doi:10.6041/j.issn.1000-1298.2019.03.041

基于自适应步长 RRT 的双机器人协同路径规划

李　洋　徐　达　周　诚
(陆军装甲兵学院兵器与控制系, 北京 100072)

摘要: 针对快速随机扩展树(Rapidly exploring-random tree,RRT)方法的步长确定过分依赖于程序调试,需耗费大量时间,且固定步长存在碰撞检测失效问题,提出了自适应步长 RRT 方法。通过建立构型空间与工作空间的范数相容不等式,把工作空间中产生的步长约束在允许范围内,进而实现有效的碰撞检测;提出了随机树被动生长方法,实现了双机器人在各自构型空间中的协同路径规划。仿真对比结果表明,采用自适应步长 RRT 方法进行双机器人路径规划时,随机树的每一次生长所产生的位移不超过设定值,保证了碰撞检测的有效性,相比传统的固定步长 RRT,自适应步长 RRT 方法无需多次调试就能确定步长,提高了机器人的路径规划速度。

关键词: 双机器人; 快速随机扩展树; 协同路径规划; 自适应步长
中图分类号: TP242.6　　**文献标识码**: A　　**文章编号**: 1000-1298(2019)03-0358-10

Cooperation Path Planning of Dual-robot Based on Self-adaptive Stepsize RRT

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

Abstract: An appropriate stepsize is required to be set up when using rapidly exploring-random tree (RRT) to perform path planning of a robot, which needs user to proceed debugging the program and it's generally time-consuming, also a fixed stepsize in RRT always resulting in invalid collision-test. Aiming at solving the above problems existing in RRT, a self-adaptive stepsize RRT was proposed. The matrix operator norm induced from configuration space norm and work space norm was founded based on Jacobi matrix and the norm inequality of configuration space and work space was established, by the means of which the displacement of robot caused by each stepsize in configuration space was limited in allowed magnitude which validated collision test. In order to coordinate dual-robot, passive growing of random tree algorithm was put forward. The algorithm can control the growth of random tree of dual-robot in different configuration spaces, and then the motion of dual-robot was coordinated to ensure generating cooperation path in work space. Numerical experiment indicated that the self-adaptive stepsize RRT can bound the displacement of each step within the value set up at beginning of algorithm which guaranteed the effectiveness of collision test. Compared with standard fixed stepsize RRT, self-adaptive stepsize RRT omitted the process of determining stepsize only needed to set maximum value of stepsize in work space which improved the efficiency of path planning. The algorithm proposed can provide a new perspective on the path planning of dual-arm robot.

Key words: dual-robot; rapidly-exploring random tree; collaborative path planning; self-adaptive stepsize

0　引言

RRT 作为具有概率完备性的规划方法[1-4]广泛应用于机器人的路径规划。该方法省略了在 C-space(构型空间)中建立障碍物模型的过程,只需在节点处进行碰撞检测,其运行速度快,对于高维空间

收稿日期: 2018-09-12　修回日期: 2018-11-05
基金项目: 国防预先研究项目(41404060201)
作者简介: 李洋(1990—),男,博士生,主要从事机器人运动控制研究,E-mail: 18510798818@163.com
通信作者: 徐达(1969—),男,教授,博士生导师,主要从事弹药装填机器人技术研究,E-mail: zxyxd@sina.com

的路径规划优势更加明显。目前，国内外学者针对 RRT 提出了多种改进方法，LIN 等[5]提出了一种自适应步长 RRT，该方法可根据障碍物自动调整随机树的生长方向，可快速生成避障路径。WANG 等[6]提出了变步长 RRT 方法，在随机树的生长过程中，加入了贪婪思想，使随机树以固定步长增量尽可能生长，加快了 RRT 算法的收敛速度，但无法控制在 W－space（工作空间）中的步长大小。文献[7－10]提出了自适应维度 RRT，通过降低 C－space 的局部维度，提高算法的运行速度，但该方法没有考虑节点之间的碰撞问题，无法保证碰撞检测的有效性。刘成菊等[11]将势函数引入到 RRT 算法中，减少了 RRT 算法的随机性，在局部搜索过程中具有良好的避障效果。

目前，RRT 及其改进方法的相关研究主要集中在移动机器人的路径规划，在双机器人的协同路径规划方面，RRT 相关应用较少。对于高维空间的运动规划，RRT 方法虽然省去了在 C－space 中建立障碍物模型的过程，但只在节点处进行碰撞检测难以保证其有效性[12－15]；利用 RRT 进行双机器人协同路径规划时，在每个节点处两个机器人末端执行器的相对位置必须保持固定[16－17]，而双机器人的随机树却在各自的 C－space 中随机生长，若要保持双机器人之间的协调，需在不同的 C－space 中对机器人随机树的生长进行协调控制。

本文提出一种自适应步长 RRT 方法，通过构建 C－space 和 W－space 中步长的范数不等式，把随机树每一次生长产生的位移约束在给定范围内，确保碰撞检测的有效性，通过被动生长算法控制两个机器人随机树的协调生长，完成双机器人在 W－space 中的协调运动，以实现双机器人协同路径规划。

1　双机器人运动学模型建立与求解

1.1　双机器人旋量模型

建立的双机器人模型如图 1 所示，双机器人系统由机器人 R1、R2 组成，其基座坐标系分别为 S_1 和 S_2，取 S_1 为惯性坐标系，S_2 在惯性坐标系下的描述为

$$ {}^{1}\boldsymbol{S}_2 = \boldsymbol{R}_z(\pi)\boldsymbol{T}_x(l_x)\boldsymbol{T}_y(l_y)\boldsymbol{S}_2 \tag{1} $$

其中

$$ \boldsymbol{R}_z(\pi) = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ \boldsymbol{T}_x(l_x) = \begin{bmatrix} 1 & 0 & l_x & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{T}_y(l_y) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & l_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

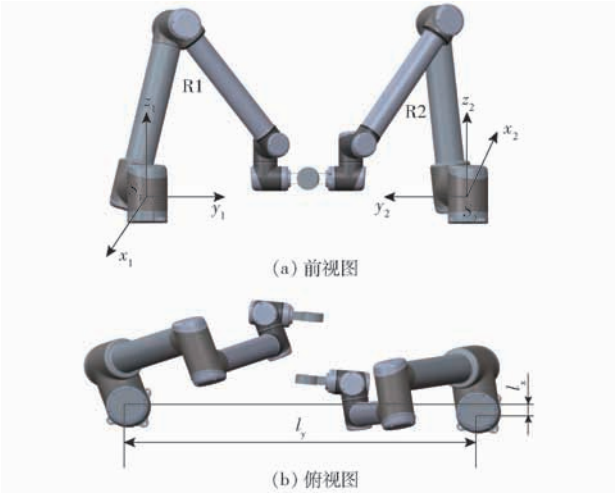

图 1　双机器人模型

Fig. 1　Models of dual-robot

通过 ${}^{1}\boldsymbol{S}_2$ 可以将机器人 R1 和 R2 的运动学统一在惯性坐标系 S_1 中表达，故只需在机器人 R1、R2 各自的基座坐标系下建立运动学模型即可。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，T 为工具坐标系，关节 1 和 5 绕 z 轴方向旋转，其余关节绕 y 轴方向旋转，r_i 为各关节轴线所在位置，建立关节的运动旋量坐标

$$ \begin{cases} \boldsymbol{\xi}_1 = [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0] \\ \boldsymbol{\xi}_2 = [0 \quad 1 \quad 0 \quad -l_1 \quad 0 \quad 0] \\ \boldsymbol{\xi}_3 = [0 \quad 1 \quad 0 \quad -(l_1+l_2) \quad 0 \quad 0] \\ \boldsymbol{\xi}_4 = [0 \quad 1 \quad 0 \quad -(l_1+l_2+l_3) \quad 0 \quad 0] \\ \boldsymbol{\xi}_5 = [0 \quad 0 \quad 1 \quad -l_5 \quad 0 \quad 0] \\ \boldsymbol{\xi}_6 = [0 \quad 1 \quad 0 \quad -(l_1+l_2+l_3+l_4) \quad 0 \quad 0] \end{cases} \tag{2} $$

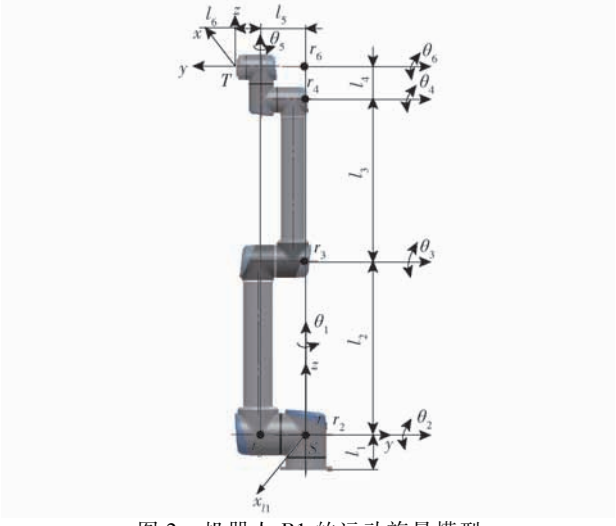

图 2　机器人 R1 的运动旋量模型

Fig. 2　Twist model of robot R1

将关节的旋量坐标映射为指数形式

$$e^{\xi_1\theta_1}=\begin{bmatrix}\cos\theta_1 & \sin\theta_1 & 0 & 0\\ \sin\theta_1 & -\cos\theta_1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$e^{\xi_2\theta_2}=\begin{bmatrix}\cos\theta_2 & 0 & \sin\theta_2 & -l_1\sin\theta_2\\ 0 & 1 & 0 & 0\\ -\sin\theta_2 & 0 & \cos\theta_2 & l_1(1-\cos\theta_2)\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$e^{\xi_3\theta_3}=\begin{bmatrix}\cos\theta_3 & 0 & \sin\theta_3 & -(l_1+l_2)\sin\theta_3\\ 0 & 1 & 0 & 0\\ -\sin\theta_3 & 0 & \cos\theta_3 & (l_1+l_2)(1-\cos\theta_3)\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$e^{\xi_4\theta_4}=\begin{bmatrix}\cos\theta_4 & 0 & \sin\theta_4 & -(l_1+l_2+l_3)\sin\theta_4\\ 0 & 1 & 0 & 0\\ -\sin\theta_4 & 0 & \cos\theta_4 & (l_1+l_2+l_3)(1-\cos\theta_4)\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$e^{\xi_5\theta_5}=\begin{bmatrix}\cos\theta_5 & -\sin\theta_5 & 0 & -l_5\sin\theta_5\\ \sin\theta_5 & \cos\theta_5 & 0 & 0\\ 0 & 0 & 1 & l_5(1-\cos\theta_5)\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$e^{\xi_6\theta_6}=$$

$$\begin{bmatrix}\cos\theta_6 & 0 & \sin\theta_6 & -(l_1+l_2+l_3+l_4)\sin\theta_6\\ 0 & 1 & 0 & 0\\ -\sin\theta_6 & 0 & \cos\theta_6 & (l_1+l_2+l_3+l_4)(1-\cos\theta_6)\\ 0 & 0 & 0 & 1\end{bmatrix}$$

1.2 正向运动学

根据串联机器人的 POE 公式[18]，计算 R1 的正向运动学方程

$$\boldsymbol{g}_1(\theta)=e^{\hat{\xi}_1\theta_1}e^{\hat{\xi}_2\theta_2}\cdots e^{\hat{\xi}_6\theta_6}\boldsymbol{T}=\begin{bmatrix}r_{11} & r_{12} & r_{13} & p_1\\ r_{21} & r_{22} & r_{23} & p_2\\ r_{31} & r_{32} & r_{33} & p_3\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3}$$

其中
$$r_{11}=-c_6(s_1s_5-c_1c_5c_{234}-c_1s_6s_{234})$$
$$r_{21}=-c_6(c_1s_5-s_1c_5c_{234}-s_1s_6s_{234})$$
$$r_{31}=-s_6c_{234}-c_5c_6s_{234}$$
$$r_{12}=-c_5s_1-c_1s_5c_{234}\quad r_{22}=c_1c_5-s_1c_5c_{234}$$
$$r_{32}=s_5s_{234}\quad r_{13}=c_1c_6s_{234}-s_6(s_1s_5-c_1c_5c_{234})$$
$$r_{23}=c_6s_1s_{234}-s_6(c_1s_5+s_1c_5c_{234})$$
$$r_{33}=c_6c_{234}-c_5s_6s_{234}$$
$$p_1=l_5s_1+l_2c_1s_2+l_3(c_1c_2c_3+c_1s_2c_3)+l_4(c_1c_2c_3s_4+c_1c_2s_3c_4+c_1s_2c_3c_4-c_1s_2s_3s_4)$$
$$p_2=-l_5c_1+l_2s_1s_2+l_3(s_1c_2s_3+s_1s_2c_3)+l_4(s_1c_2c_3s_4+s_1c_2s_3c_4+s_1s_2c_3c_4-s_1s_2s_3s_4)$$
$$p_3=l_1+l_2c_2+l_3c_{23}+l_4c_{234}$$

式中 $c_i=\cos\theta_i$，$s_i=\sin\theta_i$，$c_{ij}=\cos(\theta_i+\theta_j)$，$s_{ij}=\sin(\theta_i+\theta_j)$，下文同。

为实现双机器人协同路径规划，须将机器人 R1 和 R2 的正向运动学统一在同一个坐标系下描述，由于机器人 R1 和 R2 具有相同的结构，所以机器人 R2 的正向运动学 $\boldsymbol{g}_2(\theta)$ 在 S_2 中描述与 $\boldsymbol{g}_1(\theta)$ 相同，通过式(1)可将 $\boldsymbol{g}_2(\theta)$ 在惯性坐标系 S 下表述为

$$\boldsymbol{g}_2(\theta)={}^1\boldsymbol{S}_2\boldsymbol{g}_1(\theta) \tag{4}$$

1.3 逆向运动学

机器人 R1 和 R2 后 3 个关节的轴线不相交于一点，因此无法求解逆向运动学的解析解[19]。本文采用 Newton-Raphson 迭代法[20]，首先给出一组估计解 θ_0 和误差 Δ，通过多次迭代找到一组接近 θ_0 且误差小于 Δ 的数值解 θ_t，θ_0 的选取在下文中会进行详细的说明。

1.4 雅可比矩阵

雅可比矩阵定义了机器人单个关节速度对末端执行器速度的贡献度，将机器人 C-space 中的关节速度映射到 W-space 的矩阵函数，是构建 C-space 和 W-space 中步长关系的关键。机器人 R1 末端执行器速度与关节速度的映射关系为

$$\boldsymbol{V}=\boldsymbol{J}(\theta)\theta' \tag{5}$$

其中 $$\boldsymbol{J}(\theta)=[\boldsymbol{\xi}_1'\quad\boldsymbol{\xi}_2'\quad\boldsymbol{\xi}_3'\quad\boldsymbol{\xi}_4'\quad\boldsymbol{\xi}_5'\quad\boldsymbol{\xi}_6'] \tag{6}$$

式中 θ'——关节速度

$\boldsymbol{J}(\theta)$——雅可比矩阵

$\boldsymbol{\xi}_i'$——当前位型下第 i 个关节的运动副旋量

机器人 R1 的 6 个关节皆为转动副，因此有

$$\boldsymbol{\xi}_i'=\begin{bmatrix}\boldsymbol{\omega}_i'\\ \boldsymbol{r}_i'\times\boldsymbol{\omega}_i'\end{bmatrix}\quad(i=1,2,\cdots,6) \tag{7}$$

其中 $$\boldsymbol{\omega}_i'=e^{\theta_1\hat{\omega}_1}e^{\theta_2\hat{\omega}_2}\cdots e^{\theta_{i-1}\hat{\omega}_{i-1}}\boldsymbol{\omega}_i\in\mathbf{R}^{3\times1}$$

$$\boldsymbol{r}_i'=e^{\theta_1\hat{\xi}_1}e^{\theta_2\hat{\xi}_2}\cdots e^{\theta_{i-1}\hat{\xi}_{i-1}}\boldsymbol{r}_i(0)\in\mathbf{R}^{3\times1}$$

$$\hat{\boldsymbol{\omega}}_i=\begin{bmatrix}0 & -\omega_3 & \omega_2\\ \omega_3 & 0 & -\omega_1\\ -\omega_2 & \omega_1 & 0\end{bmatrix}\quad\hat{\boldsymbol{\xi}}_i=\begin{bmatrix}\hat{\omega}_i & v_i\\ 0 & 0\end{bmatrix}$$

$$\boldsymbol{\omega}_1'=[0\quad0\quad1]^{\mathrm{T}}\quad\boldsymbol{\omega}_2'=\boldsymbol{\omega}_3'=\boldsymbol{\omega}_4'=[-s_1\quad c_1\quad1]^{\mathrm{T}}$$

$$\boldsymbol{\omega}_5'=[-s_{234}\quad0\quad c_{234}]^{\mathrm{T}}$$

$$\boldsymbol{\omega}_6'=[-c_5-c_{234}c_1s_5\quad c_1c_5-c_{234}s_1c_5\quad s_{234}s_5]^{\mathrm{T}}$$

$$\boldsymbol{r}_1'\times\boldsymbol{\omega}_1'=[0\quad0\quad0]^{\mathrm{T}}$$

$$\boldsymbol{r}_2'\times\boldsymbol{\omega}_2'=[-l_1c_1\quad-l_1s_1\quad0]^{\mathrm{T}}$$

$$\boldsymbol{r}_3'\times\boldsymbol{\omega}_3'=[-c_1(l_2c_2+l_1)\quad-s_1(l_2c_2+l_1)\quad s_2l_2]^{\mathrm{T}}$$

$$\boldsymbol{r}_4'\times\boldsymbol{\omega}_4'=\begin{bmatrix}-c_1(l_3c_{23}+l_2c_2+l_1)\\ -s_1(l_3c_{23}+l_2c_2+l_1)\\ l_2s_{23}+l_3s_2\end{bmatrix}$$

$$\boldsymbol{r}_5' \times \boldsymbol{\omega}_5' = \begin{bmatrix} -l_5 c_{234} c_1 & l_1 s_{234} - l_5 c_{234} s_1 & l_5 s_{234} c_1 \end{bmatrix}^{\mathrm{T}}$$

$$\boldsymbol{r}_6' \times \boldsymbol{\omega}_6' = \begin{bmatrix} s_{234} s_1 c_5 (l_3 s_{23} + l_2 s_2) - (c_1 c_5 - c_{234} s_1 s_5)(l_4 c_{234} + l_3 s_{23} + l_2 c_2 + l_1) \\ -s_{234} s_5 (l_2 c_1 s_2 + l_4 (c_{23} s_4 + c_{23} c_4) + l_3 (c_1 c_2 s_3 + c_1 c_3 s_2)) - (s_1 c_5 + s_5 c_1 c_{234})(l_4 c_{234} + l_3 c_{23} l_2 c_2 + l_1) \\ (c_1 c_5 + s_1 s_5 c_{234})(l_2 c_1 s_2 + l_4 (c_{23} s_4 + s_{23} c_4) + l_3 (c_1 c_2 s_3 + c_1 s_2 c_3) + s_1 (s_1 c_5 + c_1 c_5 c_{234})(l_3 s_{23} + l_2 s_2)) \end{bmatrix}$$

2　传统 RRT 算法

采用固定步长 RRT 方法对机器人路径进行规划，需要在 C－space 中确定步长，对于多自由度的关节型机器人，C－space 通常为 6 维，因此无法在C－space 中进行碰撞检测，碰撞检测只能在 W－space 中的节点处进行，虽然在 C－space 中 RRT 的步长为固定的 $\Delta\boldsymbol{\theta}$，但是通过正向运动学将 $\Delta\boldsymbol{\theta}$ 映射到 W－space 中所产生的位移 $\Delta\boldsymbol{p}$ 却无法固定。

如图 3 所示，在 C－space 中随机树以固定步长 $\Delta\boldsymbol{\theta}$ 从节点 1 生长到节点 2 所引起的位移 $\Delta\boldsymbol{p}$ 大于障碍物 W－obs 的尺寸，而在 W－space 中节点 1 和节点 2 并没有与障碍物发生碰撞，RRT 算法会通过这次碰撞检测，但在机器人从节点 1 向节点 2 运动的过程中却发生了碰撞，由此可见，利用固定步长 RRT 方法对多自由度关节型机器人进行路径规划，无法确保生成一条避障路径。

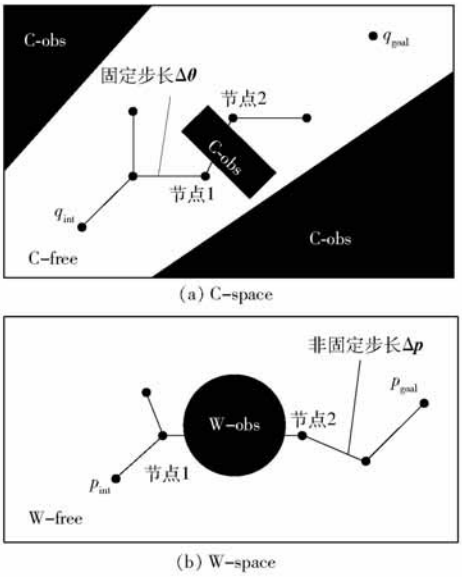

图 3　C－space 和 W－space 中的步长关系

Fig. 3　Relationship of stepsize in C－space and W－space

3　自适应步长 RRT

3.1　C－space 和 W－space 的范数不等式构建

RRT 方法以固定步长 $\Delta\boldsymbol{\theta}$ 在 C－space 中进行随机树的生长无法控制在 W－space 中所产生的位移 $\Delta\boldsymbol{p}$，为了解决这一问题引入算子范数，构建 C－space 和 W－space 中步长的范数不等式，实现在 C－space 中控制 W－space 中 $\Delta\boldsymbol{p}$ 的目的。给定两个向量范数 $\|\cdot\|_a \in \mathbf{R}^m$，$\|\cdot\|_b \in \mathbf{R}^n$ 和矩阵 $\boldsymbol{A} \in \mathbf{R}^{m\times n}$，从属于两个向量范数的矩阵算子范数为

$$\|\boldsymbol{A}\|_{a,b} = \max_{x\neq 0} \frac{\|\boldsymbol{Ax}\|_a}{\|\boldsymbol{x}\|_b} \tag{8}$$

可得到范数不等式

$$\|\boldsymbol{Ax}\|_a \leqslant \|\boldsymbol{A}\|_{a,b} \cdot \|\boldsymbol{x}\|_b \tag{9}$$

机器人 R1 的 C－space 中步长 $\Delta\boldsymbol{\theta} \in \mathbf{R}^6$，W－space 中所对应的位移 $\Delta\boldsymbol{p} \in \mathbf{R}^3$，因此只需构造从属于向量 $\Delta\boldsymbol{p}$ 和 $\Delta\boldsymbol{\theta}$ 的矩阵算子范数，就可建立 C－space和 W－space 的范数不等式，即构造一个矩阵 $\boldsymbol{A} \in \mathbf{R}^{3\times 6}$，满足 $\Delta\boldsymbol{p} = \boldsymbol{A}\Delta\boldsymbol{\theta}$ 即可通过不等式约束 $\Delta\boldsymbol{p}$ 的最大值，达到在 C－space 中控制步长的目的。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

$$\dot{\boldsymbol{P}} = \boldsymbol{\omega}_s \times \boldsymbol{P} + \boldsymbol{\nu}_s \tag{10}$$

其中　$\boldsymbol{\omega}_s = \begin{bmatrix} \boldsymbol{\omega}_1' & \boldsymbol{\omega}_2' & \cdots & \boldsymbol{\omega}_6' \end{bmatrix} \dot{\boldsymbol{\theta}}_i$

$$\boldsymbol{\nu}_s = \begin{bmatrix} \boldsymbol{r}_1' \times \boldsymbol{\omega}_1' & \boldsymbol{r}_2' \times \boldsymbol{\omega}_2' & \cdots & \boldsymbol{r}_6' \times \boldsymbol{\omega}_6' \end{bmatrix} \dot{\boldsymbol{\theta}}_i$$

根据雅可比矩阵式(10)可改写为

$$\dot{\boldsymbol{P}} = \sum_{i=1}^{6} (\boldsymbol{\omega}_i' \times \boldsymbol{P} + \boldsymbol{\nu}_i') \dot{\boldsymbol{\theta}}_i = \begin{bmatrix} \hat{\boldsymbol{\omega}}_1' \boldsymbol{p} + \boldsymbol{\nu}_1' \mid \cdots \mid \hat{\boldsymbol{\omega}}_6' \boldsymbol{p} + \boldsymbol{\nu}_6' \end{bmatrix} \Delta\dot{\boldsymbol{\theta}} \tag{11}$$

式(11)两边同乘以 Δt 可得

$$\Delta\boldsymbol{p} = \sum_{i=1}^{6} (\boldsymbol{\omega}_i' \times \boldsymbol{P} + \boldsymbol{\nu}_i') \dot{\boldsymbol{\theta}}_i = \begin{bmatrix} \hat{\boldsymbol{\omega}}_1' \boldsymbol{p} + \boldsymbol{\nu}_1' \mid \cdots \mid \hat{\boldsymbol{\omega}}_6' \boldsymbol{p} + \boldsymbol{\nu}_6' \end{bmatrix} \Delta\boldsymbol{\theta} \tag{12}$$

令矩阵 $\boldsymbol{A} = \begin{bmatrix} \hat{\boldsymbol{\omega}}_1' \boldsymbol{p} + \boldsymbol{\nu}_1' \mid \cdots \mid \hat{\boldsymbol{\omega}}_6' \boldsymbol{p} + \boldsymbol{\nu}_6' \end{bmatrix} \in \mathbf{R}^{3\times 6}$，可得到 C－space 和 W－space 范数不等式

$$\|\Delta\boldsymbol{p}\|_w \leqslant \|\boldsymbol{A}\|_{w,c} \|\Delta\boldsymbol{\theta}\|_c \tag{13}$$

式中　$\|\Delta\boldsymbol{p}\|_w$——W－space 中的范数

$\|\Delta\boldsymbol{\theta}\|_c$——C－space 中的范数

取 $w=2$，则 $\|\Delta\boldsymbol{p}\|_2 = \sqrt{\Delta p_x^2 + \Delta p_y^2 + \Delta p_z^2}$ 为 W－space 中的位移，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 $c=1$，则 $\|\boldsymbol{A}\|_{w,c} = \|\boldsymbol{A}\|_{2,1}$。当在 W－space 中给定了步长最大值 $\Delta\boldsymbol{p}_{\max}$，在 C－space 中每一个步长 $\Delta\boldsymbol{\theta}$ 都会自动调整以满足式(13)，所引起的位移 $\Delta\boldsymbol{p}$ 总是小于 $\Delta\boldsymbol{p}_{\max}$，实现了在 C－space 中控制步长的目的。

3.2　C－space 和 W－space 范数相容不等式改进

在式(13)中 $\Delta\boldsymbol{p}$ 特指双机器人末端执行器的位

移,当在 C-space 中给定一个步长 $\Delta\boldsymbol{\theta}$ 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,所以仅对机器人末端执行器的位移进行约束无法保证碰撞检测的有效性,因此需改进范数不等式。

选用长方体包围盒对双机器人的连杆进行模型等效,设第 i 个连杆的顶点集合 $\boldsymbol{L}_i=\{\boldsymbol{p}_{i1},\boldsymbol{p}_{i2},\cdots,\boldsymbol{p}_{i8}\}$,$i=1,2,\cdots,6$,则产生最大位移的位置必然在集合 $\boldsymbol{L}_i$ 中,所以有

$$\|\Delta\boldsymbol{p}_{\max}\|_w=\max\|\Delta\boldsymbol{p}_{ij}\|_w\quad(\Delta\boldsymbol{p}_{ij}\in\boldsymbol{L}_i)\tag{14}$$

将式(14)代入式(13)可得

$$\max\|\Delta\boldsymbol{p}_{ij}\|_w\leqslant\max\|\boldsymbol{A}_{ij}\|_{w,c}\|\Delta\boldsymbol{\theta}\|_c\tag{15}$$

其中 $\boldsymbol{A}=[[\boldsymbol{\omega}_1']\boldsymbol{p}_{ij}+\boldsymbol{\nu}_1'|\cdots|[\boldsymbol{\omega}_6']\boldsymbol{p}_{ij}+\boldsymbol{\nu}_6']\in\mathbf{R}^{24\times6}$

$$\max\|\boldsymbol{A}_{ij}\|_{w,c}\in\mathbf{R}^{3\times6}$$

由此得到改进 C-space 和 W-space 范数不等式为

$$\|\Delta\boldsymbol{p}_{ij}\|_w\leqslant\max\|\boldsymbol{A}_{ij}\|_{w,c}\|\Delta\boldsymbol{\theta}\|_c\tag{16}$$

式中 w、c 的取值参考 3.1 节。

通过式(16)可以控制机器人任意位置由 $\Delta\boldsymbol{\theta}$ 所引起的最大位移 $\Delta\boldsymbol{p}_{ij}$,完成双机器人的全局性避障。

3.3 随机树被动生长

为了保持机器人之间的协调,在 W-space 中,随机树每一个节点的机器人末端执行器相对位置必须保持不变,因此两个机器人在 W-space 中必须共享一颗随机树 S_tree,但是在 C-space 中,机器人 R1 和 R2 的随机树却是各自生长,故 RRT 算法无法控制双机器人随机树的生长产生相同的 S_tree。引入随机树被动生长方法,使机器人 R2 的随机树跟随机器人 R1 随机树的生长,在机器人 R2 的随机树生长的同时,完成对 S_tree 的复制,如图 4 所示,假设机器人 R1 的随机树为主动生长树 A_tree,机器人 R2 的随机树为被动生长树 P_tree,当 A_tree 在 C-space 中生长出一个新的节点 A_node 时,通过正向运动学将该节点映射到 W-space 中,完成共享随机树 S_tree 的生长并产生新的共享节点 S_node,而 P_tree 新节点 P_node 的正向运动学须满足

$$\boldsymbol{g}_1(\text{A\_node})={}^1\boldsymbol{S}_2\boldsymbol{g}_2(\text{P\_node})=\text{S\_node}\tag{17}$$

因此机器人 R2 随机树 P_tree 只能根据式(17)进行生长,通过机器人 R2 的逆向运动学反解出 P_node,为了避免机器人 R2 的关节位移产生跳动,估计解 θ_0 应取在 P_tree 中 A_node 的父节点 A_{near}_node 所对应的节点 P_{near}_node。此时 P_node 为 P_tree 的新节点,若 P_node 通过碰撞检测,则把 P_node 加入到机器人 R2 随机树 P_tree 中,完成一次双机器人随机树的生长。

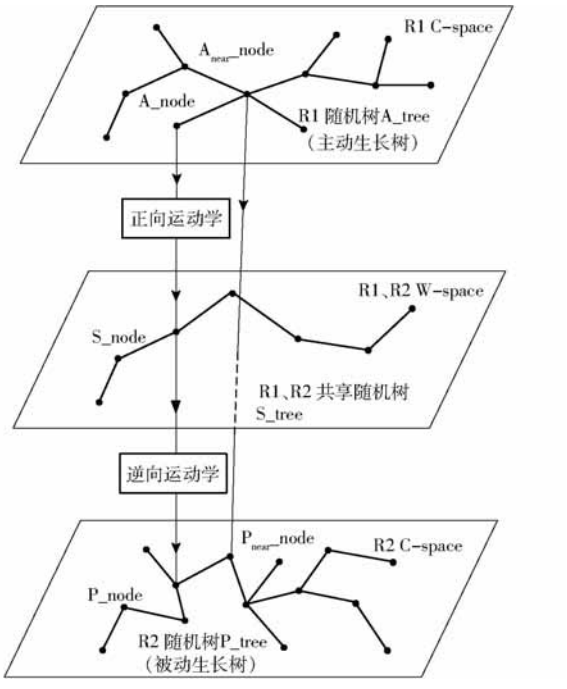

图 4 随机树被动生长

Fig. 4 Passive growing of random tree

3.4 双机器人自适应步长 RRT 算法

基于 3.2 节所建立的范数不等式和 3.3 节提出的随机树被动生长方法,建立双机器人自适应步长 RRT 算法,首先确定机器人在 W-space 中所允许的最大位移,令 $\max\|\boldsymbol{A}_{ij}\|_{w,c}\|\Delta\boldsymbol{\theta}\|_c=\Delta\boldsymbol{p}_{\max}$,代入式(16)可得

$$\|\Delta\boldsymbol{p}_{ij\max}\|_w\leqslant\max\|\boldsymbol{A}_{ij}\|_{w,c}\|\Delta\boldsymbol{\theta}\|_c=\Delta\boldsymbol{p}_{\max}\tag{18}$$

在构型空间中的步长为

$$\|\Delta\boldsymbol{\theta}\|_c=\frac{\|\Delta\boldsymbol{p}_{\max}\|_w}{\max\|\boldsymbol{A}_{ij}\|_{w,c}}\tag{19}$$

式中 $\|\Delta\boldsymbol{p}_{\max}\|_w$ 为算法的初始设定值,随机树每一次生长都会更新 $\|\boldsymbol{A}_{ij}\|_{w,c}$,从而使步长 $\Delta\boldsymbol{\theta}$ 自动满足式(19),此时 $\Delta\boldsymbol{\theta}$ 的取值即控制了机器人工作空间步长 $\Delta\boldsymbol{p}_{ij\max}$ 又可以保证随机树的生长速率。为了提高算法的搜索效率,采用双向搜索方法[21],双机器人自适应步长 RRT 算法的伪代码为

$\text{F\_JTree}^1=\theta^1_{\text{int}},\text{S\_JTree}^1=\theta^1_{\text{goal}},\Delta p_{\max}$

$\text{F\_JTree}^2=\theta^2_{\text{int}},\text{S\_JTree}^2=\theta^2_{\text{goal}}$

$\text{F\_WTree}=p_{\text{int}},\text{S\_WTree}=p_{\text{goal}}$

While $n<n_{\max}$ do

$\theta^1_{\text{rand}}=\text{sampling}()$

$\theta^1_{\text{near}}=\text{GetNearestPointOnTree}(\text{F\_JTree}^1,\theta^1_{\text{rand}})$

$d=\text{getDrection}(\theta^1_{\text{near}},\theta^1_{\text{rand}})$

$\|\Delta\theta\|_c=\text{Stepsize}(\Delta p_{\max},d,\theta^1_{\text{near}})$

$\theta^1_{\text{new}}=\text{Grow}(\text{F\_Tree}^1,\theta^1_{\text{near}},d,\|\Delta\theta\|_c)$

$p_{\text{new}}=\text{FK}(\theta^1_{\text{new}})$

θ^2_{new} = IK(p_{new}, θ^2_{idx})
if CollisionFreeAndCheckJointLimits (θ^1_{new}, θ^2_{new}, p_{max}) = True
Tree_Add(θ^1_{new}, F_JTree^1)
Tree_Add(θ^2_{new}, F_JTree^2)
Tree_Add(p_{new}, F_WTree)
if ConnectToOtherTree (θ^1_{new}, θ^2_{new}, F_JTree^1, S_JTree^1, Δp_{max}) = True
return Path_J(S_JTree, F_JTree)
return Path_W(S_WTree, F_WTree)
else
ExchangeTrees (S_JTree^1, S_JTree^2, F_JTree^1, F_JTree^2)
end if
end if
end while
Return Null

在 C－space 中分别定义以机器人 R1、R2 起始点和目标点为根的随机树 F_JTree^1、S_JTree^1、F_JTree^2、S_JTree^2，在 W－space 中定义机器人 R1、R2 的共享树 F_WTree、S_WTree，在对随机树 F_JTree^1 进行生长时，引入范数不等式，通过 Stepsize( ) 计算并且约束步长 $\Delta\boldsymbol{\theta}$，Stepsize( ) 的伪代码为

$$A = \max \| \boldsymbol{A}_{ij}(\theta) \|_{w,c}$$

$$\| \Delta\theta \|_c = \frac{\| \Delta p_{max} \|_w}{A} d$$

Return $\| \Delta\theta \|_c$

算法实质上为式(19)的计算过程，其中涉及到 w、c 的取值，前文已有 $w=2$，$c=1$，则 $\boldsymbol{A}_{ij}$ 的算子范数定义如下

$$\| \boldsymbol{A}_{ij} \|_{1,2} = \max \left(\sum_i |a_{ij}|^2 \right)^{1/2} \tag{20}$$

$\boldsymbol{A}_{ij}$ 的算子范数求解伪代码为

max = 0
J = Jacc(θ_{new})
for i = 1 to 6
for j = 1 to 8
for k = 1 to i
temp = $\| \omega_i \times p_{ij} + \nu_i \|_2$
if temp > max
max = temp
end if
end for
end for
end for
Return max

算法中的第 2 行是雅可比矩阵的求解，ω_i 和 ν_i 已经在前文中定义。

双机器人自适应步长 RRT 算法的 9～15 行为机器人 R2 随机树 F_JTree^2 被动生长的过程，当机器人 R1 的随机树 F_JTree^1 产生新的节点 θ^1_{new} 时，首先通过正向运动学在 W－space 中进行共享树 F_WTree 的生长产生新的节点 p_{new}，然后求解 p_{new} 的逆向运动学产生机器人 R2 随机树 F_JTree^2 的新节点 θ^2_{new}，若 θ^1_{new}、θ^2_{new} 和 p_{new} 通过关节限制检测和碰撞检测，则把新节点加入到各自的随机树中，完成 F_JTree^1、F_JTree^2 和 F_WTree 的一次生长。

算法第 16 行为树 F_JTree^1 和 S_JTree^1 融合的判别条件，由于在 C－space 中进行随机树的融合无法控制机器人在 W－space 中的运动轨迹，如图 5 所示，在 C－space 中随机树在节点 θ^1_{new} 和 θ^2_{new} 处进行融合时，以自适应步长的方式在方向 d 上进行融合，而在 W－space 中所对应的节点 p_{new} 和 p_{idx} 无法按期望的直线轨迹(图 5 中的虚线轨迹)融合，机器人在 p_{new} 和 p_{idx} 之间的轨迹是一条无法控制的曲线(图 5 中的实线轨迹)，故本文在 W－space 中完成随机树的融合，其伪代码为

flag = true
[θ^2_{near}, idx] = FindNearestOnS_JTree^1 (θ^1_{new}, S_JTree^1)
θ^2_0 = GetNodeOn F_JTree^2 (θ^1_{new}, F_JTree^2)
p_{idx} = GetpointOnF_WTree^1 (idx, F_WTree^1)
d = GetDirection(p_{idx}, p_{new})
dist = GetDistance(p_{idx}, p_{new})
R = GetPose(p_{new})
while flag = true
s = Stepsize(Δp_{max}, d, θ^1_{new})
temp = p_{new} + s
θ^1_{temp} = IK(*temp*, R, θ^1_{new})
θ^2_{temp} = IK(*temp*, R, θ^2_0)
if CollisionFreeAndCheckJointLimits
(θ^1_{temp}, θ^2_{temp}) = True
if Distance (*temp*, θ^1_{new}) < d
θ^1_{new} = θ^1_{temp}
θ^2_{new} = θ^2_{temp}
p_{new} = *temp*
continue
else
break
end if
else

```
        flag = false
    end if
end while
```

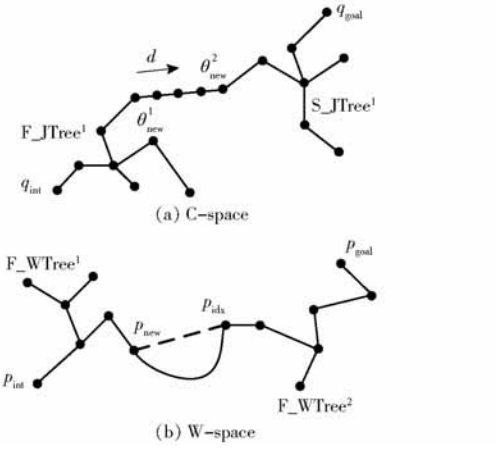

图 5 随机树的融合

Fig. 5 Merging of random tree

算法首先找出随机树 S_WTree¹ 中距离 p_{new1} 最近的点 p_{idx}，并确定随机树的融合方向 d，算法的 9 ~ 12 行是以自适应步长的方式，在 p_{new} 和 p_{idx} 之间插入 n 个节点，而后检测这些节点能否通过关节限制检测和碰撞检测，若通过检测则随机树 F_WTree、S_WTree 融合，机器人 R1 和 R2 在工作空间中走出以 p_{new1}、p_{idx} 为端点的直线轨迹；若其中有节点发生碰撞或者超出关节限制，则随机树 F_WTree、S_WTree 无法融合，继续进行随机树的生长。

当随机树 F_WTree、S_WTree 完成融合时，自适应步长 RRT 算法结束，得到双机器人的协同运动路径。

4 仿真

4.1 单次仿真

在 Matlab 中搭建双机器人模型，如图 6 所示，在双机器人的 W - space 中分别设置 4 个球形障碍物，半径分别为 0.1、0.1、0.15、0.2 m，和一个长方体障碍物，尺寸为 1.6 m × 0.4 m × 0.4 m，最小半径为 0.1 m，故设置 Δp_{max} 为 0.1 m，即机器人 R1 和 R2 在 W - space 中的步长不能超过 0.1 m，否则碰撞检测将会失效，无法保证机器人有效避障，起点的坐标为 (0.15,0.6,0)，终点的坐标为 (-0.55,0.6,0.6)。在模型中分别利用自适应步长 RRT 算法和 RRT 算法进行路径搜索并进行对比。

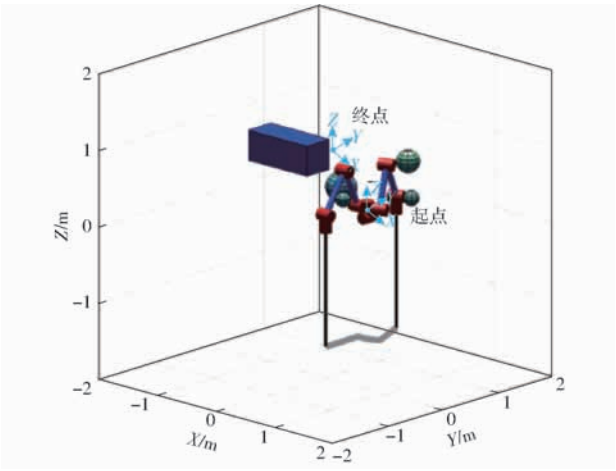

图 6 双机器人仿真环境

Fig. 6 Simulation environment of dual-robot

图 7 为自适应步长 RRT 算法和步长为 0.5 rad 的 RRT 算法进行一次路径搜索的结果，其中以起始点为根的随机树为 F_WTree，以目标点为根的随机树为 S_WTree，当两棵树达到融合条件时，随机树停止生长，经过圆滑处理产生一条距离最短的最终路径。由于 RRT 算法的步长为 0.5 rad，因此在工作空间中产生的步长也相对较大，所用的迭代次数较少，虽然 RRT 算法成功地生成了一条路径，但从图 7a 可发现，RRT 算法在工作空间中所产生的步长具有明显的跳动，在 W - space 中的步长最大值为 0.155 7 m，大于 Δp_{max}，无法保证碰撞检测的有效性。自适应步

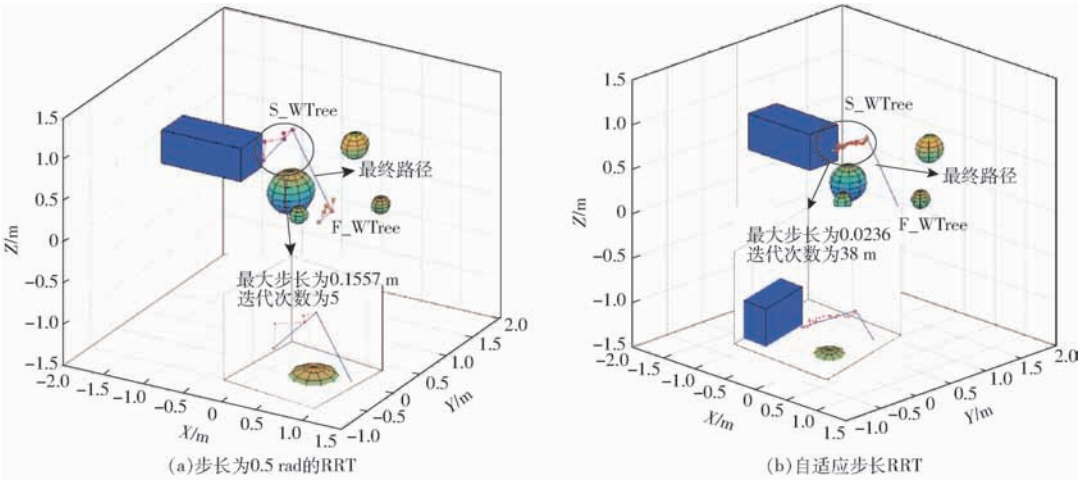

图 7 步长为 0.5 rad 的 RRT 与自适应步长 RRT 路径规划对比

Fig. 7 Comparison of path planning by RRT with stepsize of 0.5 rad and self-adaptive stepsize RRT

长 RRT 所产生的步长具有很强的稳定性，且工作空间中最大步长为 0.023 6 m，小于 $\Delta p_{\max}$，实现了双机器人的避障路径规划。

进一步减小 RRT 算法的步长并与自适应步长 RRT 算法进行对比，其结果如图 8 所示，图 8a 为步长为 0.3 rad 的 RRT 的路径搜索结果，相对于步长为 0.5 rad 的 RRT，步长为 0.3 rad 的 RRT 在W −space 中产生的步长平均值大幅减小，但仍然存在较大的步长跳动，步长最大值为 0.112 9 m，大于 $\Delta p_{\max}$，仍然无法保证碰撞检测的有效性，图 8b 为自适应步长 RRT 算法第 2 次的路径搜索结果，保持了在 W − space 中步长的稳定性，最大步长为 0.032 7 m，小于 $\Delta p_{\max}$。

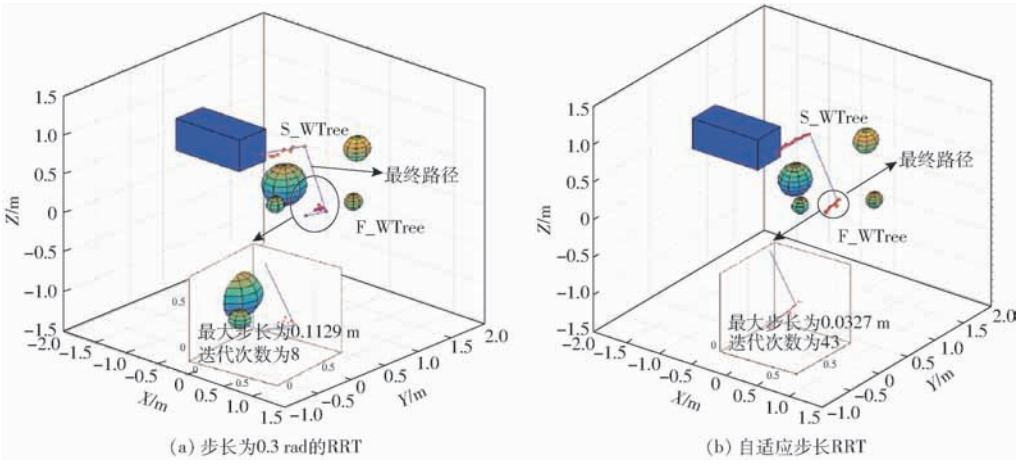

图 8　步长为 0.3 rad 的 RRT 与自适应步长 RRT 路径规划对比

Fig. 8　Comparison of path planning by RRT with stepsize of 0.3 rad and self-adaptive stepsize RRT

将 RRT 算法的步长缩小到 0.1 rad 并再次与自适应步长 RRT 算法进行对比，结果如图 9 所示，在 C − space 中 0.1 rad 是一个非常小的步长，意味着机器人 6 个关节所产生的角度变化的总和为 0.1 rad，因此随机树的每一次生长在 W − space 中所产生的步长会非常小，如图 9a 所示，步长最大值为 0.001 9 m，虽然远小于 $\Delta p_{\max}$，但过小的步长使随机树的生长速度变慢，导致迭代次数剧增，是步长为 0.3 rad 的 12 倍。在相对严格的融合条件下，两棵随机树难以融合，在可接受的时间内无法搜索出一条避障路径。自适应步长 RRT 算法的第 3 次路径搜索仍然保持了步长稳定的特性，且迭代次数约为步长 0.1 rad RRT 算法的 1/2，同时兼顾了随机树生长速度和步长大小的控制。

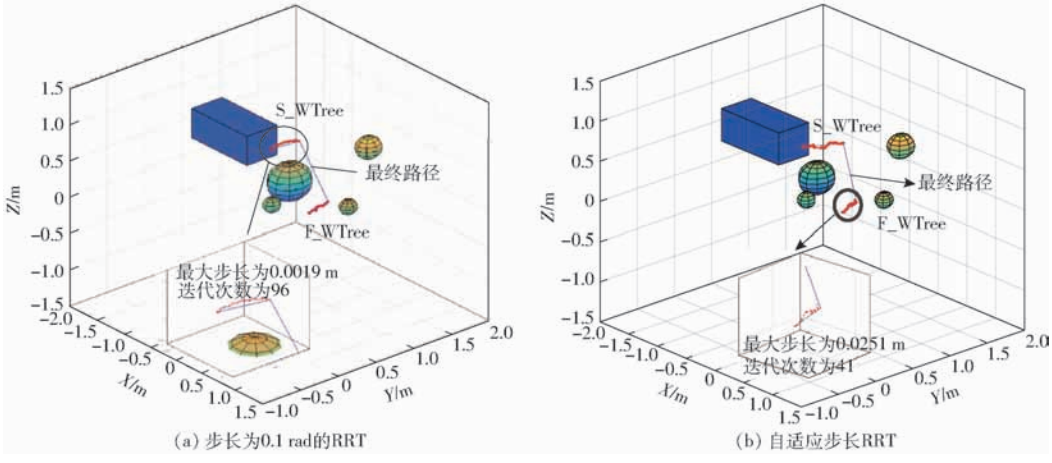

图 9　步长为 0.1 rad 的 RRT 与自适应步长 RRT 路径规划对比

Fig. 9　Comparison of path planning by RRT with stepsize of 0.1 rad and self-adaptive stepsize RRT

通过对 RRT 算法步长的多次调整，确定 0.2 rad 为 RRT 算法在当前仿真环境下的步长匹配值，进行一次路径搜索，其结果如图 10 所示。

算法的最大步长为 0.042 3 m，迭代次数为 30 次，基本与自适应步长 RRT 算法相近，步长仍有轻微跳动，最大步长已经控制在 $\Delta p_{\max}$ 的范围内，可以满足碰撞检测的要求。使用 RRT 算法进行路径搜索时，从上述的仿真过程可以发现，找到合适的步长通常需要多次调试才能够确定，而自适应步长 RRT 只需要指定 $\Delta p_{\max}$，就可自动调整 C − space 中的步长来控制 W − space 中产生步长，当机器人的工作环境发生变化时，只需根据环境模型指定新的 $\Delta p_{\max}$ 就

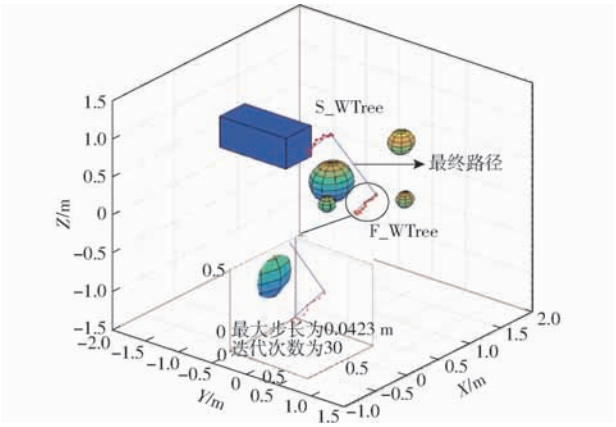

图 10 步长为 0.2 rad 的 RRT 路径规划

Fig. 10 Path planning by RRT with stepsize of 0.2 rad

可再次进行路径搜索,而固定步长 RRT 算法需再次对步长进行多次的调试。另外,传统的 RRT 算法存在步长过大所导致的碰撞检测失效问题,而自适应步长 RRT 算法,通过式(16)把机器人的关节、连杆和末端执行器的位移全部约束在 Δp_{max} 内,可得到机器人全局性的无碰撞路径。

4.2 多次仿真

分别对自适应步长 RRT 和步长 0.5、0.3、0.1、0.2 rad 的 RRT 进行 50 次仿真,其结果如表 1 所示。表中 p_{max} 为 W - space 中步长的最大值,当 p_{max} 大于 Δp_{max} 时,算法失效。从表 1 可以看出,RRT 算法随着步长的减小,p_{max} 的最大值和平均值也随之减小,但迭代次数会不断增加。当步长为 0.5 rad 时,RRT 算法的步长较大,失效次数为 37 次,失效概率为 74%,当进一步减小步长至 0.3 rad 时,虽然 p_{max} 的平均值降到可接受的范围内,但仍有 14 次失效,失效概率为 28%。当步长为 0.1 rad 时,失效次数虽然为 0,但过小的步长导致迭代次数的剧增,而自适应步长 RRT 既保证了算法的失效概率为 0,同时又控制了算法的迭代次数。当取 RRT 的步长为 0.2 rad 时,可得到与自适应步长 RRT 相接近的最大步长与迭代次数。

表 1 不同步长 RRT 与自适应步长 RRT 步长、迭代次数和失效次数

Tab. 1 Maximum stepsize, iterations and invalid times of RRT and self-adaptive stepsize RRT

	p_{max} 最大值/m	p_{max} 平均值/m	迭代次数平均值	失效次数
自适应步长 RRT	0.043 7	0.020 4	51.8	0
步长为 0.5 rad 的 RRT	0.285 6	0.113 8	11.3	37
步长为 0.3 rad 的 RRT	0.156 3	0.082 4	18.6	14
步长为 0.2 rad 的 RRT	0.088 1	0.043 1	30.2	0
步长为 0.1 rad 的 RRT	0.021 7	0.009 1	98.4	0

5 结束语

针对 RRT 算法无法控制双机器人在 W - space 中步长的问题,提出了自适应步长 RRT 算法。通过构建 C - space 和 W - space 的范数不等式,把随机树每一次生长所产生的机器人位移约束在给定范围内,在利用自适应步长 RRT 进行路径规划时,只需根据机器人的工作环境指定 Δp_{max},省去了 RRT 算法为确定步长而进行多次调试的耗时过程;提出了随机树被动生长方法,实现了双机器人末端执行器同位置不同构型的路径规划,结合自适应步长 RRT 算法完成了双机器人的协同路径规划。

参 考 文 献

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

[2] LAVALLE S M, KUFFNER J. RRT - Connect: an efficient approach to single-query path planning[C]// Proceedings of the 2000 IEEE International Conference on Robotics & Automation, 2000: 995 - 1001.

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

[4] GERAERTS R, OVERMARS M H. Creating high-quality paths for motion planning [J]. The International Journal of Robotics Research, 2007, 26(8): 845 - 863.

[5] LIN Na, ZHANG Yalun. An adaptive RRT based on dynamic step for UAVs route planning [C]// Proceedings of 2014 IEEE 5th Software Engineering and Service Science, 2014: 1111 - 1114.

[6] WANG C Q, MENG Q H. Variant step size RRT: an efficient path planner [C]// Proceedings of IEEE International Conference on Real-time Computable Robotics, Cambodia, 2016: 555 - 560.

[7] KIM D H, Y S C, KIM S H, et al. Adaptive rapidly-exploring random tree for efficient path planning of high-degree-of-freedom articulated robots [J]. Journal of Mechanic Engineering Science, 2015, 229(18): 3361 - 3367.

[8] ADIYATOV O, VAROL H A. Rapidly-exploring random tree based memory efficient motion planning [C]// Proceedings of IEEE

International Conference on Mechatronics and Automation,2013:354 – 359.

[9] GOCHEV K, SAFONOVA A, LIKHACHEVZ M. Planning with adaptive dimensionality for mobile manipulation [C] // IEEE International Conference on Robotics and Auto-oration,2012:2944 – 2951.

[10] OLZHAS A, HUSEYIN A V. A novel RRT * -based algorithm for motion planning in dynamic environments [C] // Proceedings of 2017 IEEE International Conference on Mechatronics and Automation,2017:1416 – 1421.

[11] 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 Chengju, HAN Junqiang, AN Kang. Dynamic path planning based on an improved RRT algorithm for RoboCup robot [J]. Robot,2017,39(1):8 – 15. (in Chinese)

[12] TOIT N E, BURDICK J W. Robot motion planning in dynamic uncertain environments [J]. IEEE Transactions on Robotics, 2012,28(1):101 – 115.

[13] 李东方,王超,邓宏彬,等. 基于人工势场和 RRT 算法的机器蛇水下三维避障算法 [J]. 兵工学报,2017,38(1):205 – 214.
LI Dongfang, WANG Chao, DENG Hongbin, et al. 3D intelligent obstacle avoidance algorithm based on artificial potential field method and rapidly-exploring random tree [J]. ACTA Armamentar,2017,38(1):205 – 214. (in Chinese)

[14] 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 Dongcheng, LIU Guodong. Improved RRT – Connect path planning algorithm for biped robot [J]. Journal of Computer Applications,2013,33(8):2289 – 2292. (in Chinese)

[15] 郝利波. 基于改进 RRT 与人工势场混合算法的足球机器人路径规划研究 [D]. 西安:西安科技大学,2011.
HAO Libo. Based on improved RRT with artificial potential field hybrid algorithm of robot soccer path planning research [D]. Xi'an:Xi'an University of Science and Technology,2011. (in Chinese)

[16] 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):1515 – 1518.
WANG Wei, LI Yan. RRT – based manipulation planning method for both arms of virtual human [J]. Journal of System Simulation,2009,21(20):1515 – 1518. (in Chinese)

[17] 申浩宇,吴洪涛,陈柏,等. 冗余度双臂机器人协调避障算法[J/OL]. 农业机械学报,2015,46(9):356 – 361.
SHEN Haoyu, WU Hongtao, CHEN Bai, et al. Obstacle avoidance algorithm for coordinated motion of redundant dual-arm robot [J/OL]. Transactions of the Chinese Society for Agricultural Machinery,2015,46(9):356 – 361. http://www.j-csam.org/jcsam/ch/reader/view_abstract.aspx? flag = 1&file_no = 20150952&journal_id = jcsam. DOI:10.6041/j.issn.1000-1298.2015.09.052. (in Chinese)

[18] 李国利,姬长英,顾宝兴,等. 多末端苹果采摘机器人机械手运动学分析与试验[J/OL]. 农业机械学报,2016,47(12):14 – 21.
LI Guoli, JI Changying, GU Baoxing, et al. Kinematics snalysis and experiment of apple harvesting robot manipulator with multiple end-effectors [J/OL]. Transactions of the Chinese Society for Agricultural Machinery,2016,47(12):14 – 21. http://www.j-csam.org/jcsam/ch/reader/view_abstract.aspx? flag = 1&file_no = 20161203&journal _id = jcsam. DOI:10.6041/j.issn.1000-1298.2016.12.003. (in Chinese)

[19] 韩兴国,宋小辉,殷鸣,等. 6R 焊接机器人逆解算法与焊接轨迹误差分析 [J/OL]. 农业机械学报,2017,48(8):384 – 390.
HAN Xingguo, SONG Xiaohui, YIN Ming, et al. Solution of inverse kinematics and welding trajectory error analysis for 6R welding robot [J/OL]. Transactions of the Chinese Society for Agricultural Machinery,2017,48(8):384 – 390. http://www.j-csam.org/jcsam/ch/reader/view_abstract.aspx? flag = 1&file_no = 20170846&journal_id = jcsam. DOI:10.6041/j.issn.1000-1298.2017.08.046. (in Chinese)

[20] KEVIN M L, FRANK C P. Modern robotics:mechanics, planning and control [M]. Cambridge Shire:Combridge University Press,2017.

[21] 杜爽,尚伟伟,刘坤. 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 Shuang, SHANG Weiwei, LIU Kun. Bidirectional RRT algorithm based grasping manipulation of humanoid robots [J]. Journal of University of Science and Technology of China,2016,46(1):12 – 20. (in Chinese)