山西大学学报(自然科学版)40(2):244－254,2017
Journal of Shanxi University(Nat. Sci. Ed.)

DOI:10.13451/j.cnki.shanxi.univ(nat.sci.).2017.02.009

基于改进 RRT* 的移动机器人运动规划算法

潘思宇,徐向荣*

(安徽工业大学 机械工程学院,安徽 马鞍山 243002)

摘　要:RRT*(快速搜索随机树)算法在以往研究中存在收敛速度慢、结果不稳定的缺点。针对此问题,文章在现有 RRT* 基础之上提出一种新型改进算法。该改进算法结合环境约束、车辆自身约束和运动学约束,舍弃原算法贪心思想并引入启发式采样节点插入算法,提高路径规划的速度和质量;接着对改进算法进行理论分析,证明算法具有概率完整性、渐近最优性,从理论上保证算法能快速收敛到最优路径。通过各种仿真环境的测试,验证改进算法的有效性、稳定性和正确性,也验证理论分析的正确性。

关键词:快速搜索随机树;路径规划;移动机器人;机器人操作系统

中图分类号:TP301　　**文献标志码**:A　　**文章编号**:0253-2395(2017)02-0244-11

Improved RRT*-Based Motion Planning Algorithm for Mobile Robot

PAN Siyu, XU Xiangrong*

(*School of Mechanical Engineering, Anhui University of Technology, Ma'anshan* 243002, *China*)

Abstract: RRT* (rapidly-exploring random tree) has the disadvantages of instability and slow convergence rate in previous studies. To solve this problem, a new RRT*-based algorithm, IB-RRT algorithm, is proposed. This algorithm combines the environmental constraints, the constraints of intelligent vehicle and kinematics constraints, discards the bias in the original algorithm and sample insertion heuristic algorithm is introduced to increase the planning speed and quality greatly. Moreover, through theoretical analysis of the improved algorithm, it is proved that the improved algorithm has probability completeness, asymptotic optimality, which ensures that the algorithm can quickly converge to the optimal path in theory. The validity, stability and correctness of this algorithm are verified by the simulation experiments and the correctness of the theoretical analysis is also vevified.

Key words: RRT(rapidly-exploring random tree); path planning; mobile robot; ROS

0　引言

近年来智能移动机器人在工业、农业、航空航天及空间探索等方面起到了重要的作用,因而成为学术界研究和关注的热点问题。移动机器人在任务空间自由活动的过程中,首先需要解决的问题,就是路径规划问题。路径规划是指移动机器人按照某些性质指标(如最短路径,时间最短或者代价最少等)的要求,规划出一条从起始点到目标点位置的最优或是次优的无碰撞路径,并指示移动机器人按照该路径行驶[1]。而解决这些问题需要考虑的因素有:任务空间的复杂性和不确定性,规划算法的有效性、最优性和实时性以及满足移

* 收稿日期:2016-11-22;修回日期:2016-12-22
基金项目:国家外国专家局高端外国专家项目(GDT20153400058)
作者简介:潘思宇(1992－),男,安徽阜阳人,硕士生,研究方向为机器人技术及应用,E-mail:1352674808@qq.com
* 通信作者:徐向荣(XU Xiangrong),E-mail:xuxr@ahut.edu.cn

动机器人本体的运动学和动力学特性。

在过去的一段时间里,为了解决这些问题,国内外学者进行了大量的研究,不断地进行探索,提出了很多路径规划的理论和方法。传统的路径规划算法如蚁群算法、遗传算法、人工势场算法等,这些算法在处理简单的规划问题有一定的优越性,但是在复杂环境下和高维空间中算法的复杂的会急剧增加,导致收敛时间长、求解困难[2]。此外,基于势场或启发函数的算法,如 A* 、D* 和人工势场法等,在处理规划问题时虽然能满足最优性和实时性的要求,但是因其并未考虑移动机器人本体的运动学和动力学限制,使得规划的路径不一定能被移动机器人所执行[3]。

其他改进算法需要在一个确定性空间中对障碍物进行确定的建模,在高维空间中容易引发维度灾难。为了解决高维规划问题,基于采样的算法因此被引入[4],相比于其他先进算法,其主要优势就是避免构建显式的任务空间,且其已被证明是有效解决路径规划问题的方案[5],其中,最著名的基于采样的算法有 PRM 和 RRT,但当存在未知障碍物时 PRM 算法的效率是极其低下的[6]。RRT 算法收敛速度快,同时能应用在存在未知障碍物的环境中[7],但是,RRT 算法中仍然有一些缺点[8-10]:(1)因其采用的是全局均匀随机策略,导致其路径不稳定,规划出的路径未必是最优路径,而且无谓地消耗大量资源,收敛速度缓慢;(2)最近节点选择算法在解决复杂问题时可能会导致算法失效;(3)因算法的随机性生成的路径过于粗糙,不便于移动机器人实施;(4)因算法缺少学习能力,在扩展过程中选择的节点可能会产生使路径发生碰撞导致扩展失败的情况,然而在后续的扩展过程中,当遇到同一区域选择随机目标点是,这些会发生碰撞的节点和处在障碍物区域无法实化的节点将再次被选取,产生重复而不必要的运算,影响整体算法效率。

由于这些不足的存在,国内外的学者对此展开了研究,许多 RRT 的改进算法被提出,以适应不同的应用场景。为了提高节点的扩展效率,2000 年 Kuffner 和 LaValle 提出了 RRT-connect[11],次年,他们提出了双向搜索树(Bidirectional-RRT),从起始点和目标点同时出发并行生成两棵 RRT,直至两棵树相遇,加速算法收敛[12]。C. Urmson 和 R. Simmons(2003)提出了一种启发式的偏向算法,该算法引导 RRT 树向目标区域生长,使得 RRT 算法能得到一个低时间成本的解决方案[13]。D. Ferguson 和 A. Stentz(2006)考虑多次运行 RRT 算法以逐步提高解决方案的质量,即使不能保证收敛到最优的解决方案,也能保证算法每次运行的结果是一个比较小的路径代价的路径[14]。Karaman 和 Frazzoli(2010)首次提出 RRT* 算法,用来改进由基本 RRT 算法产生的并非概率最优解的问题[15]。M. Jordan 和 A. Perez(2013)提出了 B-RRT* ,用一个轻微变异的贪心 RRT-connect 作为启发函数来连接两颗随机树[16]。A. H. Qureshi 和 S. Mumtaz(2014)提出了 TG-RRT* ,利用三角几何来选取节点,以降低得到最优解所需要的迭代次数,从而使得算法快速收敛[17]。Ahmed Hussain Qureshi, Yasar Ayaz(2015)提出了 IB-RRT* ,利用双向树的方法,通过智能样本插入的启发函数使得算法快速收敛到最优路径[5],C. Wouter Bac 和 Tim Roorda(2016)提出利用 RRT 算法在 ROS 平台上做移动机器人的路径规划研究,并应用于在密集障碍物环境中运动规划问题中[18]。

本文在这些问题的基础上提出了一个新的算法,采用智能随机抽样的方法以降低得到最优解所需要的迭代次数,从而使得算法快速收敛,同时引入了惩罚函数来记忆所选取过的节点以提高算法的效率,同时考虑移动机器人所受到的运动学和动力学限制,使得生成的轨迹可以直接被实际机器人所直接实现。通过仿真实验,证实了该算法的有效性、最优性和实时性。

1 移动机器人的非完整性约束模型

非完整性约束是指含有系统广义坐标导数且不可积分的约束[19],移动机器人是一个典型的非完整约束系统。而本文要研究的移动机器人模型,是一种典型的非完整性约束的系统。

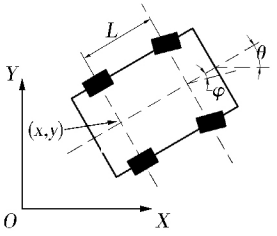


Fig. 1 Geometrics of mobile robot

图 1 移动机器人几何示意图

移动机器人在任务空间中状态变量为(x,y,θ,v,φ),其中(x,y)为移动机器人后轮轴的中心在系统坐标系下的坐标,θ 为移动机器人前进方向与 x 轴的夹角,φ 为移动机器人前轮方向与前进方向之间的夹角,v 为移动机器人的速度,由于受到非完整性约束,所以车轮与地面是点接触且在接触点只有纯滚动没有相对的滑动。由分析可知:

$$\begin{cases}\dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \dfrac{v\tan\varphi}{L}, \\ \dot{v} = u_0 \\ \dot{\varphi} = u_1 \end{cases} \tag{1}$$

从而可得移动机器人所受到的约束方程:

$$\begin{pmatrix} \dfrac{dx}{dt} \\ \dfrac{dy}{dt} \\ \dfrac{d\theta}{dt} \\ \dfrac{dv}{dt} \\ \dfrac{d\varphi}{dt} \end{pmatrix} = \begin{pmatrix} v\cos\theta \\ v\sin\theta \\ \dfrac{v\tan\varphi}{L} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} u_0 + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} u_1, \tag{2}$$

其中,移动机器人的控制变量为 u_0(加速度)和 u_1(车轮的角速度)。移动机器人的最小转弯半径(最大曲率):

$$K_{\max} = \frac{\tan\varphi_{\max}}{L} = \frac{1}{R_{\min}}. \tag{3}$$

除了上面的约束条件之外,还有:

$$\begin{cases} |u_0| \leqslant u_{0\max} \\ |u_1| \leqslant u_{1\max}, \\ |\varphi| \leqslant \varphi_{\max} \end{cases} \tag{4}$$

由此可知,没有考虑机器人的非完整性约束的规划算法,它们不考虑车辆约束,规划出来的路径也不能用于实际的路径规划当中[20]。

2 规划算法的描述与实现

2.1 RRT* 算法

RRT* 通过在新的节点附近建立周围节点分布图来改进现有搜索树,然后遍历周围的节点检查是否存在新的路径比现有的路径的代价更低,如果存在则将现有的路径换掉。通过这种方式 RRT* 算法可以保证最终所确认的路径是渐进最优解。下面是算法的基本流程。

ALGORITHM:RRT*

1. $V \leftarrow \{x_{init}\}; E \leftarrow \phi; T \leftarrow (V, E)$;
2. $for\ i \leftarrow 0\ to\ N\ do$
3. $x_{rand} \leftarrow Sample(i)$
4. $X_{near} \leftarrow NearVertices(x_{rand}, T)$
5. $if\ X_{near} = \phi\ then$
6. $X_{near} \leftarrow NearVertex(x_{rand}, T)$
7. $L_s \leftarrow GetSortedList(x_{rand}, X_{near})$
8. $x_{\min} \leftarrow ChooseBestParent(L_s)$
9. $if\ x_{\min} \neq \phi\ then$
10. $T \leftarrow InsertVertex(x_{rand}, x_{\min}, T)$
11. $T \leftarrow REWIREVERTICES(x_{rand}, L_s, E)$
12. $return\ T = (V, E)$

由 ALGORITHM:RRT* 所示,RRT* 算法以起始点 x_{init} 作为随机搜索树的根节点,不断的从给定无障

碍空间 X_{free} 中通过随机采样 Sample 函数得到 x_{rand},然后利用 NearVertices 函数得出邻近节点集合 X_{near}。邻近节点集合的定义是:给定一个样本节点,搜索树 $T=(V,E)$,一个球型空间 $B_{x_{rand},r}$ 以 x_{rand} 为球心以 r 为半径:

$$Near(x_{rand},T,r):=\{v\in V:v\in B_{x_{rand},r}\}\mapsto X_{near}\subseteq V,$$

更具体地说 $X_{near}=\left\{v\in V:d(x_{rand},v)\leqslant\gamma\left(\frac{\log i}{i}\right)^{\frac{1}{n}}\right\}$,其中 i 为邻近节点的个数,n 为维度数,γ 为常数。

如果邻近节点集合 X_{near} 是空集,则由 NearestVertex 函数计算出最近点 x_{near} 补充到邻近节点集合 X_{near} 中,即邻近节点集合 X_{near} 中只有一个元素 x_{near}。邻近节点集合 X_{near} 中的元素在 GetSortedList 函数的作用下按照代价函数 $c(\sigma)$ 升序排列生成一个列表 L_s,列表 L_s 每一个元素都是由三个变量参数$(x',c(\sigma),\sigma')$所组成,ChooseBestParent 函数遍历列表 L_s,从中选出最优父节点 $x_{\min}\in X_{near}$,使在不碰障碍物的前提下,x_{init} 通过 $x_{\min}$ 到 x_{rand} 路径代价最小。InsertVertex 函数将 $x_{\min}$ 插入搜索树 T 中,REWIREVERTICES 函数进行随机搜索树 T 重组,检查每一个节点,如存在 $x'\in X_{near}$ 从起始点 x_{init} 通过 x' 到 x_{rand} 的路径代价小于现存的路径代价,且是无障碍路径,就将现存路径换成新路径,如条件不满足,遍历搜索树 T 检查 X_{near} 中的每一个节点。依次重复上述过程直到搜到一条从起始点到目标点的无碰撞路径。

RRT* 算法虽然提供了渐近最优性的保证,但是也存在着一些缺点:(1)虽然能实现最优解收敛,但实现最优解收敛速度缓慢;(2)由于使用大量的迭代在计算最优路径,所以需要大量的资源来求解;(3)RRT* 算法舍去的随即采样点,虽然无法直接连接到搜索树现存的节点上,但是其可能在目标点附近,并且能加快实现最优解,而这样的节点不应该被舍去。

2.2 改进的 RRT* 算法(即 IB-RRT*)

IB-RRT* 是专门为了在复杂凌乱的环境中的路径规划所设计的,首先定义了搜索树 T_a 和 T_b 上的邻近节点集合为 X_{near}^a 和 X_{near}^b,连接 x_{init}^a 和 x_{rand} 的路径被定义为 $\sigma_a':[0,s_a]$,连接 x_{init}^b 和 x_{rand} 的路径被定义为 $\sigma_b':[0,s_b]$。

ALGORITHM:IB-RRT*

1. $V_a\leftarrow\{x_{init}^a\};E_a\leftarrow\phi;T_a\leftarrow(V_a,E_a);$
2. $V_b\leftarrow\{x_{init}^b\};E_b\leftarrow\phi;T_b\leftarrow(V_b,E_b);$
3. $\sigma_f\leftarrow\infty;E\leftarrow\phi;$
4. $Connection\leftarrow True$
5. $for\ i\leftarrow 0\ to\ N\ do$
6. $x_{rand}\leftarrow Sample(i)$
7. $\{X_{near}^a,X_{near}^b\}\leftarrow NearVertices(x_{rand},T_a,T_b)$
8. $if\ X_{near}^a=\phi\ \&\&\ X_{near}^b=\phi\ then$
9. $\{X_{near}^a,X_{near}^b\}\leftarrow NearestVertex(x_{rand},T_a,T_b)$
10. $Connection\leftarrow False$
11. $L_s^a\leftarrow GetSortedList(x_{rand},X_{near}^a)$
12. $L_s^b\leftarrow GetSortedList(x_{rand},X_{near}^b)$
13. $\{x_{\min},flag,\sigma_f\}\leftarrow GetBestTreeParent(L_s^a,L_s^b,Connection)$
14. $if\ (flag)\ then$
15. $T_a\leftarrow InsertVertex(x_{rand},x_{\min},T_a)$
16. $T_a\leftarrow RewireVertex(x_{rand},x_{\min},T_a)$
17. $else$
18. $T_b\leftarrow InsertVertex(x_{rand},x_{\min},T_b)$
19. $T_b\leftarrow RewireVertex(x_{rand},x_{\min},T_b)$
20. $E\leftarrow E_a\cup E_b$
21. $V\leftarrow V_a\cup V_b$
22. $return\ (\{T_a,T_b\}=V,E)$

算法首先从非障碍的任务空间 X_{free} 中选择一个随机点 x_{rand},然后由 NearVertices 函数算出两个邻近节点集合 X_{near}^a 和 X_{near}^b,需要注意的是以 x_{rand} 为球心以 r 为半径的球的区域内两棵搜索树的邻近节点集合 X_{near}^a

和 X_{near}^{b} 所包含的节点都要考虑在其内部,即:

$$X_{near}^{a}:=\{v\leftarrow V_a:v\in B_{x_{rand},r}\};\tag{5}$$

$$X_{near}^{b}:=\{v\leftarrow V_b:v\in B_{x_{rand},r}\}.\tag{6}$$

如 X_{near}^{a} 和 X_{near}^{b} 均为空集的时候,这两个集合将从两搜索树中选择最邻近节点来填充集合。这两个集合中的元素将被 GetSortedList 函数所排序,这一个随机节点将采用 BestSelectedTree 函数插入到搜索树 T_a 或 T_b 上。RRT* 中的贪心算法使该算法容易陷入局部陷阱中,而算法 IB-RRT* 不采用贪心算法,连接只发生在以 x_{rand} 为中心的球心之内,只要以 x_{rand} 为中心的球心之内存在 T_a 和 T_b 树上的节点,就可保证两搜索树快速连接。

3 对改进算法的理论分析

在任意的状态空间中,一个算法能收敛到可行解就称该算法具有概率完备性,能在有限次迭代中找到最优解就称该算法具有渐进最有性。因改进算法是一种新的算法,需要对其进行收敛性验证,证明其具有收敛性且能收敛到最优路径。

3.1 概率完备性

概率完备性的定义是:如果一个算法就有概率完备性,则对于任何一个存在可行解的路径规划问题 $(X_{free},x_{init},X_{goal})$ 就有

$$\lim_{n\to\infty}intP(\{\exists\, x_{goal}\in V_n^{ALG}\cap X_{goal}\ such\ that\ x_{init}\ is\ connected\ to\ x_{goal}\ in\ G_n^{ALG}\})=1,\tag{7}$$

对 IB-RRT* 算法分析,可得对于任何一个存在可行解的路径规划问题 $(X_{free},x_{init},X_{goal})$,始终存在一个常数 $a>0$,$n_0\in N$,且两者只取决于 X_{free} 和 X_{goal},

$$P(\{V_n^{IB-RRT^*}\cap X_{goal}\neq\phi\})>1-\mathrm{e}^{-an},\ \forall n>n_0.\tag{8}$$

换句话说,当 $n_0\in N$ 足够大的情况下 $\forall n>n_0$. 都能使得 $\exists\, x_{goal}\in V_n^{ALG}\cap X_{goal}$,这也就意味着 x_{init} 可以连接到 x_{goal}。

3.2 渐近最优性

渐近最优性的定义是:如果一个算法具有渐近最优性,那么对于任何路径规划问题 $(X_{free},x_{init},X_{goal})$ 都有最优解的存在且其代价函数 $c:\sum\to R\geqslant 0$ 会等于一个有限的路径代价 c^*,即:

$$P(\{\limsup_{n\to\infty}Y_n^{ALG}=c^*\})=1,\tag{9}$$

σ^* 表示一个最优路径,定义 $\delta:=\min\{\delta,4r_n\}$,其中 r_n 为 IB-RRT* 算法的链接半径。

对于 $n\in N$,构造一个序列 $\{B_n\}_{n\in N}$ 的球覆盖住 σ_n 记为 $B_n=\{B_{n,1},B_{n,2},\cdots,B_{n,Mn}\}:=coveringBalls(\sigma_n,r_n,2r_n)$,其中

$$r_n=\gamma_{IB-RRT*}\left(\frac{\log n}{n}\right)^{\frac{1}{d}}.\tag{10}$$

对于 $m\in\{1,2,\cdots,M_n\}$,定义 $A_{n,m}$ 事件为存在两个节点 $X_i,X_{i'}\in V_n^{IB\text{-}RRT*}$,其中 $X_i\in B_{n,m}$,$X_{i'}\in B_{n,m+1}$ 而且 $Y_{i'}<Y_i$。在这个情况下,X_i 和 $X_{i'}$ 将会连接到可行路径图 G_n 的边界上。定义 A_n 事件为对于所有的 $m\in\{1,2,\cdots,M\}A_{n,m}$ 事件的集合,即 $A_n=\bigcap_{m=1}^{M}A_{n,m}$.

命题:如果 $\gamma_{IB\text{-}RRT*}>4\left(\frac{\mu(X_{free})}{\zeta_d}\right)^{\frac{1}{d}}$,IB-RRT* 算法是渐近最优的,即:$P(\liminf_{n\to\infty}A_n)=1$.

证明:定义 $\widetilde{A}_{n,m}$ 事件为在 IB-RRT* 算法的节点中有 X_i 和 $X_{i'}$ 两个节点,而且 X_i 和 $X_{i'}$ 连接到可行路径图 $\widetilde{G}_n$ 上,$\widetilde{G}_n$ 为 IB-RRT* 算法迭代 $Poisson(\theta n)$ 次后生成的可行路径图,且 $Poisson(\theta n)$ 采样的空间为 X_{free}。很明显可以得出,$P(A_{n,m}^{c})=P(\widetilde{A}_{n,m}^{c}\,|\,\{Poisson(\theta n)=n\})$,而且:

$$P(A_{n,m}^{c})\leqslant P(\widetilde{A}_{n,m}^{c})+P(\{Poisson(\theta n)>n\}).\tag{11}$$

因为 $P(A_{n,m}^{c})$ 不随着 n 增加而增加,当 $\theta<1$ 时,$P(\{Poisson(\theta n)>n\})\leqslant e^{-an}$,其中 $a>0$ 是个与 n 无关的常数。为了计算 $P(\widetilde{A}_{n,m}^{c})$,定义 $N_{n,m}$ 表示在 $B_{n,m}$ 内部节点个数,对于所有的 $m\in\{1,2,\cdots,M_n\}$ 可得 E

$[N_{n,m}]=\frac{\zeta_d\gamma_{RRT^*}^d}{\mu(X_{free})}\log n$,为了简化表达式,令 $\alpha:=\frac{\zeta_d\gamma_{RRT^*}^d}{\mu(X_{free})}$,令 $\varepsilon\in(0,1)$是与 n 无关的常量,定义事件 $C_{n,m,\varepsilon}:=\{N_{n,m}\geqslant(1-\varepsilon)E[N_{n,m}]\}=\{N_{n,m}\geqslant(1-\varepsilon)\alpha\log n\}$,因为 $N_{n,m,\varepsilon}$是二项分布,所以 $P(C_{n,m,\varepsilon}^c)=P(\{N_{n,m,\varepsilon}^c\leqslant(1-\varepsilon)E[N_{n,m}]\})\leqslant e^{-\alpha H(\varepsilon)\log n}=n^{-\alpha H(\varepsilon)}$,其中 $H(\varepsilon)=\varepsilon+(1-\varepsilon)\log(1-\varepsilon)$,$H(\varepsilon)$是一个连续函数,其中 $H(0)=0$,$H(1)=1$,所以当 ε 趋向于 1 的时候,$H(\varepsilon)$也趋向于 1。

$$P(\widetilde{A}_{n,m}^c)=P(\widetilde{A}_{n,m}^c\mid C_{n,m,\varepsilon}\cap C_{n,m+1,\varepsilon})P(C_{n,m,\varepsilon}\cap C_{n,m+1,\varepsilon})+P(\widetilde{A}_{n,m}^c\mid(C_{n,m,\varepsilon}\cap C_{n,m+1,\varepsilon})^C)P((C_{n,m,\varepsilon}\cap C_{n,m+1,\varepsilon})^C)\leqslant P(\widetilde{A}_{n,m}^c\mid C_{n,m,\varepsilon}\cap C_{n,m+1,\varepsilon})P(C_{n,m,\varepsilon}\cap C_{n,m+1,\varepsilon})+P(C_{n,m,\varepsilon}^c)+P(C_{n,m+1,\varepsilon}^c).\tag{12}$$

首先,节点生成过程中空间的独立性可得 $P(C_{n,m,\varepsilon}\cap C_{n,m+1,\varepsilon})=P(C_{n,m,\varepsilon})P(C_{n,m+1,\varepsilon})\leqslant n^{-2\alpha H(\varepsilon)}$.

之后,观察 $P(A_{n,m}^c\mid N_{n,m}=k,N_{n,m+1}=k')$是一个非增的函数,即 $\widetilde{A}_{n,m}$事件发生的概率不会随着 $B_{n,m}$和 $B_{n,m+1}$之内的节点数增加而增加,

$$P(\widetilde{A}_{n,m}^c\mid C_{n,m,\varepsilon}\cap C_{n,m+1,\varepsilon})=P(\widetilde{A}_{n,m}^c\mid\{N_{n,m}\geqslant(1-\varepsilon)\alpha\log N_{n,m},N_{n,m+1}\geqslant(1-\varepsilon)\alpha\log N_{n,m+1}\})\leqslant P(\widetilde{A}_{n,m}^c\mid\{N_{n,m}=(1-\varepsilon)\alpha\log N_{n,m},N_{n,m+1}=(1-\varepsilon)\alpha\log N_{n,m+1}\}).\tag{13}$$

这个概率可以按照如下的方式来计算,从均匀分布的次序统计来说,最小 $\alpha\log n$ 采样点均匀分且独立的分布在$[0,1]$,其概率分布函数为:

$$f_{\min}(x)=\frac{(1-x)^{\alpha\log n-1}}{Beta(1,\alpha\log(n))},\tag{14}$$

其中,$Beta(\cdot,\cdot)$是 Beta 函数,最大 $\alpha\log n$ 采样点均匀分且独立的分布在$[0,1]$,其累积分布函数是:$F_{\max}(x)=x^{\alpha\log n}$.

$$P(\widetilde{A}_{n,m}^c\mid C_{n,m,\varepsilon}\cap C_{n,m+1,\varepsilon})\leqslant\int_0^1F_{\max}(x)f_{\min}(x)\mathrm{d}x=\frac{Gamma((1-\varepsilon)\alpha\log n)Gamma((1-\varepsilon)\varepsilon\log n)}{2Gamma(2(1-\varepsilon)\alpha\log n)}\leqslant\frac{((1-\varepsilon)\alpha\log n)!((1-\varepsilon)\alpha\log n)!}{2(2(1-\varepsilon)\alpha\log n)!}=\frac{((1-\varepsilon)\alpha\log n)!}{2(2(1-\varepsilon)\alpha\log n)(2(1-\varepsilon)\alpha\log n-1)\cdots1}\leqslant\frac{1}{2^{((1-\varepsilon)\alpha\log n)}}=n^{-\log(2)(1-\varepsilon)\alpha},\tag{15}$$

其中 $Gamma(\cdot)$是 gamma 函数,整合上式可得:

$$P(\widetilde{A}_{n,m}^c)\leqslant n^{-\alpha(2H(\varepsilon)+\log(2)(1-\varepsilon))}+2n^{-\alpha H(\varepsilon)}\tag{16}$$

因为 $2H(\varepsilon)+\log(2)(1-\varepsilon)$和 $H(\varepsilon)$在$(0.5,1)$都是连续的增函数,当 ε 逐渐逼近于 1 的时候,前者等于 $2-\log(4)>0.5$,后者等于 1,所以存在一个$\bar{\varepsilon}\in(0.5,1)$使得 $2H(\bar{\varepsilon})+\log(2)(1-\bar{\varepsilon})>0.5$ 和 $H(\bar{\varepsilon})>0$ 同时成立,

$$P(\widetilde{A}_{n,m}^c)\leqslant n^{-\alpha/2}+2n^{-\alpha/2}=3n^{-\alpha/2},\tag{17}$$

然后,

$$P(A_{n,m}^c)\leqslant P(\widetilde{A}_{n,m}^c)+P(Poisson(\theta n)>n)\leqslant3n^{-\alpha/2}+e^{-an},\tag{18}$$

那么事件 A_n 发生的概率:

$$P(A_n^c)=P((\bigcap_{m=1}^{Mn}A_{n,m})^c)=P(\bigcup_{m=1}^{Mn}A_{n,m}^c)\leqslant\sum_{m=1}^{Mn}P(A_{n,m}^c)=MnP(A_{n,1}^c),\tag{19}$$

B_n 中含有球的个数可以表示为:

$$|B_n|=M_n\leqslant\beta\left(\frac{n}{\log n}\right)^{\frac{1}{d}},\tag{20}$$

其中,结合上面的不等式可以得到:

$$P(A_n^c)\leqslant\beta\left(\frac{n}{\log n}\right)^{\frac{1}{d}}(3n^{-\frac{\alpha}{2}}+e^{-an}),\tag{21}$$

其中,$P(\limsup_{n\to\infty}A_n^c)=0$ 这就意味着 $P(\limsup_{n\to\infty}A_n)=1$。即可得证 IB-RRT* 是渐近最优的算法。

3.3 快速收敛到最优路径

给定一个随机节点 x_{rand} 和最小路径代价函数 $\sigma_a[0,s_a]:=\{\sigma_a(0)=x_{init}^a,\sigma_a(s_a)=x_{rand}\}$ 和 $\sigma_b[0,s_b]:=\{\sigma_b(0)=x_{init}^b,\sigma_b(s_b)=x_{rand}\}$. IB-RRT* 算法的智能样本插入的过程可以被归纳为$\{x_{rand}\in V_a:c(\sigma_a)\leqslant c(\sigma_b)$或

$x_{rand}\in V_b:c(\sigma_b)<c(\sigma_a)\}$,也就是随机节点 x_{rand} 总是被插到距离更近的那棵随机树上,这一措施保证随机节点 x_{rand} 总插到邻近节点密度较高的区域中。又因重组随机树过程中试图减少两棵随机树与理论最短距离之差 $\|\sigma'_i-\sigma_i\|$,这一过程在每一次 x_{rand} 插入到随机树时都会被执行。如果在最邻近节点集合 X_{near} 特定节点 x' 的路径代价小于现有代价,就将 x_{rand} 变为 x' 的父节点。IB-RRT* 插入节点到邻近密度较高的区域中,使得重组搜索树在每一次的循环中作用发挥到最大,同时 IB-RRT* 是同时生长出两棵随机树,使得 IB-RRT* 的收敛速度远快于 RRT*。

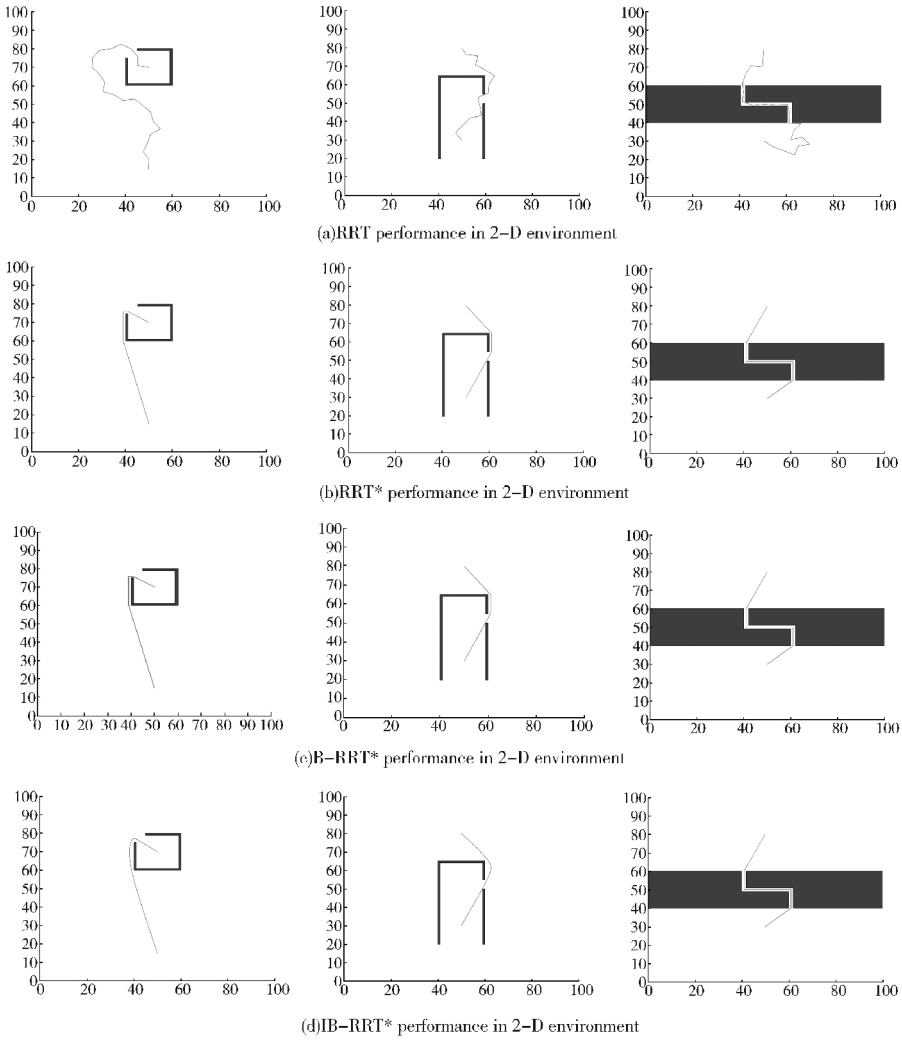
4 实验与仿真

4.1 算法仿真实验

将进行仿真实验来验证算法的有效性,并作为上一节中提出的概率完备性,渐近最优性提供一个实验的验证。通过对典型障碍物的规划实验,对比原有的 RRT[21]、RRT*[15]、B-RRT*[16] 算法和本文提出的 IB-RRT* 的运算速度、运算效率、迭代次数、路径代价和路径规划的成功率。每一个算法对应于每一个障碍物地图时都将运行程序 50 次,并记录以上参数,所有实验均在一个有着 4 GB 内存的 Corei5 的处理器,主频为 2.4 MHz 的电脑上运行得出的。

4.1.1 二维环境中的仿真实验

在二维环境的仿真中,选用一些比较有代表性的能够验证算法性能的障碍物地图(图 2)。



(a)RRT performance in 2-D environment

(b)RRT* performance in 2-D environment

(c)B-RRT* performance in 2-D environment

(d)IB-RRT* performance in 2-D environment

Fig. 2 ALG performance in 2-D environment

图 2 算法在二维环境中的规划

其中,图 2(a)—(d)为四种规划算法在三种不同的规划问题中求解的结果。

4.1.2 三维环境中的仿真实验

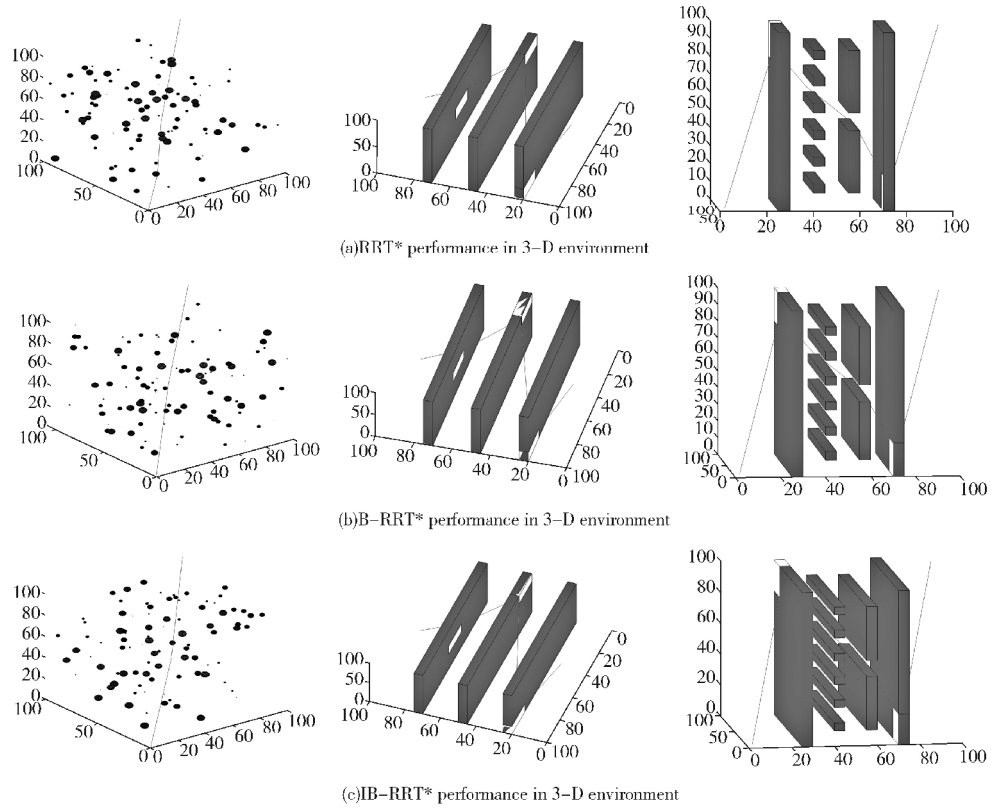
在三维环境模型的仿真试验中,采用了三种不同的障碍物环境对算法进行仿真实验,以探究算法的性能。

表 1 计算最优路径的实验结果

Table 1 Experimental results for computing optimal path solution

障碍物地图环境	算法	$i_{\min}$	$i_{\max}$	i_{avg}	$t_{\min}(s)$	$t_{\max}(s)$	$t_{avg}(s)$	C	$Fail$
三维随机障碍物地图环境	RRT*	107 810	110 851	108 965	17.8	19.3	18.8	**	6
	B-RRT*	30 901	38 086	36 128	6.4	8.1	7.4	**	4
	IB-RRT*	19 507	22 525	21 290	4.4	5.3	5.1	**	1
三维复杂障碍物地图环境	RRT*	1 430 381	1 440 619	1 437 342	242.1	247.4	244.1	205.6	14
	B-RRT*	495 961	503 240	498 972	102.1	106.5	105.1	205.6	6
	IB-RRT*	97 885	112 857	111 139	22.3	27.3	26.2	205.6	3
三维窄道障碍物地图环境	RRT*	1 277 376	1 301 698	1 290 674	216.9	221.9	218.5	345.1	9
	B-RRT*	533 276	561 347	551 771	110	117.1	115.9	345.1	3
	IB-RRT*	127 363	143 806	134 421	30.3	35.2	31.4	345.1	0

其中,i 为迭代次数;t 为搜索时间;C 为路径代价;Fail 这一栏是指当迭代次数超过 300 万次,就默认为算法搜索失败。



(a)RRT* performance in 3-D environment

(b)B-RRT* performance in 3-D environment

(c)IB-RRT* performance in 3-D environment

Fig. 3 ALG performance in 3-D environment

图 3 算法在三维环境中的规划(左为随机障碍物地图,中为复杂障碍物地图,右为窄道障碍物地图)

从图 3 和表 1 中可以看出来对于同一个路径规划问题,经过大量的迭代后,IB-RRT* 在任务空间中搜索到的路径和 RRT*、B-RRT* 有许多相似的地方,这也间接的证明了这三种算法都是渐近最优的算法,最终都能收敛到最优解。但从表 1 中的数据可以看出 IB-RRT* 在搜索速度和搜索效率上均有显著提高,在三维随机障碍物地图环境中,IB-RRT* 的平均迭代次数是 21 290 次,而 B-RRT* 的平均迭代次数是 36 128 次,是 IB-RRT* 的 1.69 倍,而 RRT* 的平均迭代次数是 108 965 次,是 IB-RRT* 的 5.12 倍,搜索用时也从 RRT* 的 18.8 s,B-RRT* 的 7.4 s 降低到 IB-RRT* 的 5.3 s;在三维复杂障碍物地图环境中,IB-RRT* 的平均迭代次数是 111 139 次,而 B-RRT* 的平均迭代次数是 498 972 次,是 IB-RRT* 的 4.49 倍,而 RRT* 的平

均迭代次数是 1 437 342 次,是 IB-RRT* 的 12.93 倍,搜索用时也从 RRT* 的 244.1 s,B-RRT* 的 105.1 s 降低到 IB-RRT* 的 26.5 s;在三维窄道障碍物地图环境中,IB-RRT* 的平均迭代次数是 134 421 次,而 B-RRT* 的平均迭代次数是 551 771 次,是 IB-RRT* 的 4.10 倍,而 B-RRT* 的平均迭代次数是1 290 674次,是 IB-RRT* 的 9.61 倍,搜索用时也从 RRT* 的 218.5 s,B-RRT* 的 115.9 s 降低到 IB-RRT* 的 31.4 s。由此我们可以得知,IB-RRT* 在搜索速度方面远快于 B-RRT* 和 RRT*,且其搜索路径成功的概率也大幅提高,算法的稳定性增强,验证了前文的理论证明是正确。同时搜索的路径也相对其他两种种算法更为平缓,并且最终生成的可执行路径曲率也是连续变化的,满足了车辆的运动约束要求。

4.2 基于 ROS 的移动机器人仿真实验

移动机器人模型如图 4 所示,底面直径为 0.25 m,高度为 0.3 m。环境模型如图 5a 所示,每个大房间墙壁长度和宽度均为 5 m,高度为 2.5 m,厚度为 0.2 m;小房间墙壁长度和宽度均为 3 m,高度为 2.5 m,厚度为 0.2 m;四个门洞高度为 2 m,宽度为 1 m。模型构建完成后,移动机器人调用 ROS 中的 gmapping 功能包完成 SLAM,实现场景地图的构建,最后获得室内场景的地图(图 6)。

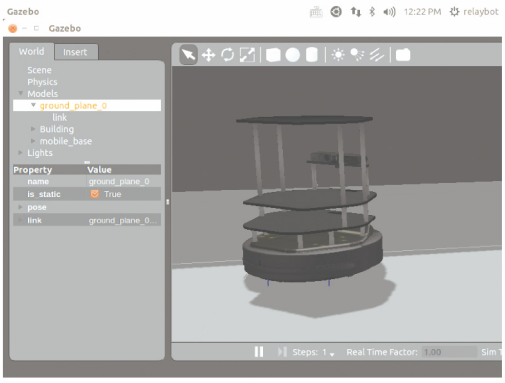
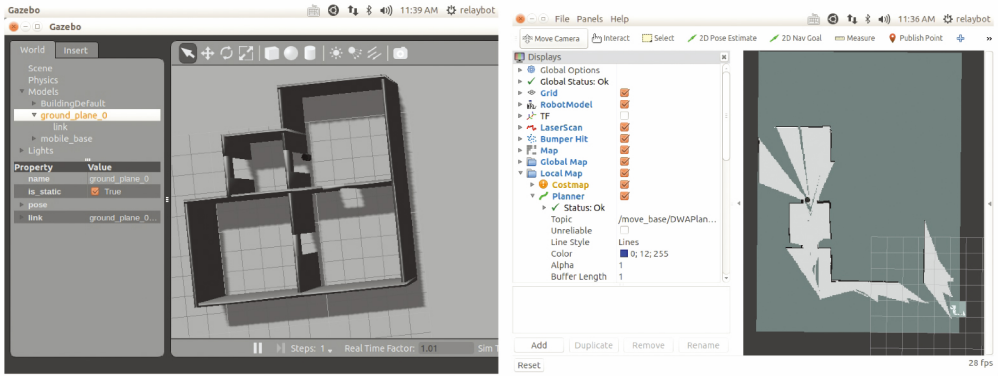


Fig. 4 Mobile robot

图 4 移动机器人



a) Gazebo 下的 SLAM 过程 b)RVIZ 下的 SLAM 的过程

Fig. 5 The process of SLAM

图 5 SLAM 过程

这些文件配置好了之后,便可以利用改进的 RRT 算法对移动机器人进行路径规划,在地图上随机选择一目标点,并选定位置和姿态,改进的 RRT 算法便会生成一条无障碍的路径,如图 7 所示,绿色的小轨迹为算法规划出来的轨迹。

图 7 给出了 IB-RRT* 动态规划过程。IB-RRT* 采用的是实时重规划,Laserscan_ nodelet_ manager 节点实时感知外界地图环境消息,将移动机器人附近的障碍物信息以 scan 的主题发布出去,IB-RRT* 算法规划器接收该消息,并检测有没有新的障碍物,如果有新的障碍物 IB-RRT* 则进行实时重规划,以避开障碍物,图中的彩色区域为移动机器人感知到的周围障碍物的信息,绿色的线为初始规划出来的路径,红色的线为实时重规划的路径。因此,该算法也可以适用于动态环境的路径规划问题。

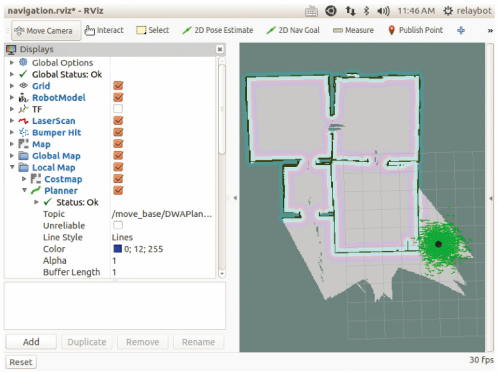
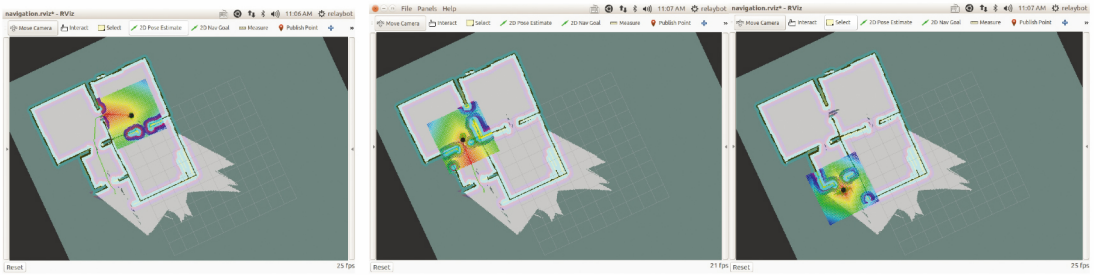


Fig. 6 Map information in RVIZ

图 6 RVIZ 下显示的场景地图信息



a)开始规划 b)规划的途中 c)规划完成

Fig. 7 Process of planning

图 7 规划过程

5 结论

以往的规划算法中基本没有考虑到移动机器人自身的约束,可能会导致虽然已经规划出路径,但无法被机器人所执行。本文提出了一种新的 RRT 算法,它舍弃原算法贪心思想并引入启发式采样节点插入算法,同时从起始点和目标点开始扩展搜索树,使得算法的收敛速度远快于改进之前;并进行理论分析证明改进算法具有概率完备性、渐近最优性,且确实优于之前的算法。通过改进算法与各种算法在不同的障碍物环境中进行反复的仿真实验,仿真的结果也说明了有效快速的解决复杂环境下的路径规划问题。但是该算法只能解决低速环境下的路径规划问题,下一步将对受滑动干扰的移动机器人模型的规划问题及高速运动的规划问题,进一步的提高算法的实用性和鲁棒性。

参考文献:

[1] **李磊,叶涛,谭民,等. 移动机器人技术研究现状与未来[J]. 机器人**,2002,**24**(5):475-480. DOI:10.1397 3/j. cnki. robot. 2002. 05. 020.

[2] **王勇,蔡自兴,周育人. 肖赤心约束优化进化算法[J]. 软件学报**,2009,**20**(1):11-29. DOI:10. 3724/SP. J. 1001. 20 09. 03363.

[3] Song J Z,Dai B,Shan E Z,*et al*. An Improved RRT Path Planning Algorithm[J]. *Acta Electronica Sinica*,2010,**38**(B02):225-228.

[4] Elbanhawi M,Simic M. Sampling-based Robot Motion Planning:A Review[J]. *IEEE Access*,2014,**2**(1):56-77. DOI:10. 1109/ACCESS. 2014. 2302442.

[5] Qureshi A H,Ayaz Y. Intelligent Bidirectional Rapidly-exploring Random Trees for Optimal Motion Planning in Complex Cluttered Environments[J]. *Robotics and Autonomous Systems*,2015,**68**:1-11. DOI: 10. 1016/j. robot. 2015. 02. 007.

[6] Karaman S,Frazzoli E. Sampling-based Algorithms for Optimal Motion Planning[J]. *The International Journal of Robotics Research*,2011,**30**(7):846-894. DOI:10. 1177/0278364911406761.

[7] Park J H,Tai Y W. A Simulation Based Method for Vehicle Motion Prediction[J]. *Computer Vision and Image Under-*

standing, 2015, **136**: 79-91. DOI: 10.1016/j.cviu.2015.03.004.

[8] Vonásek V, Saska M, Winkler L, *et al*. High-level Motion Planning for CPG-driven Modular Robots[J]. *Robotics and Autonomous Systems*, 2015, **68**: 116-128. DOI: 10.1016/j.robot.2015.01.006.

[9] Doshi A A, Postula A J, Fletcher A, *et al*. Development of Micro-UAV with Integrated Motion Planning for Open-cut Mining Surveillance[J]. *Microprocessors and Microsystems*, 2015, **39**(8): 829-835. DOI: 10.1016/j.micpro. 2015.07.008.

[10] Park K J, Won M. People Tracking and Accompanying Algorithm for Mobile Robot Using Kinect Sensor and Extended Kalman Filter[J]. *Transactions of the Korean Society of Mechanical Engineers A*, 2014, **38**(4): 345-354. DOI: 10.3795/KSME-A.2014.38.4.345.

[11] Kuffner J, RRT-Connect S L V. An Efficient Approach to Single-query Path Planning ieee International Conference on Robotics and Automation[J]. *San Francisco*, 2000, **2**: 473-479. DOI: 10.1109/ROBOT.2000.844730.

[12] LaValle S M, Kuffner J J. Randomized Kinodynamic Planning[J]. *The International Journal of Robotics Research*, 2001, **20**(5): 378-400. DOI: 10.1177/02783640122067453.

[13] Urmson C, Simmons R G. Approaches for Heuristically Biasing RRT Growth[C]// IROS, 2003, **2**: 1178-1183. DOI: 10.1109/IROS.2003.1248805.

[14] Ferguson D, Stentz A. Anytime Rrts[C]// 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2006: 5369-5375. DOI: 10.1109/IROS.2006.282100.

[15] Karaman S, Frazzoli E. Incremental Sampling-based Algorithms for Optimal Motion Planning[J]. *Robotics Science and Systems VI*, 2010, **104**. DOI: 10.15607/rss.2010.vi.034.

[16] Jordan M, Perez A. Optimal Bidirectional Rapidly-exploring Random Trees, MIT-CSAIL-TR-2013-021[R]. Cambridge, UK: Computer Science and Artificial Intelligence Laboratory, 2013.

[17] Qureshi A H, Mumtaz S, Iqbal K F, *et al*. Triangular Geometry based Optimal Motion Planning using RRT*-motion Planner[C]// 2014 IEEE 13th International Workshop on Advanced Motion Control (AMC). IEEE, 2014: 380-385. DOI: 10.1109/AMC.2014.6823312.

[18] Bac C W, Roorda T, Reshef R, *et al*. Analysis of a Motion Planning Problem for Sweet-pepper Harvesting in a Dense Obstacle Environment[J]. *Biosystems Engineering*, 2015, **146**: 85-97. DOI: 10.1016/j.Biosystemseng. 2015.07.004.

[19] **徐娜, 陈雄, 孔庆生, 等. 非完整约束下的机器人运动规划算法**[J]. **机器人**, 2011, **33**(6): 666-672. DOI: 10.3724/SP.J.1218.2011.00666.

[20] **杜明博, 梅涛, 陈佳佳, 等. 复杂环境下基于** RRT **的智能车辆运动规划算法**[J]. **机器人**, 2015, **37**(4): 443-450. DOI: 10.13973/j.cnki.robot.2015.0443.

[21] LaValle S M. Rapidly-exploring Random Trees: A New Tool Path Planning[R]. Ames, USA: Iowa State University, 1998.