控制与决策
Control and Decision

基于B-RRT*FND算法的移动机器人路径规划

张腾龙, 李擎

引用本文:
张腾龙,李擎. 基于B-RRT*FND算法的移动机器人路径规划[J]. 控制与决策, 2023, 38(11): 3121-3127.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2022.0236

基于B-RRT*FND算法的移动机器人路径规划

张腾龙, 李 擎†

(北京信息科技大学 自动化学院，北京 100085)

摘 要: 针对RRT*FN算法获取路径解的速度慢,且无法应用于动态环境等问题,提出固定节点数的动态双向渐近最优快速随机扩展树算法(bidirectional RRT* fix-node dynamic, B-RRT*FND),用于解决移动机器人在二维空间内快速实时获取无碰撞路径的问题. 所提出算法基于RRT*FN算法,采用双向贪婪搜索方法加快路径搜索速度,解决单向RRT算法由于随机采样的盲目性造成的搜索速度慢、在狭窄环境下难以搜索到解的问题;利用固定节点算法在规划过程中不占用过多计算量的特点,在路径迭代优化过程中,实时更新地图信息,并对被破坏的原始路径进行修复重连,以完成算法的动态规划. 将所提出算法与RRT、RRT*FN等算法在3种环境下进行对比仿真,验证结果表明,所提出算法在规划速度、路径解长度以及动态规划性能方面具有较好效果.

关键词: 移动机器人；RRT*FN算法；动态路径规划；双向贪婪搜索；渐近最优；路径修复

中图分类号: TP273 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2022.0236

引用格式: 张腾龙,李擎. 基于B-RRT*FND算法的移动机器人路径规划[J]. 控制与决策, 2023, 38(11): 3121-3127.

Path planning of AGV based on B-RRT*FND algorithm

ZHANG Teng-long, LI Qing†

(School of Automation，Beijing Information Science and Technology University，Beijing 100085，China)

Abstract: Aiming at the problems of the RRT*FN algorithm about its slow speed to obtain the path solutions and unable to applied in dynamic environment, a dynamic bidirectional RRT* algorithm with fixed nodes (B-RRT*FND) is proposed, which is used to solve the problem of how to obtain collision-free paths in 2D space quickly in real-time with a robot. The algorithm is based on the RRT*FN algorithm, using the bidirectional greedy search method to speed up the searching and solve the problems of the unidirectional RRT algorithm about its slow searching speed as well as the difficulty of solving in the narrow environment caused by blindly random sampling. Meanwhile, taking the advantage of the fact that fixed nodes do not occupy too much computation in planning, in the process of path iterative optimization, the algorithm updates the map information in real-time, and repairs the broken original path to complete the dynamic path planning. Compared with the RRT, the RRT*FN and other algorithms in three environments, the B-RRT*FND algorithm is superior in planning speed, length of path and dynamic programming performance.

Keywords: AGV；RRT*FN algorithm；dynamic path planning；bidirectional greedy-search；asymptotic optimality；path repairing

0 引 言

移动机器人的路径规划问题一直以来均是机器人研究领域的热点问题,随着机器人承担工作的复杂化,移动机器人的使用场景也呈现多元化的发展趋势[1],这要求搭载在移动机器人上的路径规划算法具有出色的规划效果[2]. 路径规划即规划出一条机器人从起始点到目标点的全局无碰撞路径. 学者们已对路径规划方法进行了大量研究,根据路径规划算法的发展趋势可将其分为基于图的路径规划算法和基于采样的路径规划算法,其中基于图路径规划算法包括*A**(a star)算法、Dijkstra算法[3]等,传统的规划算法不具备动态规划能力. 为了使得算法满足AGV的实施规划目标,Maw等[4]提出了能够在动态环境下应用实时动态*A**算法(anytime dynamic *A**, ADA*),刘军等[5]引入了导向函数和评价函数优化的导向*D**算法,基于图的算法需要对地图进行栅格化建模,在栅格空间中采用最佳优先搜索算法来减少状态总数,进而确定由一组无碰撞节点组成的最小路径[6]. 由

收稿日期: 2022-02-11；**录用日期:** 2022-06-15.

基金项目: 国家自然科学基金项目(61971048)；国家重点研发计划项目(2020YFC1511702).

责任编委: 张文安.

†通讯作者. E-mail: liqing@bistu.edu.cn.

于分辨率最优性和分辨率完整性[7],该类算法可返回最优解,相对的路径搜索时间也会变得相当长,基于图的路径规划算法对于要求实时规划的情况并不适用. 基于采样的路径规划方法主要有概率路线图算法(probabilistic road maps, RPM)、快速随机搜索树算法(rapidly exploring random tree, RRT)[8]等,此类算法基于增量式随机采样,能够快速高效地规划出可行路径. 基于随机采样的路径规划算法避免对空间进行建模,获得路径解的速度较快,适用于本文要求实时性较高的场合,但是生成采样点的方式随机,RRT规划出的路径通常并非最优路径,因此通常需要进行路径优化,RRT*、B-RRT*等算法引入了父节点重选和节点重连方法,通过采样点的增加对目标路径进行渐近优化,在采样点趋于无穷时,一定能够获得最优路径[9]. 然而,增加采样点而获得渐近最优路径的过程较为漫长. 为了提高路径解收敛速度,Gammell等[10]提出的informedRRT*算法改变了采样空间的选择,将全局随机采样限制在一个椭圆空间内的随机采样;Taheri等[11]提出了基于模糊贪婪RRT的搜索算法(FG-RRT),通过评价父节点质量改变生成后代节点的权重,魏武等[12]提出了一种基于双树Quick-RRT*算法路径规划算法,从起始点和终点轮流生长两树以提高收敛速度. 此外,最优化RRT算法在运行过程中随着节点数量的增加,计算复杂度和运行时间会呈指数级增长. 为了解决这一问题,Spanogianopoulos等[13]提出了RRT*FN(RRT fixed nodes)算法,可减少路径上的多余叶子节点,提高规划速度.

受RRT*FN算法的启发,本文提出一种能够应用于动态环境下且引入双向贪婪搜索思想的B-RRT*FND算法,以提高算法在规划时间、路径解长度、收敛速度以及路径形状等多方面的性能,以满足实时规划要求,使得AGV在动态环境中也能够快速实时获得最优无碰撞路径.

1 相关工作

1.1 问题定义

本文针对AGV移动机器人在空间中进行从起始点向目标点移动的路径规划问题进行建模,与文献[14-16]定义的方式类似,定义路径规划问题中的状态空间$X\subseteq R_n$,通过AGV激光传感器获得的数字地图中包含的障碍物信息定义为障碍物空间$X_{\rm obs}\subsetneqq X$,定义障碍物空间对于状态空间的补集为自由空间X_f,$\boldsymbol{X}_f=\boldsymbol{X}_{\rm obs}C_UX$,即机器人能够自由运动的空间. 设AGV机器人在状态空间中的起始点为$p_{\rm start}\in\boldsymbol{X}_f$,所移动的目标点为$p_{\rm end}\in\boldsymbol{X}_f$,定义路径点集$\sigma:[0,1]\subset X_f$,且$\sigma(0)=p_{\rm start},\sigma(1)=p_{\rm end}$. 则路径规问题转化为在自由空间$X_f$内求解点集$\sigma[0,1]$. 若点$p\in\sigma[0,1]$以及点$p$间的连线与$X_{\rm obs}$不相交,则称此$\sigma[0,1]$上的点集为一条可行路径. 定义路径代价函数$c(\sigma)$,$c(\sigma)$在数值上等于路径解$\sigma$点集间连线的欧氏距离之和,有

$$c(\sigma)=\sum_{i=1}^{n}\left\|\sigma\left(\frac{i}{n}\right)-\sigma\left(\frac{i-1}{n}\right)\right\|. \tag{1}$$

其中:n为集合σ中元素个数,空间中的可行路径σ不唯一. 定义Σ为所有可行σ的集合,$C:\sum\to R_{\geqslant 0}$为所有路径代价集合. 本文使用路径代价作为评价路径解的评价指标,对于最优路径求解问题,路径代价越低意味着路径越优,则最优路径的求解问题转化为在自由空间X_f中求解使得路径代价最小的σ^*,有

$$\sigma^*=\mathrm{argmin}\{c(\sigma)|\sigma(0)=p_{\rm start},\sigma(1)=p_{\rm end},\ \forall s\in[0,1],\sigma(s)\in X_f\}. \tag{2}$$

而对于本文的动态规划情况,随着智能体和障碍物的变化,$p_{\rm start}$和$p_{\rm end}$可能不唯一,于是定义起始点和路径目标点的集合为$P_{\rm start}$和$P_{\rm end}$,$\sigma(0)\in P_{\rm start},\sigma(1)\in P_{\rm end}$. 则定义(2)可改写为

$$\sigma^*=\mathrm{argmin}\{c(\sigma)|\sigma(0)\in P_{\rm start},\sigma(1)\in P_{\rm end},\ \forall s\in[0,1],\sigma(s)\in X_f\}. \tag{3}$$

1.2 RRT算法

RRT算法是一种最具代表性的基于采样的路径规划算法,该算法可通过从空间中随机生成采样点,以起始点作为根节点,从与其最近的且链接无碰撞的采样节点以连线方向按规定步长生长,实现对树进行扩展的目的,直至叶子节点与目标点间的连线无碰撞,则路径规划完毕. 该算法的基本步骤如下.

step 1: 给定地图空间M,起始点$p_{\rm start}$和目标点$p_{\rm end}$.

step 2: 通过在空间中随机采样得到$p_{\rm rand}$.

step 3: 从与其最近的节点$p_{\rm near}$点开始以步长s向$p_{\rm rand}$生长,并定义新生成的点为$p_{\rm new}$.

step 4: 采样点会有一定概率不沿着随机采样点$p_{\rm rand}$生长,而是选择直接向终点$p_{\rm end}$生长.

step 5: 当树生长至$p_{\rm end}$或$p_{\rm new}$与$p_{\rm end}$的连线与障碍物不相交时,则可行路径σ生成完毕.

虽然RRT算法具有概率完备性,能够快速得到空间中的可行解,但其搜索解的过程是盲目的,具体体现在由于$p_{\rm rand}$通过随机采样获得,使得树的生长方向随机,且算法缺乏对节点拓展的记忆性,导致产生冗余.

1.3　RRT*FN算法

RRT* 算法属于最优算法,通过在更新迭代中获得足够多的采样点可获得渐近最优路径,算法添加了父节点重选和节点重连2个操作.

父节点重选: 以 p_{new} 为圆心,以 R 为半径的区域内节点为备选父节点,计算以这些节点为父节点的路径代价,选择使得路径代价最小的节点作为新的父节点并连接,若路径存在碰撞,则选择其他备用父节点.

节点重连: 以 p_{new} 为圆心,选定以 R 为半径的范围,尝试将范围内的节点的父节点改为 p_{new},若如此可降低路径总代价,则断开该节点与其父节点的连接,与 p_{new} 相连,若连接存在碰撞,则放弃本次连接,继续选择范围内其他节点依次进行尝试.

若随机采样点 p_n 恰好取于空间中 p_{start} 到 p_{end} 的最优路径 σ^* 上,则路径代价会由于 p_n 成为其邻近节点的父节点而减小,经过足够多次的采样,总有

$$\begin{cases} L\{p_1, p_2, \ldots, p_n\} = L^*, \\ \forall p \in L, p_{\text{rand}}(x) = p,\ x \in (0, \text{iter}). \end{cases} \tag{4}$$

其中 iter 为迭代次数,则其渐近最优性得证. 由于算法在扩大搜索规模的同时,父节点重选和节点重连的操作对树中所有点进行遍历,造成巨大的内存负担,可通过固定节点数的方式提高算法的搜索效率. RRT*FN 算法 (fixed-nodes RRT*) 在 RRT* 算法的基础上引入了最大节点数的概念,该算法设定了树中允许的最大节点数,当状态空间中节点数大于预设的节点时,随机删去一个除末端节点外的的无子叶子节点. RRT*FN算法步骤如下.

step 1: 同RRT算法step 1 ~ step 3.

step 2: 进行父节点重选和节点重连.

step 3: 每生成一个 p_{new},检查空间中存在的节点数,若大于最大节点数FixNodes,则随机删除非路径最后节点的无子叶子节点 ($p_{\text{delete}} \in p_{\text{leaf}}$ except p_{new}).

step 4: 当树生长至 p_{end} 或 p_{new} 与 p_{end} 的连线与障碍物不相交时,则可行路径 σ 生成完毕. 重复 step 1 ~ step 3 对路径解进行渐近优化.

2　B-RRT*FND算法

受 RRT*FN 算法的启发,为了进一步提高 RRT*FN 算法的搜索效率,并将算法运用于动态环境下,本文提出了 B-RRT*FND 算法 (bidirectional RRT* fix-node dynamic),改进算法针对原 RRT*FN 算法作出以下改进,将双向贪婪搜索策略与RRT*FN算法相结合,进一步加快了 RRT*FN 算法的规划速度. 并利用动态更新和路径修复的方法,使得算法能够应用于未知和移动障碍物的情况.

2.1　贪婪双向搜索

改进算法将双向贪婪搜索策略与RRT*FN算法相结合,以解决单侧树生长盲目性的问题. RRT*FN 算法较RRT* 算法在搜索速度上没有优势,仅能够减少冗余采样点以避免冗余生长,在迭代次数较多导致树的规模较大时具有限制树的规模以提高程序运行速度的作用,与快速获得路径解无关. 添加了双向贪婪搜索策略后,算法规划出的路径具有更明显的指向性,可较为快速地获得一个初始路径.

贪婪双向搜索需要在起始点和目标点同时建立2棵随机搜索树 Tree 1 和 Tree 2, 2 棵树分别向对方生长,在生长过程中使用贪婪策略. 具体如下.

1) 规定 2 棵树的起点 p_{start} 与 p_{end} 互为对方的终点.

2) 在 Tree 1 生成新的节点 p_{new} 后,从 p_{new} 向 Tree 2 中距离自己最近的节点连续生长 (以确保路径代价最小),直至碰撞或到达目标.

3) 若生长后,连续 Tree 1 节点数大于另一棵树的节点数,则选择 Tree 2 为生长目标.

本文将双向贪婪搜索策略与RRT*FN算法相结合以改善传统单向随机树生长目的性差的缺陷. 从起始点 p_{start} 和目标点 p_{end} 分别建立2棵树,并分别向对方贪婪生长. 绿色线为起点树,蓝色线为终点树. 黑色圆圈为障碍物. 状态空间中的节点按下标顺序生成,若规定每棵树中节点数量不大于5,当 $\text{Tree}_{\text{start}} \bigcap \text{Tree}_{\text{end}} \neq \varnothing$ 时,2棵树连接. 改进算法如图1所示.

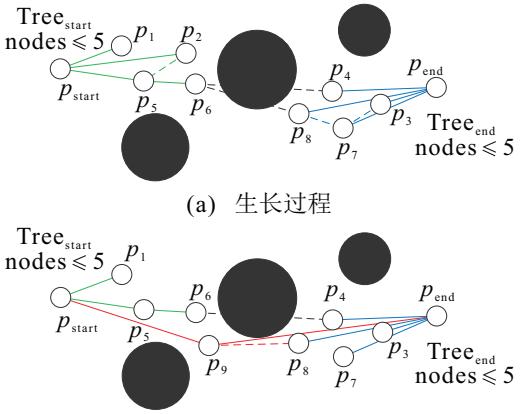


图 1　双向贪婪搜索示意图

由图1可见,加入双向贪婪搜索策略的RRT*FN算法较RRT*FN算法具有更加明显的指向性,且路径也随着迭代过程的增加趋于最优.

2.2　动态更新和路径修复

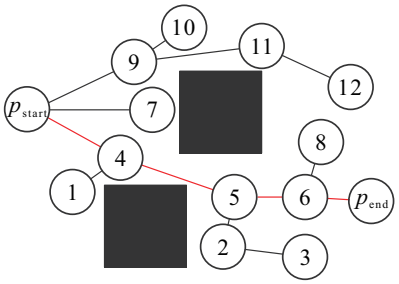
由于实际的机器人运行环境具有复杂性,主要体现于机器人在运行过程中可能存在导致原规划路径

损坏的情况,如障碍物空间随时间变化,地图空间中存在未能事先输入到障碍物列表的未知障碍物等. 同时,动态规划算法在更新环境信息时,采样点也会随着时间的推移逐渐增多,树的规模越大,动态算法实时更新时的负担越大,会拖慢运行速度. 针对上述问题,改进算法将动态更新以及路径修复思想引入RRT*FN算法,在每轮迭代过程中,算法会更新环境信息,基于原有节点进行规划更新,此外也会更新智能体位置信息,丢弃智能体经过的节点和附属枝干,以智能体所在位置为新的根节点,减小规划过程中树的规模. 在规划路径被障碍物破坏时,算法会丢弃被障碍物覆盖的树上节点,并利用空间中的完整节点对树进行重构,如图2所示,在修复路径的同时提高节点利用效率,路径修复的步骤如下.

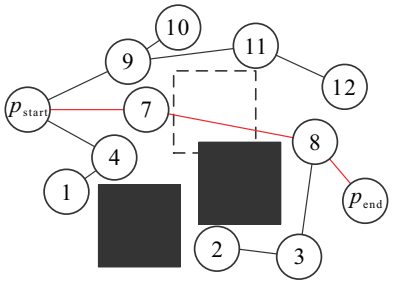
step 1: 当路径被障碍物破坏时,断开被覆盖节点与其他节点的连线,记录节点编号,并舍弃这些节点.

step 2: 选择距根节点最远的被舍节点,以其子节点为圆心,在半径为R的区域内为其重新寻找潜在父节点,使得路径代价最小的潜在父节点为新的父节点,并进行节点重连.

step 3: 继续以剩余的最远被舍节点的子节点,重复step 2,直至除根节点外的所有被舍节点均找到父节点. 则路径修复完成.



(a) 移动前



(a) 移动后

图 2 路径修复

2.3 算法完备性分析和固定节点数的选择

已验证RRT*算法的概率完备性和渐近最优性,这里给出所提出算法概率完备性和渐近最优性的证明. 首先,双向贪婪搜索策略不改变算法最终的概率完备性和渐近最优性,理由如下.

设状态空间$X\subseteq R_n$,2棵搜索树集合$T_1,T_2\subseteq X$,2棵树生长次数分别为iter 1和iter 2. 双向生长过程中若存在路径解σ,则足够次采样后,随机采样一定会使得

$$\begin{cases}\forall p\in\sigma, p_{\text{rand}}(x)=p,\ x\in(0,\text{iter1}),\\ \forall q\in\sigma, Q_{\text{rand}}(x)=q,\ x\in(0,\text{iter2}),\\ \sigma\subseteq T_1\bigcup T_2,\ p_{\text{rand}}\subseteq T_1,\ Q_{\text{rand}}\subseteq T_2.\end{cases}\tag{5}$$

即双向贪婪搜索的RRT*算法仍然具有概率完备性,在寻找到路径解σ后,将2棵树合并,则双树算法渐近最优性同单树RRT*算法. 改进算法规定了最大节点数,若固定节点数$n\to\infty$,则改进算法的概率完备性和渐近最优性与RRT*算法一致. 若固定节点数$n<$路径解中节点数m,则算法不具备概率完备性和渐近最优性. 针对$m\leqslant n$的情况进行分析,当最大节点数$n=m$且仍然未找到路径解σ时,算法会舍去随机无子叶子节点,继续进行采样和规划,并进行父节点重选和节点重连,相当于随机点p_{rand}的分布仍然为整个状态空间,在足够多次迭代后同样有

$$\forall p\in\sigma, p_{\text{rand}}(x)=p,\ x\in(0,\text{iter}).\tag{6}$$

若路径存在,则最终所得路径L一定为σ集合内元素的排列组合,即概率完备性满足. 同样地,由于节点重选和父节点重连,若存在$p\in\sigma, p_{\text{rand}}(x)=p$,则最终若选取该点作为节点重连的对象,则一定会使得路径代价减小,iter足够大时,仍然有

$$\begin{cases}L\{p_1,p_2,\dots,p_n\}=\sigma^*,\\ \forall p\in\sigma^*, p_{\text{rand}}(x)=p,\ x\in(0,\text{iter}).\end{cases}\tag{7}$$

即改进算法具备概率完备性和渐近最优性. 但是在$m\leqslant n<\infty$范围内,若固定节点数设定过小,则虽然算法具有概率完备性,但是获得路径解的概率几乎为0,若固定节点数值设置过大,则占用过大内存空间. 通常固定节点数会根据经验法选择,本文为使得固定节点数的选择更加科学,在固定状态空间和步长的情况下测试多组固定节点数-获得最优解的平均时间数据,并进行5次曲线拟合,选择拟合曲线区间最小值点处的固定节点数作为实验的固定节点数. 由于RRT算法选取采样点的随机性,该固定节点数的选择方式可能不是最优的,但是这种选择方法相对合理,避免了固定节点数错误选择对实验造成影响.

2.4 B-RRT*FND算法实现

B-RRT*FND算法的具体实现如下.

step 1: 初始化地图和搜索树Tree 1和Tree 2,分别将p_{start}和p_{end}添加至节点列表Tree 1、Tree 2中.

step 2: 以2棵树中规模较小的树作为生长目标(若树中节点数量一致则选择Tree 1),在空间中随机

取采样点 p_{rand}, 选择距离 p_{rand} 最近的节点 p_{nearest} 向 p_{rand} 以一定步长生长, 获得 p_{new}.

step 3: 进行父节点重选和节点重连.

step 4: 以 p_{new} 为父节点, 向对树距离最近的节点 p'_{nearest} 连续生长, 直至到达该点或遇到障碍物.

step 5: 执行一次生长过程后, 更新地图, 将智能体所在节点设置为根节点, 删除上一次智能体所在位置的节点及其枝干, 若 $P \bigcap X_{\text{obs}} \neq \varnothing, P \in \sigma$, 则删除被障碍物覆盖的节点以及连线与障碍物相交的节点, 断开这些点与周围节点的连接, 执行 step 3.

step 6: 比较 Tree 1 与 Tree 2 列表中的节点数目, 选择规模较小的为生长对象, 执行 step 2 ~ step 5.

step 7: 生长过程中若 2 棵树连接 (Tree 1 和 Tree 2 中存在两节点距离小于设定值 connectDis), 定义 Tree 1、Tree 2 连接处的节点为 $p_{\text{ctree 1}}$、$p_{\text{ctree 2}}$, 以 $p_{\text{ctree 1}}$ 作为根节点, 交换 Tree 2 中的节点父子关系, 并以 $p_{\text{ctree 1}}$ 作为 $p_{\text{ctree 2}}$ 的父节点, 将 2 棵树合并为树 Tree.

step 8: 以 p_{start} 作为起始点, p_{end} 作为目标点, 选择合并树 Tree 中代价最短的路径作为路径解.

step 9: 按 RRT*FN 算法的 step 1 ~ step 3 对路径进行渐近优化, 每次生长/重连完成后, 执行 step 5.

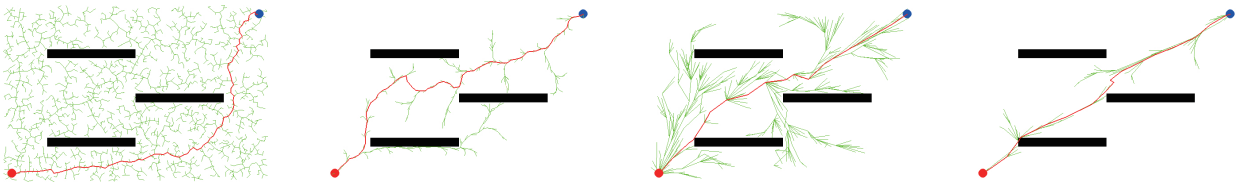
3　算法验证

为了验证所提出算法的有效性, 本文将 B-RRT*FND 算法与其他算法在 3 种地图下进行比较, 并评价其性能指标. 为方便仿真分析且兼顾合理性, 本文将障碍物看作膨胀后的规则图形, 将移动机器人视为点, 并控制除算法外的其他无关变量保持一致. 由于 RRT 类算法基于随机采样, 在仿真过程中存在偶然性, 每次测量结果可能由于采样点位置不同造成较大差异, 为排除偶然性带来的影响, 本文对每种情况进行 50 次独立实验, 整理结果进行对比分析.

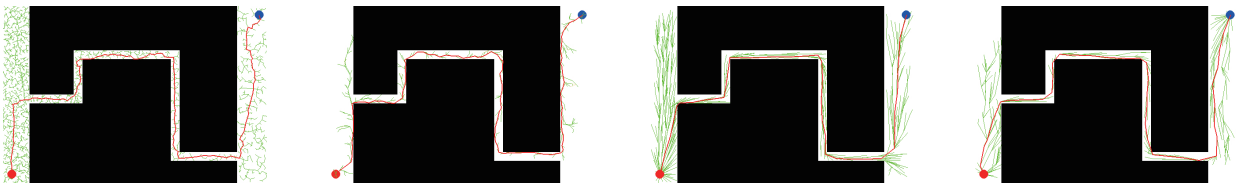
3.1　地图 1 下的仿真验证

本文设定地图 1 (600×400) 为普通障碍物地图, 用于验证算法在正常障碍物环境下的响应速度和路径指标. 本文设置左上角为坐标原点 [0, 0], 地图位于第 4 象限, 移动机器人的起点终点位置为 [20, 380]、[580, 20], 并分别用红色、蓝色圆表示. RRT、B-RRT*、RRT*FN 算法、B-RRT*FND 算法的路径规划结果如图 3(a) 所示, 红色线为所得路径解. 后 3 者为迭代算法首次获得路径解的规划结果. 可以看出, 单向算法在规划过程中出现过多冗余采样点, 而双向算法指向性更加明显, 为直观对比, 将 50 次实验所得平均路径解长度、迭代次数等列于表 1.

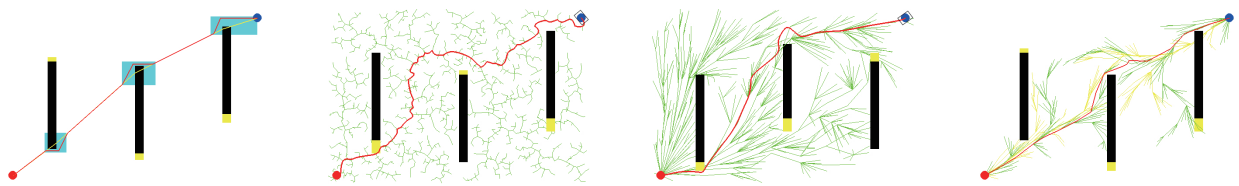
由表 1 可见, 双向算法在快速寻解方面具有一定优势, 由于改进算法使用了双向贪婪搜索策略, 整体路径代价进一步缩小. 因 B-RRT* 算法、RRT*FN 算法以及 B-RRT*FND 算法均属于最优求解算法, 本文将路径解的迭代过程路径解参数进行整理, 如图 4(a)



(a)　地图1 下各算法路径规划结果



(b)　地图2 下各算法路径规划结果



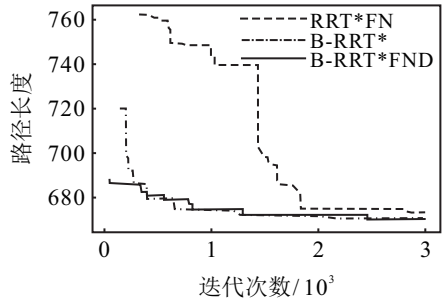
(c)　地图3 下各算法路径规划结果

图 3　3 种地图下各算法路径规划结果

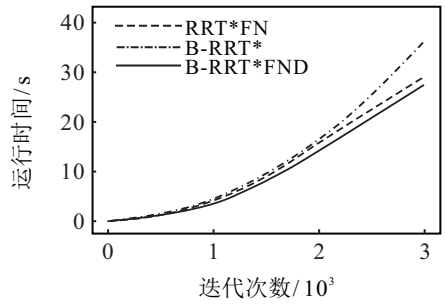
和图4(b)所示.

表 1 地图1中算法性能对比数据

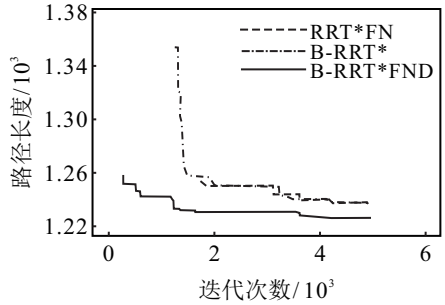
算法	平均路径长度	平均运行时间/s
RRT	820.662 3	41.596 5
B-RRT*	726.307 8	0.624 8
RRT*FN	754.156 7	2.188 8
B-RRT*FND	708.778 5	0.167 3



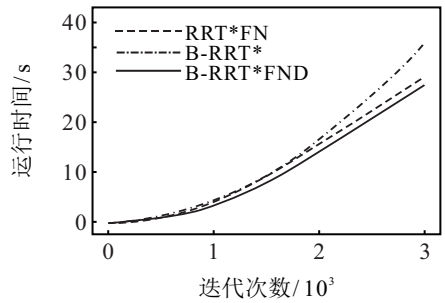
(a) 地图1中各算法路径长度-迭代次数关系



(b) 地图1中各算法时间-迭代次数关系



(c) 地图2中各算法路径长度-迭代次数关系



(d) 地图2中各算法时间-迭代次数关系

图 4 地图1和地图2下各算法性能对比分析

由图4(a)和图4(b)可见,B-RRT*FND算法的路径解长度和搜索时长优于其他两种算法. B-RRT*算法500次迭代后长度与B-RRT*FND算法基本持平,但是由于其没有固定节点数,运行时间随着迭代次数的增加呈凹增趋势(如图4(b)),1 500次迭代后,改进算法的运行时间与迭代次数近似呈线性关系. 3 000次迭代时,B-RRT*算法耗时36.117 s,B-RRT*FND为29.221 s.

3.2 地图2下的仿真验证

本文设置地图2(600×400)为窄缝地图,除地图设置外,其余设置与第3.1节相同,4种算法的路径规划结果如图3(b)所示,后3者为首次获得路径解的结果.

对于窄道地图,单向算法由于生长的盲目性,很难在窄道内的自由空间取到采样点,导致迭代次数和运行时间远高于双向算法,采样点多集中于左侧自由空间. 实验所得平均路径解长度和运行时间如表2所示.

表 2 地图2中算法性能指标对比

算法	平均路径长度	平均运行时间/s
RRT	1 383.031 7	145.188 1
B-RRT*	1 355.370 6	2.482 5
RRT*FN	1 239.629 2	4.919 8
B-RRT*FND	1 239.042 0	1.415 5

由于采样点选取在窄道内的概率降低,导致采样点数整体增加. RRT算法运行时间超过145 s,迭代次数超20 000次,对内存造成了极大的负担,而双向算法在运行时间上存在一定优势,为进一步对最优求解算法的性能进行对比,将3种最优算法迭代优化过程的结果整理,如图4(c)和图4(d)所示. 由图4(c)和图4(d)可见,B-RRT*FND算法获取路径的长度和迭代速度均优于B-RRT*、RRT*FN算法.

3.3 地图3下的仿真验证

为了进一步研究动态环境下的算法性能,本文设置了含有移动障碍物的动态地图(大小400×600),动态障碍物(黑色部分,大小20×200)沿y轴往复移动,运动范围为$y=20\sim580$. 本节引入D*(dynamic A*)、RRT-DWA算法、RRT*DWA算法作为对比算法,由于动态环境下路径解长度变化不具规律性,仅将智能体移动路径长度和全局路径规划时间整理,如表3所示,算法路径规划结果如图3(c)所示.

表 3 地图3中算法性能指标对比

算法	平均路径长度	规划时间(全局)/s
B-RRT*FND	759.658 0	0.284 4
D*	691.375 9	41.303 1
RRT-DWA	989.506 6	36.907 8
RRT*DWA	766.310 3	6.742 4

由图3(c)可见,图中算法均具有动态规划能力,可对运动的障碍物进行躲避,其中 D^* 算法所规划的平均路径较短,但是由于 D^* 算法为基于图的启发式搜索算法,其规划速度较慢,在地图3中平均规划时间超过40 s,RRT-DWA和RRT*DWA算法首先使用RRT和RRT*算法进行全局路径规划,再使用DWA算法在智能体移动过程中进行局部路径规划,以达到动态避障效果,但是易陷入局部最优,且所得路径平均长度以及全局先验路径规划时间均长于改进算法. 总体而言,改进算法具有较好的动态规划能力,但是易出现路径被障碍物破坏后,由于部分采样节点被舍去导致的规划路径暂时较差的情况,这种情况在保证路径完好的数次迭代后有所改观.

4 结　论

机器人的实时路径规划问题是智能移动机器人以及无人驾驶领域的研究重点,针对RRT及其改进算法存在盲目性采样造成的路径曲折,难以在狭窄环境下快速求解以及父节点重选和节点重连优化在树的规模扩大时增大计算量、拖慢运行速度的问题,本文提出了B-RRT*FND算法,该算法基于RRT*FN算法,增加了双向贪婪搜索、动态更新和路径修复的思想,通过与各对比算法在3种地图下的对比分析,结果表明,所提出算法在规划速度、路径解长度以及动态性能方面具有较好的性能,验证了所提出算法的有效性.

参考文献(References)

[1] Wu X J, Xu L, Zhen R, et al. Biased sampling potentially guided intelligent bidirectional RRT algorithm for UAV path planning in 3D environment[J]. Mathematical Problems in Engineering, 2019, 2019: 5157403.

[2] Lyu D S, Chen Z W, Cai Z S, et al. Robot path planning by leveraging the graph-encoded Floyd algorithm[J]. Future Generation Computer Systems, 2021, 122: 204-208.

[3] Peng J H, Li I H, Chien Y H, et al. Multi-robot path planning based on improved D lite algorithm[C]. IEEE the 12th International Conference on Networking, Sensing and Control. Taipei, 2015: 350-353.

[4] Maw A A, Tyan M, Nguyen T A, et al. iADA*-RL: Anytime graph-based path planning with deep reinforcement learning for an autonomous UAV[J]. Applied Sciences, 2021, 11(9): 3948.

[5] 刘军, 冯硕, 任建华. 移动机器人路径动态规划有向 D^* 算法[J]. 浙江大学学报: 工学版, 2020, 54(2): 291-300.
(Liu J, Feng S, Ren J H. Directed D^* algorithm for dynamic path planning of mobile robots[J]. Journal of Zhejiang University: Engineering Science, 2020, 54(2): 291-300.)

[6] Muhammad A, Ali M A H, Shanono I H. A novel algorithm for mobile robot path planning[C]. IEEE the 11th IEEE Symposium on Computer Applications & Industrial Electronics. Penang, 2021: 48-52.

[7] Mashayekhi R, Idris M Y I, Anisi M H, et al. Hybrid RRT: A semi-dual-tree RRT-based motion planner[J]. IEEE Access, 2020, 8: 18658-18668.

[8] 谭建豪, 潘豹, 王耀南, 等. 基于改进RRT*FN算法的机器人路径规划[J]. 控制与决策, 2021, 36(8): 1834-1840.
(Tan J H, Pan B, Wang Y N, et al. Robot path planning based on improved RRT*FN algorithm[J]. Control and Decision, 2021, 36(8): 1834-1840.)

[9] Jeong I B, Lee S J, Kim J H. Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate[J]. Expert Systems with Applications, 2019, 123: 82-90.

[10] Gammell J D, Srinivasa S S, Barfoot T D. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic[J]. IEEE/RSJ International Conference on Intelligent Robots and Systems. Chicago, 2014: 2997-3004.

[11] Taheri E, Ferdowsi M H, Danesh M. Fuzzy greedy RRT path planning algorithm in a complex configuration space[J]. International Journal of Control, Automation and Systems, 2018, 16(6): 3026-3035.

[12] 魏武, 韩进, 李艳杰, 等. 基于双树Quick-RRT算法的移动机器人路径规划[J]. 华南理工大学学报: 自然科学版, 2021, 49(7): 51-58.
(Wei W, Han J, Li Y J, et al. Path planning of mobile robots based on dual-tree quick-RRT algorithm[J]. Journal of South China University of Technology: Natural Science Edition, 2021, 49(7): 51-58.)

[13] Spanogianopoulos S, Sirlantzis K. Non-holonomic path planning of car-like robot using RRT FN[C]. The 12th International Conference on Ubiquitous Robots and Ambient Intelligence. Goyangi, 2015: 53-57.

[14] Mashayekhi R, Idris M Y I, Anisi M H, et al. Informed RRT*-connect: An asymptotically optimal single-query path planning method[J]. IEEE Access, 2020, 8: 19842-19852.

[15] Salzman O, Halperin D. Asymptotically near-optimal RRT for fast, high-quality motion planning[J]. IEEE Transactions on Robotics, 2016, 32(3): 473-483.

[16] Zhang X, Lütteke F, Ziegler C, et al. Self-learning RRT* algorithm for mobile robot motion planning in complex environments[C]. Intelligent Autonomous System 13. Berlin, 2015: 57-69.

作者简介

张腾龙(2000−), 男, 硕士生, 从事智能导航、自主系统等研究, E-mail: 1678144012@qq.com;

李擎(1964−), 女, 教授, 博士, 从事智能导航、自主系统等研究, E-mail: liqing@bistu.edu.cn.