基于改进 RRT-Connect 算法的机械臂路径规划

韩 康 程卫东
(北京交通大学机械与电子控制工程学院 北京 100044)

摘 要 针对 RRT* 算法速度较慢问题,提出一种快速收敛至最优路径的最优双向快速扩展随机树(Optimal Bidirectional Rapidly-exploring Random Trees,Obi-RRT)算法。Obi-RRT 使用改进的 RRT-Connect 算法快速得到较低成本路径,通过路径修剪得到关键点,围绕关键点提出三种采样空间并进行采样,通过不断更新关键点从而得到最优或接近最优的路径。平面和机械臂关节空间下的仿真实验表明,Obi-RRT 算法运行时间仅为 RRT* 算法的十分之一,并且路径成本更低。

关键词 改进 RRT-Connect 机械臂 路径规划 最优路径

中图分类号 TP3 TP242.2 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2022.03.042

PATH PLANNING OF ROBOT ARM BASED ON IMPROVED RRT ALGORITHM

Han Kang Cheng Weidong
(*School of Mechanical, Electronic and Control Engineering, Beijing Jiaotong University, Beijing 100044, China*)

Abstract Aiming at the problem of the low speed of RRT* algorithm, this paper proposes an Obi-RRT algorithm that quickly converges to the optimal path. Obi-RRT used the improved RRT-Connect algorithm to quickly obtain a lower cost path, and obtained key points through path pruning. Three sampling spaces were proposed around the key points for sampling, and the key points were continuously updated to obtain an optimal or near optimal path. The simulation experiments in the plane and the joint space of the robot arm show that the running time of the Obi-RRT algorithm is only one tenth of the RRT* algorithm, and the path cost is lower.

Keywords Improved RRT-Connect Robot arm Path planning Optimal path

0 引 言

路径规划是指在状态空间中规划一条包含初始状态和目标状态的无碰撞路径[1]。近年来,由于实际生产活动对智能自动化的高度需求,路径规划被广泛地应用到自动驾驶汽车[2]、无人机航迹规划[3]和机器人路径规划[4]等多个领域中。多数应用场景下,对快速获得低成本路径都有需求。机械臂是一种常应用于工业生产的机器人,由于其状态空间维度高,一些传统路径规划方法无法在短时间内得到解决方案[5],目前主要使用的方法是快速扩展随机树[6](Rapid-exploration Random Tree,RRT)系列算法。

RRT 是由 Lavalle 等提出的基于采样的路径规划算法。该算法理论上可以解决任意维度空间的路径规划问题,但在高维空间中速度比较慢。因此,其又提出改进算法 RRT-Connect[1],通过双树扩展提高算法速度。为解决 RRT 算法易产生高成本路径的问题,Karaman 等[7]提出最优路径规划算法 RRT*。该算法通过在每次迭代中寻找最优父节点以及节点的重新布线来降低路径成本,从而使路径收敛至最优,但收敛速度较慢。许多学者为加速收敛速度不断提出新的改进算法。Jordan 等[8]提出的 B-RRT* 算法是 RRT* 的双树版本,使用双向搜索和局部偏置优化的方法来加速收敛速度;Islam 等[9]提出的 RRT*-Smart 算法使用 RRT* 生成初始路径后,通过删除冗余节点和偏置采样,改善了 RRT* 收敛缓慢问题;Gammell 等[10]提出 Informed RRT* 算法,采用直接限制采样范围来提高整体收敛速度,在此基础上 Burget 等[11]结合 B-RRT* 提出双树版本 BI-RRT* 算法;Zhang 等[12]提出的 Self-

收稿日期:2020-01-01。北京市自然科学基金项目(3192025)。**韩康**,硕士生,主研领域:路径规划。**程卫东**,教授。

learning RRT* 采用基于自学习策略和混合偏差抽样方案来提高规划效率;陈晋音等设计了 EB-RRT*[13] 和贪心 MB-RRT* 算法[14],分别利用地图栅格分区和贪心策略来提高算法速度。这些 RRT* 的改进算法在实现路径规划时仍存在以下两个问题:

(1) 在近邻节点查询过程中,大量的节点遍历操作严重影响算法速度;

(2) 局部偏置采样注重收敛速度而牺牲随机探索特性,可能会丢失更优解[15],导致最终优化路径的质量往往取决于初始路径的质量。

针对 RRT 系列算法运行时间与路径成本之间的平衡问题,本文提出 Obi-RRT 进行解决。Obi-RRT 通过添加目标偏置并且拒绝可能产生高成本路径的采样点,快速得到较低成本的初始路径,进而进行路径修剪得到路径关键点;在关键点周围,定义三种采样空间,不断进行随机采样,当使用新点得到的路径成本低于原始成本并且路径通过碰撞检测时,替换旧点,从而使路径不断收敛至最优。由于省去了 RRT* 系列算法近邻节点查询的过程,Obi-RRT 能在更短的时间内产生低成本路径。此外,因为路径中节点很少,更有利于机械臂控制。

1　Obi-RRT 算法

1.1　问题描述与定义

文中所研究的状态空间分别为二维平面和六维工业机械臂关节空间。平面空间的状态节点定义为在直角坐标系下点的坐标所组成的向量,记为 $q_2(x,y)$。节点之间成本定义为坐标系下两点的直线距离。机械臂关节空间的状态节点定义为机械臂某个姿态下关节角所组成的六维向量 $\boldsymbol{q}_6(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6)$,节点之间成本定义为它们关节角向量的欧拉距离。

给定状态空间 X,障碍物区域和无障碍区域分别定义为 X_{obs} 和 X_{free}。起始状态 $q_{\text{init}} \in X_{\text{free}}$,目标状态 $q_{\text{goal}} \in X_{\text{free}}$。Obi-RRT 算法的目的是找到一条路径 s,使 q_{init},$q_{\text{goal}} \in s$ 和 $s \in X_{\text{free}}$ 并且使 s 的成本达到最小。为此,算法构建了两棵随机树 $\boldsymbol{T}_{\text{a}}=(V_{\text{a}},E_{\text{a}})$,$\boldsymbol{T}_{\text{b}}=(V_{\text{b}},E_{\text{b}})$,其中 $V_{\text{a}},V_{\text{b}} \in X_{\text{free}}$ 是树中状态节点集合,$E_{\text{a}},E_{\text{b}} \in X_{\text{free}}$ 是连接两个状态节点边的集合。

1.2　算法基本流程

Obi-RRT 算法分为路径搜索、路径修剪和路径优化三个阶段。路径搜索阶段使用改进的 RRT-Connect 算法快速得到一条较低成本的路径。再经过路径修剪阶段,连接了路径中可直接相连的节点来初步降低路径成本,并大幅度降低路径中节点数量,得到路径关键点集 *keypoints*。在路径优化阶段,通过在关键点周围采样不断更新关键点,产生更低成本的路径,达到快速收敛的目的。算法主体过程如算法 1 所示。

算法 1　Obi-RRT 算法

Obi-RRT(q_{init}, q_{goal})
1　$V_{\text{a}}=\{q_{\text{init}}\}$; $V_{\text{b}}=\{q_{\text{goal}}\}$; $E_{\text{a}}=\varnothing$; $E_{\text{b}}=\varnothing$; $i=0$;
2　**while** ($i<N$) **do**
3　　q_{rand} = **SmartSample**($\boldsymbol{T}_{\text{a}}$, $\boldsymbol{T}_{\text{b}}$); i++;
4　　**if not** (**Extend**($\boldsymbol{T}_{\text{a}}$, q_{rand}) = *Trapped*) **then**
5　　**if** (**Connect**($\boldsymbol{T}_{\text{b}}$, q_{new}) = *Reached*) **then**
6　　**return** *path* = **Path**($\boldsymbol{T}_{\text{a}}$, $\boldsymbol{T}_{\text{b}}$);
7　　**Swap**($\boldsymbol{T}_{\text{a}}$, $\boldsymbol{T}_{\text{b}}$);
8　**return** *Failure*;
9　*keypoints* = **GetKeyPointsPWTBX**](**path**);
10　$\textbf{length}_{\text{old}}$ = **PathLength**(*keypoints*);
11　(*keypoints*, *length*) = **PathOptimization**(*keypoints*);
12　**return** *path* = *keypoints*;

1.3　路径搜索阶段

Obi-RRT 算法同 RRT-Connect 算法一样,使用两棵树 $\boldsymbol{T}_{\text{a}}$ 和 $\boldsymbol{T}_{\text{b}}$ 分别从起始状态和目标状态开始,向对方生长。不同的是在新算法中使用目标偏置采样和启发式采样来加速算法。

算法存在两种目标偏向,一种是以一定概率采用 q_{init} 或 q_{goal} 作为采样点,一种是以一定概率采用对方最新扩展的状态节点 $q_{\text{pre\_node}}$ 作为采样点。第一种目标偏置引导两棵树偏向起点和终点生长,第二种目标偏置也引导两棵树偏向对方最新扩展节点生长,通过两种偏置结合,加速了两棵随机树的连接过程。

同时,算法还使用了启发式采样来保证获取到较低成本的初始路径。假设当前为树 $\boldsymbol{T}_{\text{a}}$ 生长,采样点为 q_{rand},其与 $\boldsymbol{T}_{\text{a}}$ 距离最近的节点为 $q_{\text{nst\_a}}$,与 $\boldsymbol{T}_{b}$ 距离最近的节点为 $q_{\text{nst\_b}}$,计算路径预期成本 C_{ex}:

$$C_{\text{ex}}=q_{\text{nst\_a}}.\,cost+q_{\text{nst\_b}}.\,cost+Cost(q_{\text{rand}},q_{\text{nst\_a}})+Cost(q_{\text{rand}},q_{\text{nst\_b}}) \tag{1}$$

当 $C_{\text{ex}}>C_{\max}$ 时,q_{rand} 被拒绝,并重新进行采样。拒绝低质量采样点可以减少不必要的扩展过程,提高了算法速度。算法采样过程伪代码如算法 2 所示。

算法 2　采样过程

SmartSample ($\boldsymbol{T}_{\text{a}}$, $\boldsymbol{T}_{\text{b}}$)
1　**while** (q_{rand} = null) **do**
2　　p = *Random* (0, 1);
3　　**if** ($p<p_{\text{goal}}$) **return** $q_{\text{rand}}=q_{\text{goal\_a}}$; **break**;
4　　**if** ($p>p_{\text{goal}}$) **return** $q_{\text{rand}}=q_{\text{pre\_node}}$; **break**;
5　　q_{rand} = **Random**();
6　　$q_{\text{nst\_a}}$ = **Nearest**($\boldsymbol{T}_{\text{a}}$, q_{rand});

$q_{nst\_b} = \textbf{Nearest}(T_b, q_{rand})$;
$h = q_{nst\_a}.cost + q_{nst\_b}.cost +$
  $\textbf{Cost}(q_{nst\_a}, q_{rand}) + \textbf{Cost}(q_{nst\_b}, q_{rand})$;
if $(h > h_{pre\_set})$ $q_{rand} = \text{null}$;
else return q_{rand}

1.4 路径修剪阶段

当算法成功得到一条无碰撞路径时,便进入路径修剪阶段。该阶段以三角不等式原理来优化路径。如图 1 所示,c 边的长度总是小于 a 边和 b 边的总和。

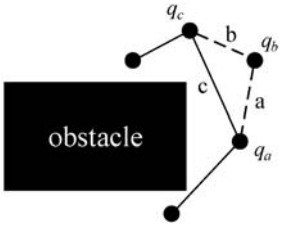

图 1 基于三角不等式优化路径原理

路径修剪将连接路径节点中可以直接相连的节点,删掉中间节点,得到路径关键点集 *keypoints*。该点集中任意两个非相邻节点不可以直接相连。由此点集构成的路径成本比原路径更低。路径修剪过程的伪代码如算法 3 所示。

算法 3 路径修剪过程

GetKeyPoints (*path*)
$q_1 = q_{goal}$; $q_2 = q_1.parent.parent$; $keypoints = \varnothing$;
while $(q_2 != null)$ **do**
  if ObstacleFree (q_1, q_2) $q_2 = q_2.parent$;
  else $keypoints = \{q_2.child, keypoints\}$
    $q_1 = q_2.child$; $q_2 = q_1.parent.parent$;
$keypoints = \{q_{init}, keypoints, q_{goal}\}$
for $(q_1 = q_{goal}; q_1 != q_{init}.child.child; q_1 = q_1.parent)$
  for $(q_2 = q_1.parent.parent; q_1 != q_{init}; q_2 = q_2.parent)$
    if ObstacleFree (q_1, q_2)
      $keypoints = \textbf{Remove}(q_2.child)$;
return *keypoints*;

keypoints 中节点数量代表着此条路径中需要转折的次数,也一定程度上反映了该状态空间的复杂程度。路径关键点越多,状态空间中障碍物分布越复杂。

1.5 路径优化阶段

路径优化阶段中将在关键点周围的特定范围空间内产生一定数量的样本,并假设新点替代原关键点,进行路径成本计算与碰撞检测。如果使用新点可以产生更低成本的路径并且通过了碰撞检测则使用新点替换原始关键点。

本文提出三种采样空间,图 2 以二维平面空间进行说明。如图 2(a)所示,一个关键点集合中,随机选择一个关键点 *b* 作为采样基准,节点 *a* 为其父节点,节点 *c* 为其子节点。采样将围绕这三个连续的节点进行。第一种采样范围是由这三个连续节点组成的最小包围盒,称作 A 类空间,如图 2(b)所示。第二种采样范围是由以节点 *a* 和节点 *c* 的中心节点 *o* 为中心,*ob* 为半径的空间球体,称作 B 类空间,如图 2(c)所示。第三种采样范围是由节点 *b* 为中心,半径为 *r* 的空间球体中进行采样,称作 C 类空间,如图 2(d)所示。

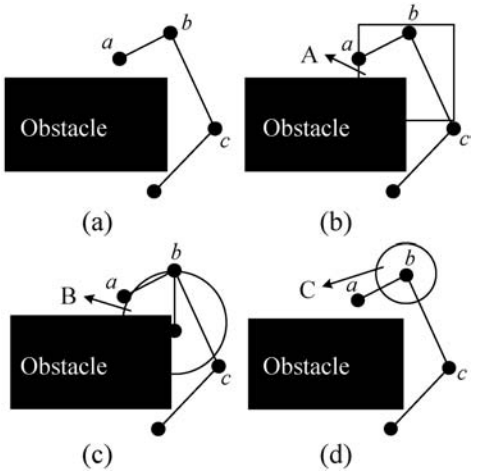

图 2 三种采样空间二维示意图

路径关键点一定意义上代表着路径的转折点,其给出了关于障碍物顶点或者边缘的信息[9]。局部重采样希望在已有路径关键点周围尽可能生成靠近障碍的节点,进行关键点更新,从而实现路径快速收敛于最优。路径优化过程伪代码如算法 4 所示。

算法 4 路径优化过程

PathOptimization (*keypoints*)
while $(i < N)$ **do**
  $q_n = \textbf{RandomSelectionNode}(keypoints)$;
  $q_{np} = q_n.parent$; $q_{nc} = q_n.child$;
  $p = \textbf{Random}(0, 1)$;
  if $(p < p_a)$ $q_{rand} = \textbf{ASample}(q_n, q_{np}, q_{nc})$; **break**;
  if $(p > p_b)$ $q_{rand} = \textbf{BSample}(q_n, q_{np}, q_{nc})$; **break**;
  $q_{rand} = \textbf{CSample}(q_n, r)$;
  $length_{old} = \textbf{Cost}(q_n, q_{np}) + \textbf{Cost}(q_n, q_{nc})$
  $length_{new} = \textbf{Cost}(q_{rand}, q_{np}) + \textbf{Cost}(q_{rand}, q_{nc})$
  if $(length_{new} < length_{old})$
    if $(\textbf{ObstacleFree}(q_{np}, q_{rand}, q_{nc}))$
      $q_n = q_{rand}$;
      $length = length - length_{old} + length_{new}$
  if (**TimeOut** || **IterationOverflow**) **break**;
return $(keypoints, length)$

采样范围的大小决定了优化的效率。对于上述所提出的三种采样空间,采样范围大小一般是 A 大于 C,B 大于 C。其中,A 类和 B 类采样空间的大小与连续三个节点的分布有关,其大小无法直接比较,不过随着节点的不断更新,A、B 两类采样空间的大小在不断减

小。C 类采样空间需要人为指定半径 r。其设置得过大则失去了降低采样范围的意义,过小又会导致采样次数的增加,两者都不同程度上影响优化速度,其取值与状态空间障碍物的分布有关。这也是引入 A 类和 B 类采样空间的原因,它们的大小完全取决于节点本身。

一般来说,只使用 C 类空间算法会更快,但会产生路径只收敛至局部最优的问题,丢失了全局更优解。如图 3(a)的情况,初始路径并不理想,如果此时单独使用 C 类采样空间,则最终只能得到图 3(a)中 $a-b'-c$ 的结果。而最优路径是 $a-b_{\text{optimal}}-c$。引入 A 类和 B 类采样空间可以改善这个问题。如图 3(b)和图 3(c)所示,在这两种采样空间中始终可以获得靠近 b_{optimal} 的点,从而使路径收敛向最优。在图示情况下,一开始 A 类采样空间效率要高于 B 类。但这并不是固定不变的,在路径较为平缓的情况下,B 类采样空间效率要高于 A 类。A 类和 B 类采样均有一定几率将质量较差的路径修正。

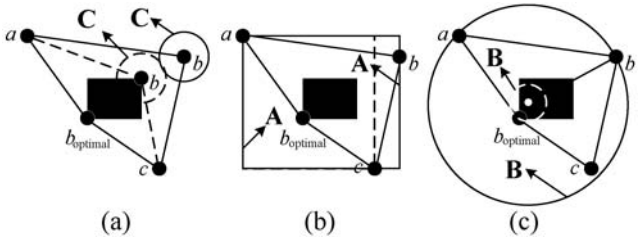

图 3　使用不同采样方法得到的最优结果

2　状态空间模型

仿真实验将在平面和机械臂关节两种状态空间中进行。随着状态空间维度增加,计算复杂度依次增加。测试的算法为 RRT* 和 Obi-RRT。

平面空间的仿真采用 MATLAB 2018 仿真平台,图 4 是两种不同的平面空间,大小均为 500×500。其中,深色部分为障碍物,白色部分为自由空间。每个空间的起点和终点均固定。

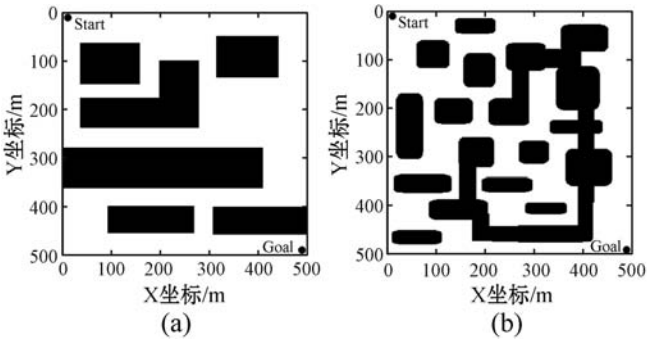

图 4　平面仿真环境

六轴机械臂关节空间仿真使用 OpenRAVE 仿真平台。图 5 展示了机器人关节空间仿真环境的三维场景,其中:(a)为机器人起始位置,(b)为机器人目标位置。这个仿真实验模拟机械臂抓取深箱物体的场景。

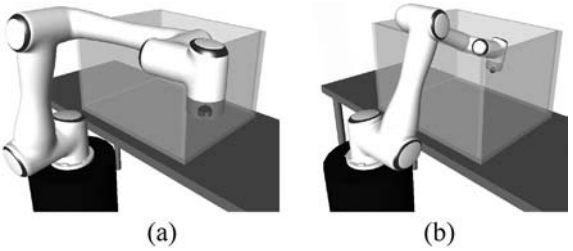

图 5　六轴机械臂关节空间仿真环境

3　仿真与实验验证

3.1　二维平面空间实验结果

首先对 RRT* 与 Obi-RRT 算法进行了相同迭代次数的对比实验。图 6(a)和图 6(b)分别是 RRT* 迭代 2 000 次和 4 000 次的结果,图 6(c)和图 6(d)分别是 Obi-RRT 迭代 2 000 次和 4 000 次的结果。

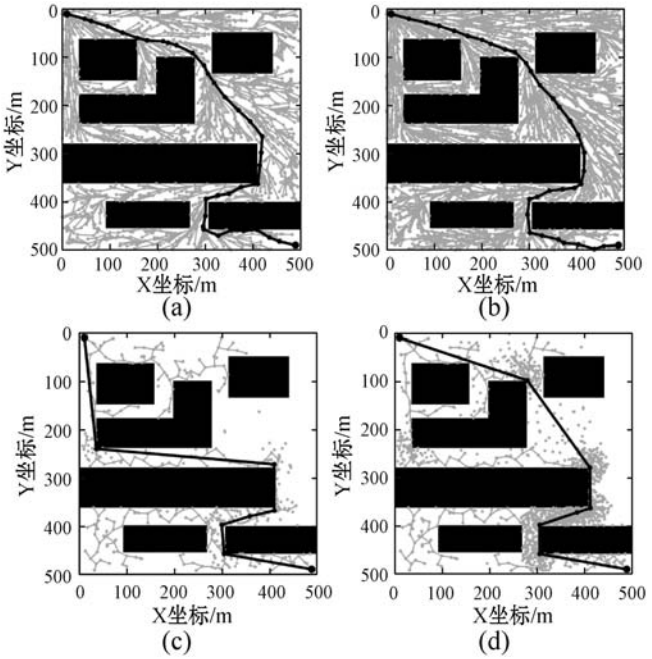

图 6　RRT* 与 Obi-RRT 在相同迭代次数下仿真结果

在迭代 2 000 次时,RRT* 得到的初始路径成本更低。这是由于 Obi-RRT 得到了一条较差的初始路径导致的。当迭代次数到达 4 000 时,Obi-RRT 算法所得到的路径成本低于 RRT*。值得注意的是,Obi-RRT 通过 A 类和 B 类采样,将质量较差的初始路径成功修正。RRT* 和 Obi-RRT 两种算法执行完 4 000 步迭代分别需要 10.56 s 和 0.423 s,得到的路径长度分别为 969.77 m 和 951.35 m。可以看出,Obi-RRT 可以在更短的时间内获得低成本路径。

对图 4(b)所示的平面状态空间分别使用两种算法进行测试,实验结果如图 7 所示。

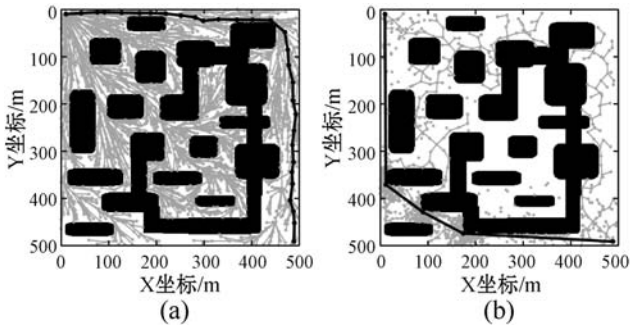

图 7 两种算法在复杂平面空间中的实验结果

为了证明算法的一般性,对每个空间使用两种算法分别运行 100 次求算法运行耗时与路径成本的平局值,实验统计结果如图 8 所示。在所有的实验中,仅 RRT* 出现四次路径搜索失败的情况。图 8(a)显示,Obi-RRT 算法运行时长平均不超过 0.5 s,仅为 RRT* 算法的十分之一。图 8(b)显示,Obi-RRT 算法规划的路径平均成本更低。

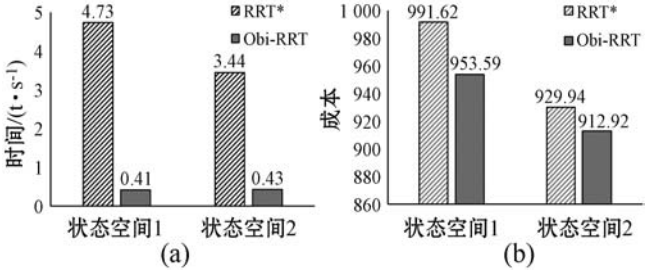

图 8 两种算法在平面状态空间的实验结果

3.2 六维机械臂关节空间实验结果

对图 5 所示的机械臂关节空间分别使用两种算法进行路径规划。两算法搜索到的路径结果如图 9 所示。

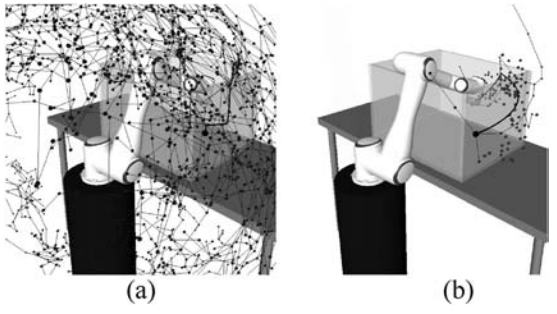

图 9 两种算法在六轴机械臂关节空间中搜索到的路径结果

为了证明算法的一般性,在同一个状态空间中,每个算法分别运行 100 次,对计算耗时与路径成本进行统计,实验统计结果如图 10 所示。

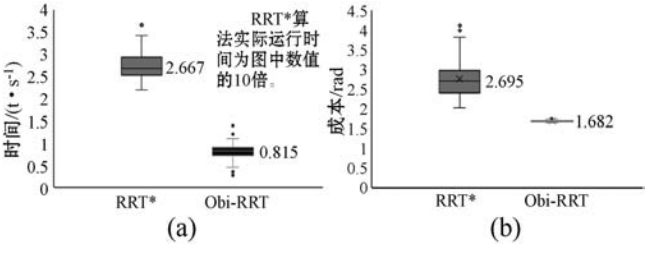

图 10 两种算法在六轴机械臂关节状态空间的实验结果

RRT* 算法运行时间长,平均为 26.67 s。Obi-RRT 算法平均运行时长为 0.815 s,较 RRT* 有很大改善。在路径成本方面,从图 10(b)可以看出,RRT* 算法所规划出的路径成本波动性仍比较大,其路径平均成本为 2.759 rad。Obi-RRT 算法所规划出的路径成本波动性很小,其路径的平均成本为 1.684 rad,比 RRT* 所规划的路径平均成本低 39%。此外,在 100 次迭代中,RRT* 算法有 31 次没能找到可行路径。这主要是由于在高维环境下,采样的随机性太强,导致树没有及时探索到目标状态附近导致的。

4 结 语

针对 RRT 系列算法规划的路径成本较高或收敛速度慢的不足,本文提出了一种快速收敛至最优路径的 Obi-RRT 算法。算法做出了以下几点改进:

(1) 路径搜索阶段,采用智能采样,加速搜索过程并拒绝生长那些可能产生高成本路径的采样点以保证算法得到较低成本的初始路径;

(2) 路径修剪阶段,对路径节点进行修剪得到少量路径关键点;

(3) 路径优化阶段,在关键点周围进行重新采样,有效减小采样范围,提高算法收敛速度。

仿真实验表明所提算法是有效的。Obi-RRT 可以在短时间内得到最优或近似最优的路径,符合实际工程使用的需求。通过测试结果可以看到,在文中测试的所有维度的状态空间中,Obi-RRT 算法运行时间仅是 RRT* 的十分之一,并且所产生的路径成本波动很小。随着维度增加,Obi-RRT 的优势越明显。

参 考 文 献

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

[2] Lan X, Cairano S D. Continuous curvature path planning for semi-autonomous vehicle maneuvers using RRT*[C]//2015 European Control Conference (ECC). IEEE, 2015: 2360 - 2365.

[3] 尹高扬,周绍磊,吴青坡. 基于改进 RRT 算法的无人机航迹规划[J]. 电子学报,2017,45(7):1764 - 1769.

[4] 陈敏,李笑,武交峰. 基于改进 RRT 算法的差动机器人路径规划[J]. 计算机应用与软件,2019,36(9):276 - 280.

[5] 由弘扬,贺帅,刘宏伟,等. 基于 bi-RRT 算法的九自由度机械臂路径规划[J]. 计算机仿真,2019,36(7):308 - 313.

[6] Lavalle S M, Kuffner J. Rapidly-exploring random trees: Progress and prospects[C]//4th International Workshop on Algorithmic Foundations of Robotics. Wellesley, 2000: 293 – 308.

[7] Karaman S, Frazzoli E. Incremental sampling-based algorithms for optimal motion planning[C]//The 2010 Robotics: Science and Systems Conference, 2010.

[8] Jordan M, Perez A. Optimal bidirectional rapidly-exploring ran-dom trees: MIT-CSAIL-TR-2013-021[R]. CSAIL MIT, 2013.

[9] Islam F, Nasir J, Malik U, et al. RRT*-Smart: Rapid convergence implementation of RRT* towards optimal solution [C]//2012 IEEE International Conference on Mechatronics and Automation. IEEE, 2012:1651 – 1656.

[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[C]//IEEE/RSJ Inter-national Conference on Intelligent Robots & Systems. IEEE, 2014:2997 – 3004.

[11] Burget F, Bennewitz M, Burgard W. BI-RRT*: An efficient sampling-based path planning framework for taskconstrained mobile manipulation [C]//2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2016:3714 – 3721.

[12] Zhang X, Felix L, Ziegler C, et al. Self-learning RRT* algorithm for mobile robot motion planning in complex environments[M]//Intelligent Autonomous Systems 13, 2016:57 – 69.

[13] 陈晋音,李玉玮,杜文耀. 基于 EB-RRT* 的无人机航迹规划算法研究[J]. 计算机科学,2017,44(S2):72 – 79.

[14] 陈晋音,胡可科,李玉玮. 基于 MB-RRT* 的无人机多点航迹规划算法研究[J]. 计算机科学, 2018, 45(S1):98 – 103.

[15] 陈秋莲,蒋环宇,郑以君. 机器人路径规划的快速扩展随机树算法综述[J]. 计算机工程与应用,2019,55(16):10 – 17.

(上接第 192 页)

4　结　语

针对于单幅图像的去雾,本文提出一种优化算法,经过大量的实验结果证实,此算法无论在主观分析方面或是在客观分析方面,去雾效果都优于之前的算法,体现出了其去雾的优势。但是,优势是有的,缺点也存在,本文算法的去雾相比于其他算法所需时间要长一些,如何在提升去雾效果的同时压缩时间也是接下来的重要目标与工作。

参　考　文　献

[1] Xu Y, Wen J, Fei L, et al. Review of video and image defogging algorithms and related studies on imagerestoration and enhancement[J]. IEEE Access, 2017, 4:165 – 188.

[2] Seow M J, Asari V K. Ratio rule and homomorphic filter for enhancement of digital colour image[J]. Neurocomputing, 2006,69(7 – 9):954 – 958.

[3] Wang W, Yuan X. Recent advances in image dehazing[J]. IEEE/CAA Journal of Automatica Sinica, 2017, 4(3): 24 – 50.

[4] He L, Zhao J, Zheng N, et al. Haze removal using the difference-structure-preservation prior[J]. IEEE Transactions on Image Processing, 2017, 26(3):1063 – 1075.

[5] Zhang C, Yang Y. Single image dehazing algorithm based on fusion and gaussian weighted dark channel[J]. Guangzi Xuebao/Acta Photonica Sinica, 2019, 48(1):110002.

[6] Russo F. An image enhancement technique combining sharpening and noise reduction[J]. IEEE Transactions on Instrumentation and Measurement, 2002, 51(4):824 – 828.

[7] Hu W W, Wang R G, Fang S, et al. Retinex algorithm for image enhancement based on bilateral filtering[J]. Journal of Engineering Graphics,2010,31(2):104 – 109.

[8] Hao W, He M, Ge H, et al. Retinex-Like method for image enhancement in poor visibility conditions[J]. Procedia Engineering,2011,15:2798 – 2803.

[9] Kopf J, Neubert B, Chen B, et al. Deep photo: Model-Based photograph enhancement and viewing[J]. ACM Transactions on Graphics,2008,27(5):116.

[10] Narasimhan S G, Nayar S K. Contrast restoration of weather degraded images[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003, 25(6):713 – 724.

[11] He K, Sun J, Tang X. Guided image filtering[J]. IEEE Transactions on Software Engineering, 2013, 35(6):1397 – 1409.

[12] Tarel J P, Hautière N. Fast visibility restoration from a single color or gray level image[C]//2009 IEEE 12th International Conference on Computer Vision,2009.

[13] Meng G, Wang Y, Duan J, et al. Efficient image dehazing with boundary constraint and contextual regularization[C]//2013 IEEE International Conference on Computer Vision, 2014.

[14] Fattal R. Single image dehazing[J]. ACM Transactions on Graphics,2008,27(3):1 – 9.

[15] Ren W, Liu S, Zhang H, et al. Single image dehazing via multi-scale convolutional neural networks [C]//European Conference on Computer Vision,2016.