2\,{}^1\boldsymbol{S}_2\boldsymbol{p}_w) \tag{22}$$

式中 $\boldsymbol{I}_2$ 为机器人 R2 的逆向运动学，其解通常有 8 组，其中正确的期望解需满足以下条件：

$$\boldsymbol{q}_p=\arg\min_{\boldsymbol{\theta}_i}\|\boldsymbol{\theta}_i-\boldsymbol{q}^p\|_1,\quad i=1,2,\cdots,8 \tag{23}$$

式中 $\boldsymbol{\theta}_i$ 为 8 组机器人 R2 的逆解，$\|\cdot\|_1$ 表示构型空间中的 1 范数，$\boldsymbol{q}^p$ 为 q_p 的父节点坐标，可通过 T_a 中节点 q_a 的父节点 q_a^p 确定．每当 T_a 进行生长产生新节点时，T_p 的新节点会自动通过式 (22) 和式 (23) 确定，故机器人 R2 的随机树被动地跟随机器人 R1 的随机树进行生长，因此只需在机器人 R1 的构型空间中进行 T_a 的生长，算法会自动完成对随机树 T_p 和 T_w 的生长．如果 q_p 通过了碰撞检测，算法会将其添加到 T_p 中，此时，T_a、T_p 和 T_w 完成一次生长，否则，q_a 无效，算法会重新进行机器人 R1 的构型抽样．

3.3 引力函数

在 RRT 方法中，节点是均匀随机产生的，导致随机树的生长具有很强的随机性，算法通常会产生大量的无用节点．在算法引入了自适应步长和随机树被动生长方法后，在节点处除了完成对碰撞检测的计算，还需进行运动学与步长的计算．过多的节点会增加算法的运行时间，为了减少随机树产生过多的无用节点，加快算法的融合速度，将引力函数加入 RRT 算法中，引导随机树朝着目标点方向生长，如图 6 所示．

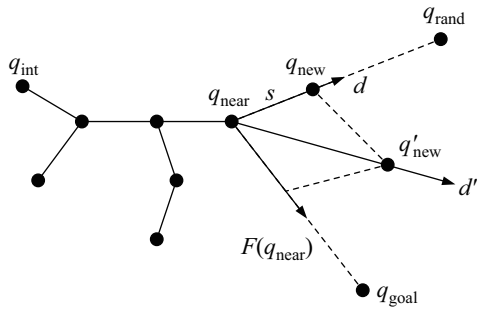


图 6 加入引力函数的 RRT

Fig.6 The RRT integrating the attractive function

RRT 在进行随机树生长时首先在构型空间中产生一个随机的节点 q_{rand}，其父节点为 q_{near}，而后随机树在节点 q_{near} 处以步长长度 s 向节点 q_{rand} 生长并产生新的节点 q_{new}，其生长方向为

$$\boldsymbol{d}=\frac{\boldsymbol{q}_{rand}-\boldsymbol{q}_{near}}{\|\boldsymbol{q}_{rand}-\boldsymbol{q}_{near}\|} \tag{24}$$

其中，$\|\cdot\|$ 为构型空间中的范数，q_{goal} 在 q_{near} 处的引力势能为

$$U=\frac{1}{2}k\|\boldsymbol{q}_{goal}-\boldsymbol{q}_{near}\|^2 \tag{25}$$

其中，k 为引力系数，可得到 q_{goal} 对 q_{near} 的引力为

$$F = k \cdot \|\boldsymbol{q}_{\text{goal}} - \boldsymbol{q}_{\text{near}}\| \tag{26}$$

进而得到目标点对任意节点 q 的引力函数：

$$\boldsymbol{F}(\boldsymbol{q}) = k \cdot \frac{\boldsymbol{q}_{\text{goal}} - \boldsymbol{q}}{\|\boldsymbol{q}_{\text{goal}} - \boldsymbol{q}\|} \tag{27}$$

将引力函数作用在 q_{near} 上，使得 q_{near} 在一定程度上偏向 q_{goal}，进而得到新的生长方向：

$$\boldsymbol{d}' = \frac{\boldsymbol{q}_{\text{rand}} - \boldsymbol{q}_{\text{near}}}{\|\boldsymbol{q}_{\text{rand}} - \boldsymbol{q}_{\text{near}}\|} + k \cdot \frac{\boldsymbol{q}_{\text{goal}} - \boldsymbol{q}_{\text{near}}}{\|\boldsymbol{q}_{\text{goal}} - \boldsymbol{q}_{\text{near}}\|} \tag{28}$$

此时新的节点坐标为

$$\boldsymbol{q}'_{\text{new}} = \boldsymbol{q}_{\text{near}} + s \cdot \boldsymbol{d}' \tag{29}$$

加入了引力函数后，随机树会在一定程度上朝着目标生长，可以通过引力系数 k 来控制生长方向．当 k 较大时，目标对随机树的引力作用较大，随机树会朝着目标生长，会加快随机树的搜索速度；当 k 较小时，随机树的生长相对发散，有利于绕开障碍物．可根据 $\|\boldsymbol{q}_{\text{goal}} - \boldsymbol{q}_{\text{near}}\|$ 的大小来调整系数 k，以确保算法在 q_{goal} 附近不发生振荡，本文将采用 Bi-RRT 方法，通过引入引力函数，2 个随机树会朝着彼此生长，加快随机树的融合．

4　算法实现（The algorithm implementation）

通过上述方法建立引力自适应步长 RRT，首先设定工作空间中所允许的最大步长值 $\Delta p_{\max}$：

$$\Delta p_{\max} = \max \|\boldsymbol{A}_{ij}\|_{W,C} \|\Delta\boldsymbol{\theta}\|_C \tag{30}$$

根据式 (19) 有

$$\|\Delta\boldsymbol{p}_{ij\max}\|_W \leqslant \max \|\boldsymbol{A}_{ij}\|_{W,C} \|\Delta\boldsymbol{\theta}\|_C = \Delta p_{\max} \tag{31}$$

可得到构型空间中的步长：

$$\|\Delta\boldsymbol{\theta}\|_C = \frac{\|\Delta\boldsymbol{p}_{\max}\|_W}{\max \|\boldsymbol{A}_{ij}\|_{W,C}} \tag{32}$$

算法的每次迭代会更新 $\|\boldsymbol{A}_{ij}\|_{W,C}$，工作空间中的步长会自动被约束在 $\|\Delta\boldsymbol{p}_{\max}\|_W$ 的范围内．引力自适应步长 RRT 的伪代码见算法 1.

算法 1 为主程序，在第 1～3 行中，$firstT_{\text{a}}$ 和 $secondT_{\text{a}}$ 是机器人 R1 在构型空间中分别以起始点 q_{int} 和目标点 q_{goal} 为根的随机树，为主动生长树；$firstT_{\text{p}}$ 和 $secondT_{\text{p}}$ 是机器人 R2 在构型空间中分别以 q^2_{int} 和 q^2_{goal} 为根的随机树，为被动生长树；$firstT_{\text{w}}$ 和 $secondT_{\text{w}}$ 是机器人 R1 和 R2 在工作空间中的共享随机树．第 5～7 行通过引力函数 ActiveForce 确定了随机树的生长方向，其伪代码见算法 2.

算法 1：attractive force self-adaptive step size RRT

1 $firstT_{\text{a}} = q^1_{\text{int}}$, $secondT_{\text{a}} = q^1_{\text{goal}}$, $\|\Delta\boldsymbol{p}_{\max}\|_W$
2 $firstT_{\text{p}} = q^2_{\text{int}}$, $secondT_{\text{p}} = q^2_{\text{goal}}$
3 $firstT_{\text{w}} = p_{\text{int}}$, $secondT_{\text{w}} = p_{\text{goal}}$
4 **While** $n < n_{\max}$ **do**
5 $q^1_{\text{rand}} = \text{sampling}()$
6 $q^1_{\text{near}} = \text{GetParentNodeOnTree}(firstT_{\text{a}}, q^1_{\text{rand}})$
7 $\boldsymbol{d}' = \text{AttrativeForce}(\boldsymbol{q}^1_{\text{near}}, \boldsymbol{q}^1_{\text{rand}}, \boldsymbol{q}^1_{\text{goal}})$
8 $\|\Delta\boldsymbol{\theta}\|_C = \text{Stepsize}(\|\Delta\boldsymbol{p}_{\max}\|_W, \|\boldsymbol{A}_{ij}\|_{W,C}, q^1_{\text{near}})$
9 $q^1_{\text{new}} = \text{ActiveGrowTree}(firstT_{\text{a}}, q^1_{\text{near}}, \boldsymbol{d}, \|\Delta\boldsymbol{\theta}\|_C)$
10 $q^2_{\text{new}} = \text{PassiveGrowTree}(firstT_{\text{a}}, firstT_{\text{p}}, q^1_{\text{new}})$
11 **if** CollisionFree $(q^1_{\text{new}}, q^2_{\text{new}}, \|\Delta\boldsymbol{p}_{\max}\|_W) = \text{True}$
12 TreeAdd$(q^1_{\text{new}}, firstT_{\text{a}})$
13 TreeAdd$(q^2_{\text{new}}, firstT_{\text{p}})$
14 TreeAdd$(p_{\text{new}}, firstT_{\text{w}})$
15 **if** TreesMerge$(q^1_{\text{new}}, q^2_{\text{new}}, secondT_{\text{a}}, secondT_{\text{p}})$=True
16 **return** $PathinCspace^1$ $(firstT_{\text{a}}, secondT_{\text{a}})$
17 **return** $PathinCspace^2$ $(firstT_{\text{p}}, secondT_{\text{p}})$
18 **return** $PathinWspace$ $(firstT_{\text{w}}, secondT_{\text{w}})$
19 **else**
20 ExchangeTrees$(firstT_{\text{a}}, secondT_{\text{a}})$
21 ExchangeTrees$(firstT_{\text{p}}, secondT_{\text{p}})$
22 ExchangeTrees$(firstT_{\text{w}}, secondT_{\text{w}})$
23 **end if**
24 **end if**
25 **End while**
26 **Return Null**

算法 2：AttrativeForce$(\boldsymbol{q}^1_{\text{near}}, \boldsymbol{q}^1_{\text{rand}}, \boldsymbol{q}^1_{\text{goal}})$

1 $\boldsymbol{d} = \text{GetDrection}(\boldsymbol{q}^1_{\text{near}}, \boldsymbol{q}^1_{\text{rand}})$
2 $k = \text{Getk}(\boldsymbol{q}^1_{\text{near}}, \boldsymbol{q}^1_{\text{goal}})$
3 $\boldsymbol{F}_{\text{att}} = k \cdot \dfrac{\boldsymbol{\theta}^1_{\text{goal}} - \boldsymbol{\theta}^1_{\text{near}}}{\|\boldsymbol{\theta}^1_{\text{goal}} - \boldsymbol{\theta}^1_{\text{near}}\|}$
4 $\boldsymbol{d}' = \boldsymbol{d} + \boldsymbol{F}_{\text{att}}$
5 **Return** $\boldsymbol{d}'$

在确定了生长方向之后，算法会通过 Stepsize 函数计算步长 $\|\Delta\boldsymbol{\theta}\|_C$，定义算子范数 $\|\boldsymbol{A}_{ij}\|_{1,2}$ 为

$$\|\boldsymbol{A}_{ij}\|_{1,2} = \max\left(\sum_i |a_{ij}|^2\right)^{1/2} \tag{33}$$

Stepsize 函数的伪代码如算法 3 所示.

在得到步长大小与方向之后，作为主动生长随机树的 $firstT_{\text{a}}$ 进行生长并产生新的节点 q^1_{new}，被动生长树 $firstT_{\text{p}}$ 通过随机树被动生长方法进行生长并产生新的节点 q^2_{new}，其伪代码见算法 4.

算法 3：Stepsize($\|\Delta \boldsymbol{p}_{\max}\|_W$, $\|\boldsymbol{A}_{ij}\|_{1,2}$, q^1_{near})

$A = \max \|\boldsymbol{A}_{ij}(q^1_{\text{near}})\|_{1,2}$

$\|\Delta \boldsymbol{\theta}\|_C = \dfrac{\|\Delta \boldsymbol{p}_{\max}\|_W}{A}$

Return $\|\Delta \boldsymbol{\theta}\|_C$

算法 4：PassiveGrowTree($firstT_{\text{a}}$, $firstT_{\text{p}}$, q^1_{new})

$T_{\text{a}} = firstT_{\text{a}}$

$T_{\text{p}} = firstT_{\text{p}}$

$q_{\text{a}} = q^1_{\text{new}}$

$p_{\text{w}} = F_1(q_{\text{a}})$

$q^{\text{p}}_{\text{a}} = \text{GetParentNodeOnTree}(q_{\text{a}}, T_{\text{a}})$

$q^{\text{p}} = \text{GetAccordingNodeOnTree}(q^{\text{q}}_{\text{a}}, firstT_{\text{a}})$

$q_{\text{p}} = I_2(p_{\text{w}}, q^{\text{p}})$

$q^2_{\text{new}} = q_{\text{p}}$

Return q^2_{new}

若 p^1_{new} 与随机树 $secondT_{\text{w}}$ 上最近节点 p^1_{near} 的距离小于 $\|\Delta \boldsymbol{p}_{\max}\|_W$，随机树 $firstT_{\text{w}}$ 和 $secondT_{\text{w}}$ 开始进行融合，其过程如图 7 所示.

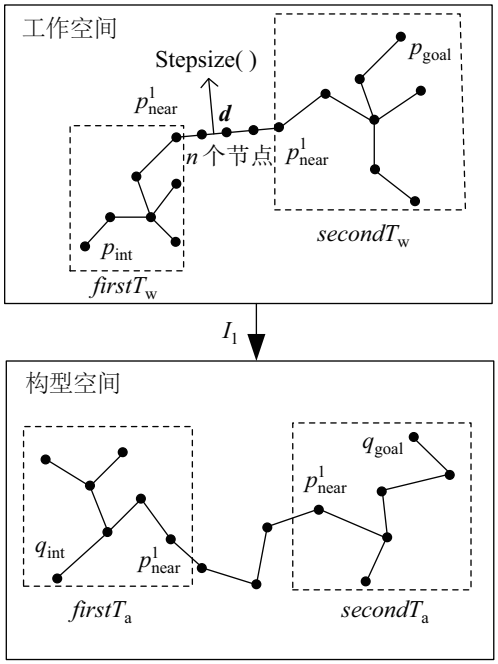


图 7 随机树融合

Fig.7 Merging of random trees

首先算法在随机树 $secondT_{\text{w}}$ 中找到离节点 p^1_{new} 最近的点进而确定随机树的融合方向. 而后在节点 p^1_{new} 与节点 p^1_{near} 之间插入 n 个节点，相邻节点距离通过自适步长函数 Stepsize 确定. 最后算法将对这些节点进行碰撞检测，若节点通过检测则随机树融合成功，主程序结束并生成新路径；否则随机树继续生长并再次进行融合，直至找出一条无碰撞路径，随机树融合的伪代码见算法 5.

算法 5: TreesMerge(p^1_{new}, $firstT_{\text{a}}$, $secondT_{\text{a}}$, $\|\Delta \boldsymbol{p}_{\max}\|_W$)

flag = true

[p^1_{near}, idx] = NearestNodeOnTree (p^1_{new}, $secondT_{\text{a}}$)

$q^1_{\text{near}} = I_1(p^1_{\text{near}}, secondT_{\text{a}})$

$d = \text{GetDirection}\ (p^1_{\text{near}}, p^1_{\text{new}})$

while flag = true

  $s = \text{Stepsize}(\|\Delta \boldsymbol{p}_{\max}\|_W, \|\boldsymbol{A}_{ij}\|_{1,2}, q^1_{\text{near}})$

  $t = p^1_{\text{new}} + s \cdot d$

  $q^1_{\text{temp}} = I_1(t)$

  $q^2_{\text{temp}} = I_2(t)$

  if CheckCollision & JointLimits (q^1_{temp}, q^2_{temp})=True

    if Distance (t, p^1_{near}) > $\|\Delta \boldsymbol{p}_{\max}\|_W$

      $p^1_{\text{new}} = t$

      continue

    else

      break

    end if

  else

    flag = false

  end if

end while

5 仿真与实验（Simulation and experiment）

5.1 单次仿真实验

将引力自适应步长 RRT 与 3 种固定步长的 Bi-RRT [23] 和一种自适应 RRT [24] 作对比，为了实现双臂机器人的协同路径规划，将 3.2 节中的随机树被动生长方法引入到 Bi-RRT 和自适应 RRT 方法中. 双臂机器人的仿真模型如图 8 所示.

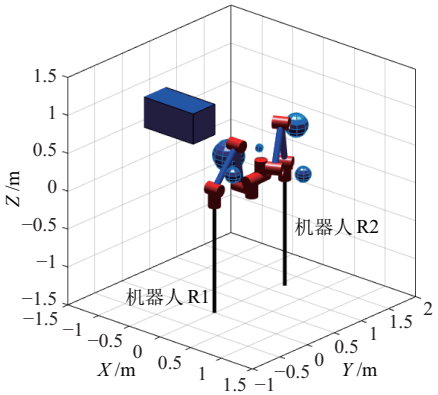


图 8 双臂机器人仿真模型

Fig.8 Simulation model of the dual-arm robot

在双臂机器人的工作空间中分别放置一个尺寸为 1.6 m×0.4 m×0.4 m 的长方体障碍物和 5 个直径分别为 0.05 m、0.1 m、0.1 m、0.15 m、0.2 m 的

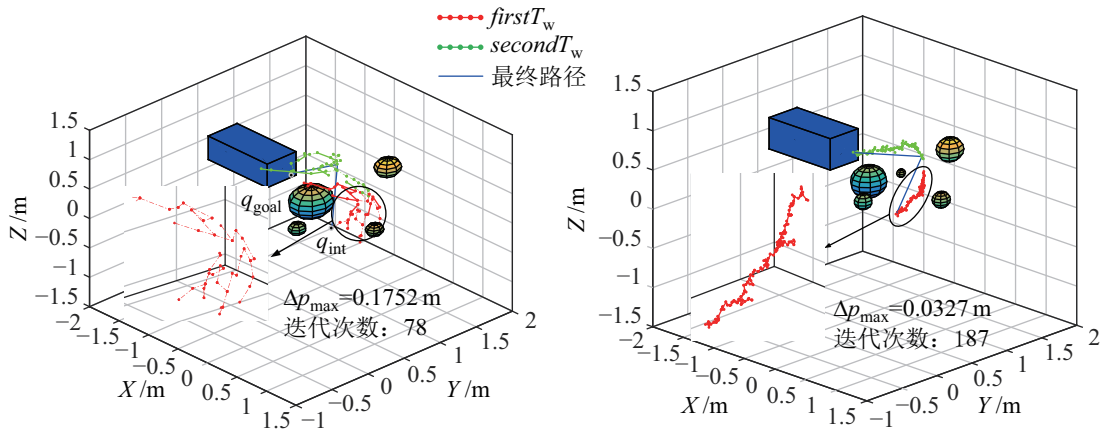
球体障碍物．最小障碍物的尺寸为 0.05 m，所以 $\|\Delta \boldsymbol{p}_{\max}\|_W = 0.05$，双臂机器人在工作空间中的最大步长不能超过 $\|\Delta \boldsymbol{p}_{\max}\|_W$，否则碰撞检测失效．起始点与目标点的坐标分别为 (0.15 m, 0.6 m, 0) 和 (−0.55 m, 0.6 m, 0.6 m)．在路径生成后，算法对其进行后处理去掉路径中冗余的节点．算法在 Matlab 中实现，运行平台配置为 Intel Core I7-8700 3.20 GHz，16 GB RAM．分别利用引力自适应步长 RRT 算法和步长为 0.5 rad 的 Bi-RRT 算法在上述模型中进行路径搜索并进行对比，结果如图 9 所示．

图 9(a) 中固定步长 Bi-RRT 算法在工作空间中的步长 Δp 较大，虽然算法迭代次数较少，但最大步长值 $\Delta p_{\max}$ 为 0.1752 m，大于 $\|\Delta \boldsymbol{p}_{\max}\|_W$，无法保证碰撞检测的有效性．图 9(b) 中引力自适应步长 RRT 算法的步长相对稳定，$\Delta p_{\max}$ 为 0.0327 m，小于 $\|\Delta \boldsymbol{p}_{\max}\|_W$，故该方法所生成的路径是无碰撞的．

将 Bi-RRT 的步长降至 0.3 rad，并同引力自适应步长 RRT 方法进行路径搜索，其结果如图 10 所示．

图 10(a) 中固定步长 Bi-RRT 在工作空间中的平均步长显著减小，同时迭代次数也随之增加，最大步长 $\Delta p_{\max}$ 为 0.062 m，大于 $\|\Delta \boldsymbol{p}_{\max}\|_W$，无法保证碰撞检测的有效性，需要进一步减小步长．图 10(a) 中引力自适应步长 RRT 的最大步长 $\Delta p_{\max}$ 为 0.0428 m，仍然小于 $\|\Delta \boldsymbol{p}_{\max}\|_W$．进一步减小 Bi-RRT 的步长至 0.1 rad 并再次同引力自适应步长 RRT 方法进行路径搜索，其结果如图 11 所示．

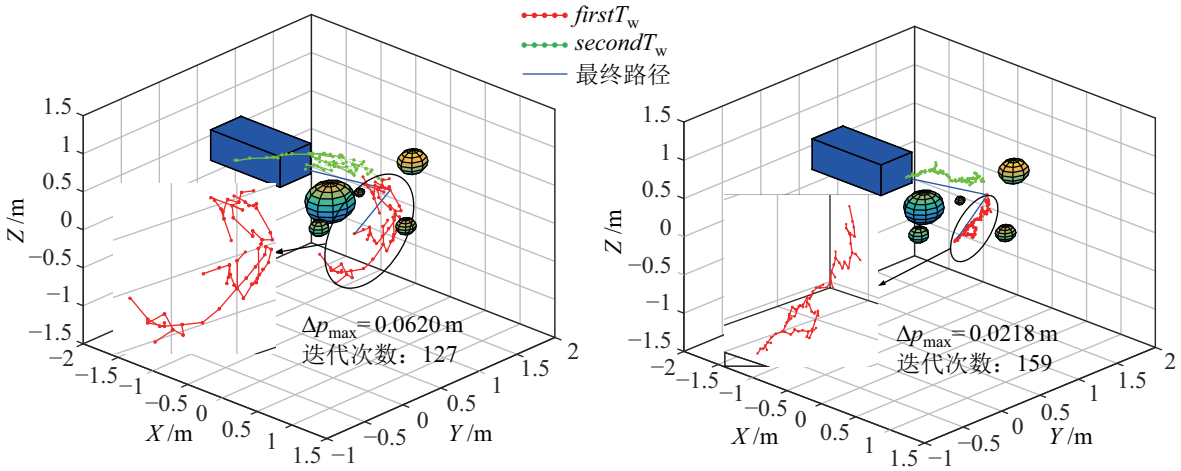
图 11(a) 中，Bi-RRT 的 $\Delta p_{\max}$ 为 0.0092 m，虽然小于 $\|\Delta \boldsymbol{p}_{\max}\|_W$，但过小的步长导致迭代次数的急剧上升，算法生成过多的无用节点，这些节点并没有起到促进随机树 *firstT*$_w$ 和 *secondT*$_w$ 融合的作用，且占用大量内存，导致算法运行速度下降．图 11(b)



(a) 步长为 0.5 rad 的 Bi-RRT　　(b) 引力自适应步长 RRT

图 9　引力自适应步长 RRT 与步长为 0.5 rad 的 Bi-RRT 的路径规划

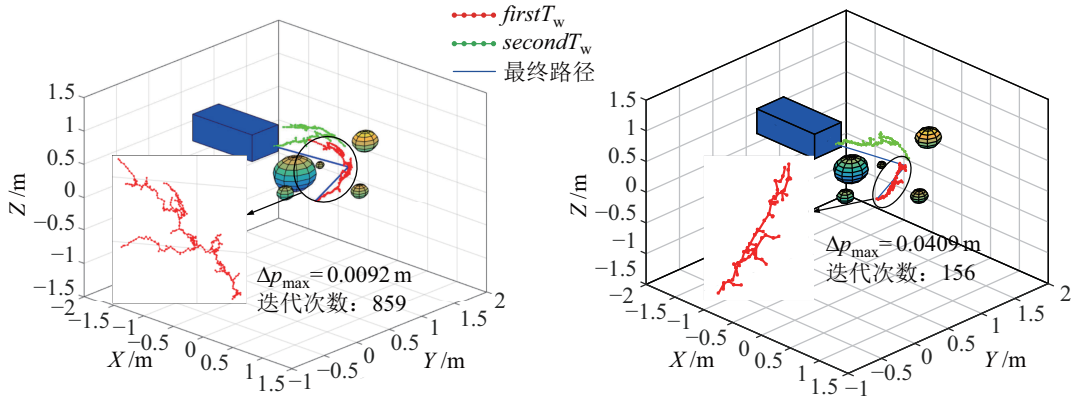
Fig.9　Path planning by attractive force self-adaptive step size RRT and Bi-RRT with 0.5 rad fixed step size



(a) 步长为 0.3 rad 的 Bi-RRT　　(b) 引力自适应步长 RRT

图 10　引力自适应步长 RRT 与步长为 0.3 rad 的 Bi-RRT 的路径规划

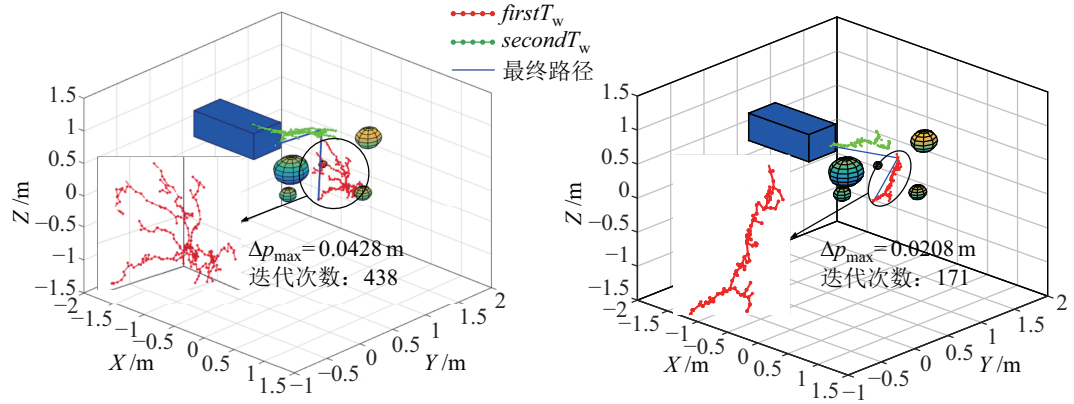
Fig.10　Path planning by attractive force self-adaptive step size RRT and Bi-RRT with 0.3 rad fixed step size



(a) 步长为 0.1 rad 的 Bi-RRT (b) 引力自适应步长 RRT

图 11 引力自适应步长 RRT 与步长为 0.1 rad 的 Bi-RRT 的路径规划

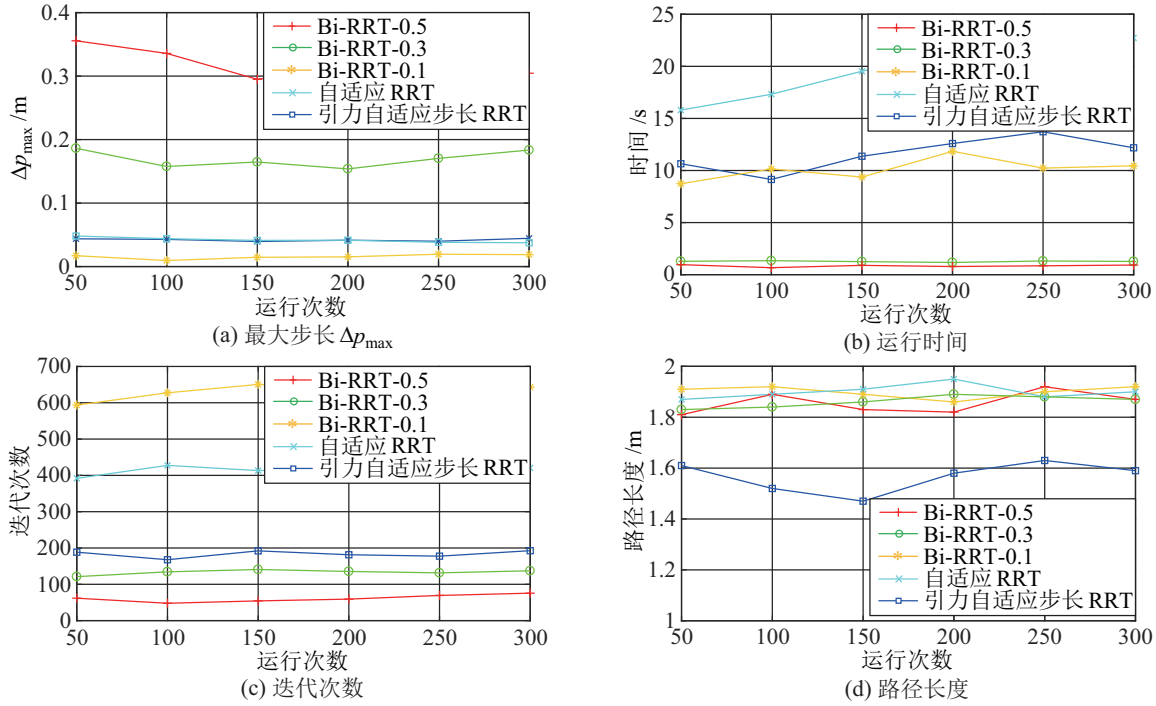
Fig.11 Path planning by attractive force self-adaptive step size RRT and Bi-RRT with 0.1 rad fixed step size



(a) 自适应 RRT (b) 引力自适应步长 RRT

图 12 引力自适应步长 RRT 与自适应 RRT 的路径规划

Fig.12 Path planning by attractive force self-adaptive step size RRT and adaptive RRT



(a) 最大步长 Δp_{max} (b) 运行时间

(c) 迭代次数 (d) 路径长度

图 13 不同算法多次运行下的参数对比

Fig.13 The parameters comparison of different algorithms in multiple trials

中，引力自适应步长 RRT 在工作空间中步长依然稳定，迭代次数远低于步长为 0.1 rad 的 Bi-RRT，同时兼顾了随机树的生长速度与迭代次数．最后将引力自适应步长 RRT 同自适应 RRT 方法进行路径搜索，其结果如图 12 所示．

不难发现 2 种算法在工作空间中的步长相似，都可保证碰撞检测的有效性．图 12(a) 中，自适应 RRT 算法中随机树的生长相对发散，并产生了许多的“枝叶”，迭代次数也相对较高；而引力自适应步长 RRT 算法由于有引力函数的作用，随机树 $firstT_w$ 和 $secondT_w$ 始终朝着彼此生长，减少了无用节点的生成，提高了随机树的融合速度．

5.2　多次仿真实验

在同样的仿真模型下，利用上述 5 个算法进行 300 次路径搜索，算法的关键参数如图 13 所示．

对于固定步长的 Bi-RRT 算法，较大的步长可以缩短运行时间和减少迭代次数，但与此同时也会增大机器人在工作空间中的位移，一旦 Δp_{max} 大于最小障碍物的尺寸，算法将无法确保生成无碰撞路径．较小的步长虽然可以确保碰撞检测的有效性，但同时也会导致迭代次数和运行时间的增加，如何选取一个合适的步长通常需要进行多次程序调试．自适应 RRT 算法和引力自适应步长 RRT 算法在步长方面都可确保碰撞检测的有效性，但引力自适应步长 RRT 算法在迭代次数和运行时间上具有明显的优势，分别降低了 50% 和 30%．在路径长度方面，固定步长 Bi-RRT 算法和自适应 RRT 算法平均约为 1.9 m，引力自适应步长 RRT 算法平均约为 1.6 m，路径长度缩短了 16%．

5.3　实验验证

建立双臂机器人协同搬运系统的物理平台，如图 14 所示．该平台包括 2 台遨博 i5 机器人、2 套气动抓手、总线式控制器、关节驱动器．控制器采用灵思创奇 Links-box-03 型号，用来实现与关节驱动器的 EtherCAT 通信．在双臂机器人的工作空间中设置 3 个球形障碍物．基于本文提出的方法在 Matlab 中进行路径规划，并利用 3 次 B 样条完成轨迹规划．双臂机器人按照所规划的轨迹进行运动，其运动过程如图 15 所示．双臂机器人在运动过程中其末端执行器和连杆均未与障碍物发生碰撞，并始终保持着协同运动，验证了算法的可行性．

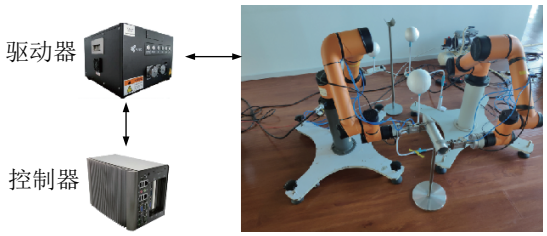


图 14　双臂机器人协同搬运实验平台

Fig.14　The cooperative transport experiment platform of the dual-arm robot

6　结论（Conclusion）

针对双臂机器人路径规划的特点，提出了引力自适应步长 RRT 方法，在传统的 RRT 算法基础上做出了如下改进：

(1) 构建了机器人构型空间和工作空间的范数不等式，把在工作空间中由随机树生长所引起的步长控制在指定范围内，确保了碰撞检测的有效性；

(2) 提出了随机树被动生长方法，在保持双臂机器人协同运动的前提下，降低了路径规划空间的维度，提高了算法的运行速度；

(3) 建立了节点的引力函数，减少了无用节点的生成，提高了随机树的融合速度．

仿真结果表明，该方法可在较短的时间内生成无碰撞的双臂机器人协同运动路径．和其他算法对比表明，在无碰撞的前提下，引力自适应步长 RRT

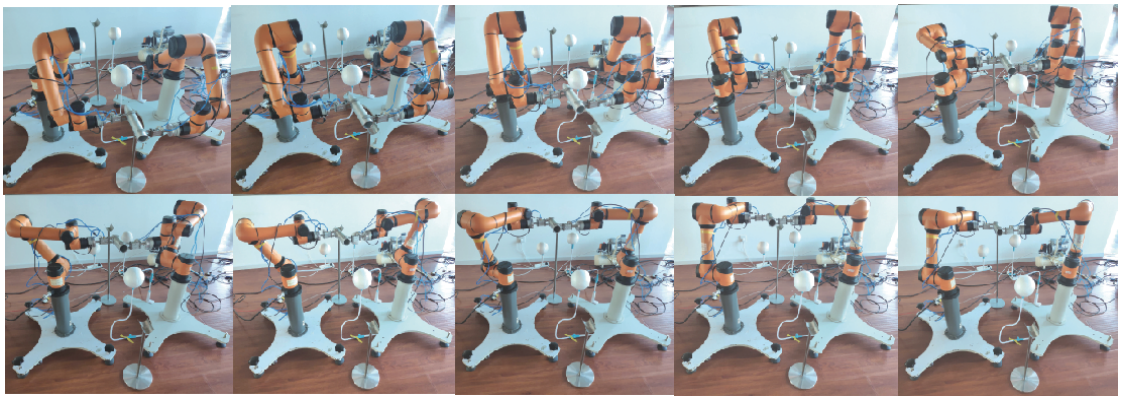


图 15　双臂机器人的运动过程

Fig.15　Motion process of the dual-arm robot

减少了迭代次数，降低了运行时间，并缩短了路径长度．实验表明该算法可实现双臂机器人的协同避障路径规划．在后续的研究工作中，将进一步优化步长以提高算法的搜索效率，并将其应用于非静态结构化环境下的路径规划．

参考文献（References）

[1] 欧阳帆．双机器人协调运动方法的研究 [D]．广州：华南理工大学，2013.
Ouyang F. Research on coordinated motion of dual robots[D]. Guangzhou: South China University of Technology, 2013.

[2] 张瑞星，李秀娟，高唤．双焊接机器人协同路径规划研究 [J]．组合机床与自动化加工技术，2019(6)：81-85.
Zhang R X, Li X J, Gao H. Research on cooperative path planning of double welding robot[J]. Modular Machine Tool and Automatic Manufacturing Technique, 2019(6): 81-85.

[3] Peng Y C, Carabis D S, Wen J T. Collaborative manipulation with multiple dualarm robots under human guidance[J]. International Journal of Intelligent Robotics and Applications, 2018, 2(2): 252-266.

[4] 谢生良，刘祚时．双臂机器人工作空间的分析与仿真 [J]．机械传动，2018，42(6)：139-143.
Xie S L, Liu Z S. Analysis and simulation of workspace of dual-arm robot[J]. Journal of Mechanical Transmission, 2018, 42(6): 139-143.

[5] Andreas V, Knut G. An optimization-based approach to dual-arm motion planning with closed kinematics[C]//IEEE/RSJ International Conference on Intelligent Robots and Systems. Piscataway, USA: IEEE, 2018: 8346-8351.

[6] Sanchez G, Latombe J C. Using a PRM planner to compare centralized and decoupled planning for multi-robot systems [C]//IEEE International Conference on Robotics and Automation. Piscataway, USA: IEEE, 2002: 2112-2119.

[7] Lim S J, Han C S. Operational space path planning of the dual-arm robot for the assembly task[J]. International Journal of Precision Engineering and Manufacturing, 2014, 15(10): 2071-2075.

[8] Steven B, Wasif N, Stuart F. Improved APF strategies for dual-arm local motion planning[J]. Transactions of the Institute of Measurement and Control, 2015, 37(1): 73-90.

[9] LaValle S M. Rapidly-exploring random trees: A new tool for path planning [R]. Ames, USA: Computer Science Department, Iowa State University, 1998.

[10] Karaman S, Frazzoli E. Sampling-based algorithms for optimal motion planning [J]. International Journal of Robotics Research, 2011, 30(7): 846-894.

[11] Kuffner J, LaValle S M. RRT-connect: An efficient approach to single-query path planning[C]//IEEE International Conference on Robotics and Automation. Piscataway, USA: IEEE, 2000: 995-1001.

[12] 王坤，黄勃，曾国辉，等．基于改进 RRT-Connect 的快速路径规划算法 [J]．武汉大学学报，2019，65(3)：283-289.
Wang K, Huang B, Zeng G H, et al. Faster path planning based on improved RRT-Connect algorithm[J]. Journal of Wuhan University, 2019, 65(3): 283-289.

[13] 莫栋成，刘国栋．改进的 RRT-connect 双足机器人路径规划算法 [J]．计算机应用，2013，33(8)：2289-2292.
Mo D C, Liu G D. Improved RRT-connect path planning algorithm for biped robot[J]. Journal of Computer Applications, 2013, 33(8): 2289-2292.

[14] 王维，李焱．基于 RRT 的虚拟人双臂操控规划方法 [J]．系统仿真学报，2009，21(20)：6515-6518.
Wang W, Li Y. RRT-based manipulation planning method for both arms of virtual human [J]. Journal of System Simulation, 2009, 21(20): 6515-6518.

[15] 杜爽，尚伟伟，刘坤，等．基于双向 RRT 算法的仿人机器人抓取操作 [J]．中国科学技术大学学报，2016，46(1): 12-20.
Du S, Shang W W, Liu K, et al. Bidirectional RRT algorithm based grasping manipulation of humanoid robots [J]. Journal of University of Science and Technology of China, 2016, 46(1): 12-20.

[16] Kim D H, Lim S J, Lee D H, et al. A RRT-based motion planning of dual-arm robot for (Dis) assembly tasks[C]//IEEE Conference on Intelligence and Safety for Robotics. Piscataway, USA: IEEE, 2013: 6pp.

[17] Chen P F, Zhao H, Zhao X, et al. Dimensionality reduction for motion planning of dual-arm robots[C]//IEEE International Conference on Mechatronics and Automation. Piscataway, USA: IEEE, 2018: 718-723.

[18] 刘成菊，韩俊强，安康．基于改进 RRT 算法的 RoboCup 机器人动态路径规划 [J]．机器人，2017，39(1)：8-15.
Liu C J, Han J Q, An K. Dynamic path planning based on an improved RRT algorithm for RoboCup robot[J]. Robot, 2017, 39(1): 8-15.

[19] Hao J, Zhang Y K, Wang J Z, et al. Path planning of industrial robot based on improved RRT algorithm in complex environments[J]. IEEE Access, 2018(6): 53296-53306.

[20] 申浩宇，吴洪涛，陈柏，等．冗余度双臂机器人协调避障算法 [J]．农业机械学报，2015，46(9)：356-361.
Shen H Y, Wu H T, Chen B, et al. Obstacle avoidance algorithm for coordinated motion of redundant dual-arm robot[J]. Transactions of the Chinese Society of Agricultural Machinery. 2015, 46(9): 356-361.

[21] Wang H X, Li R F, Gao Y F, et al. Comparative study on the redundancy of mobile single and dual-arm robots [J]. International Journal of Advanced Robotic Systems, 2016, 13(6): 1-19.

[22] 李洋，徐达，周诚．基于自适应步长 RRT 的双机器人协同路径规划 [J]．农业机械学报，2019，50(3)：358-367.
Li Y, Xu D, Zhou C. Cooperation path planning of dual-robot based on self-adaptive stepsize RRT[J]. Transactions of the Chinese Society for Agricultural Machinery. 2019, 50(3): 358-367.

[23] Zang X Z, Yu W T, Zhang L, et al. Path planning based on Bi-RRT algorithm for redundant manipulator[C]//International Conference on Electrical, Automation and Mechanical Engineering. Paris, France: Atlantis-Press, 2015: 189-191.

[24] An B, Kim J, Park C. An adaptive step size RRT planning algorithm for open-chain robots[J]. IEEE Robotics and Automation Letters, 2018, 3(1): 312-319.

作者简介：

李 洋（1990–），男，博士生．研究领域：多机器人运动规划，智能控制算法．

徐 达（1969–），男，教授，博士生导师．研究领域：智能武器技术，机器人智能控制技术．