

# 一种复杂场景下的路径规划问题解决方法

郭书杰, 田 华, 王 伟

(大连东软信息学院 智能与电子工程学院, 辽宁 大连 116023)

**摘 要:** 路径规划在诸多领域都有重要的应用价值, 研究者们提出了多种解决路径规划问题的算法。但复杂场景下的路径规划问题并未得到很好的解决, 特别是窄通道和 Z 字形场景下的路径规划问题。为了解决这一问题, 提出了一种关键点路径规划方法。该方法通过关键点的引入, 将整个路径规划切分成多个较短的子路径规划, 从而降低问题的规模, 提高规划效率; 同时, 这些关键点将随机路径搜索转化为启发式搜索, 从而大幅提高了在窄通道和 Z 字形下的路径规划效率。算法首先确定绕过障碍物的关键点, 然后使用这些关键点引导规划过程, 初步规划出一条路径。最后通过路径压缩来优化路径, 降低路径长度。实验结果表明, 该方法不仅能够高效地完成窄通道场和 Z 字形场景中的路径规划, 在其他常见场景的应用中也有不俗的表现。

**关键词:** 路径规划; 路径压缩; 关键点; 子路径; RRT-connect

中图分类号: TP301

文献标识码: A

文章编号: 1673-629X(2022)03-0027-07

doi:10.3969/j.issn.1673-629X.2022.03.005

## A Solution to Path Planning Problems in Complex Scenarios

GUO Shu-jie, TIAN Hua, WANG Wei

(School of Electronic Engineering, Dalian Neusoft University of Information, Dalian 116023, China)

**Abstract:** Path planning has valuable application in many fields. Researchers have proposed a variety of algorithms to solve the path planning problem. However, the problem of path planning in complex scenes has not been well solved, especially in narrow channel and zigzag scenes. For this, a key point path planning (KPP) algorithm is proposed. Through the introduction of key points, the whole path planning is divided into several shorter sub-path planning, which reduces the scale of the problem and improves the planning efficiency. At the same time, these key points transform random path search into a heuristic one, so as to greatly improve the efficiency of path planning in narrow channel and zigzag. In the proposed approach, key points are selected near the vertices of obstacles that obstruct the straight line connection of the start and goal points. Sub-paths between key points are created and connected to form an initial path, which is then optimized through path compression for reducing the path length. Experiment shows that the KPP algorithm can not only efficiently complete path planning in narrow channel fields and zigzag scenes, but also has good performance in other common scenes.

**Key words:** path planning; path compression; key point; sub-paths; RRT-connect

## 0 引 言

路径规划是在有障碍物的环境中, 按照某种评价标准寻找一条从起始点到目标点的无碰撞路径<sup>[1]</sup>。路径规划在移动机器人、电子游戏、汽车导航等多个领域均有重要的应用价值, 是一个具有较高使用价值的研究方向。常用的路径规划算法有两种: 基于采样的路径规划算法和基于搜索的路径规划算法。Dijkstra<sup>[2]</sup>和 A\* 算法<sup>[3]</sup>是典型的基于搜索的路径规划算法。这类算法是完备的和最优的。由于要使用较小的步长遍历规划空间, 其规划效率相对较低。典型的基于采样的路径规划算法有 PRM<sup>[4]</sup>、RRT<sup>[5]</sup>和 EST<sup>[6]</sup>等。这类算

法使用随机采样的方式探索规划空间, 所以具有比基于搜索的路径规划更高的效率。基于采样的规划方式有一定的随机性, 仅仅是概率完备的, 而且不是最优的。为了进一步提高算法的搜索效率, JJK Jr 和 Steven M. LaValle 提出了一种改进算法 RRT-connect<sup>[7]</sup>。RRT-connect 能够提高 RRT 算法的效率, 不过它仍然不是最优的, 甚至不是渐近最优的。为了提高规划路径的质量, 出现了一些改进算法, 如 RRT\*、PRM\*<sup>[8]</sup>、LBT-RRT<sup>[9]</sup>。这类改进算法能够达到渐近最优。还有一个改进方向是将基于搜索的相关技术引入到基于采样的规划中来, 结合基于采样的规

收稿日期: 2021-03-24

修回日期: 2021-07-28

基金项目: 辽宁省教育基金资助项目(2020122)

作者简介: 郭书杰(1978-), 男, 博士, 讲师, 研究方向为智能优化和机器视觉。

划算法和基于搜索的规划算法的优势,兼顾了效率和路径质量,如 HRRT<sup>[10]</sup>、Anytime RRT<sup>[11]</sup>、Bi-RRT\*<sup>[12]</sup>、C-FOREST<sup>[13]</sup>、Informed RRT\*<sup>[14]</sup>等。这类算法在一定程度上提高了算法的收敛速度,但由于它们均是基于 RRT 的改进算法,所以仍然难以保证规划出来的路径质量的最优性。为了进一步提高路径质量,出现了 BIT\*、AIT\* 和 ABIT\*<sup>[15-16]</sup>等算法,通过引入节点排序和边排序,在限定的子集中执行排序搜索,从而达到提高路径质量的目的。还有其他 A\* 算法的改进算法<sup>[17-18]</sup>和基于智能优化算法的路径规划算法<sup>[19]</sup>。

实验发现,尽管这些改进算法在一定程度上提高了规划效率和路径质量,但在限制较多的空间中,特别是在窄通道场景和 Z 字形场景中,它们的规划效率会急剧降低,难以在有限的时间内规划出一条路径来。为了解决复杂场景下的路径规划问题,提出了一种相对高效的关键点路径规划算法(key point path planning, KPP),该算法在提高路径优化效率的同时,也优化了规划出来的路径的质量,较好地解决了特殊环境下的路径规划问题。实验结果表明,KPP 不仅能够高效地完成窄通道场景和 Z 字形场景中的路径规划,在其他常见场景的应用中也有不俗的表现。

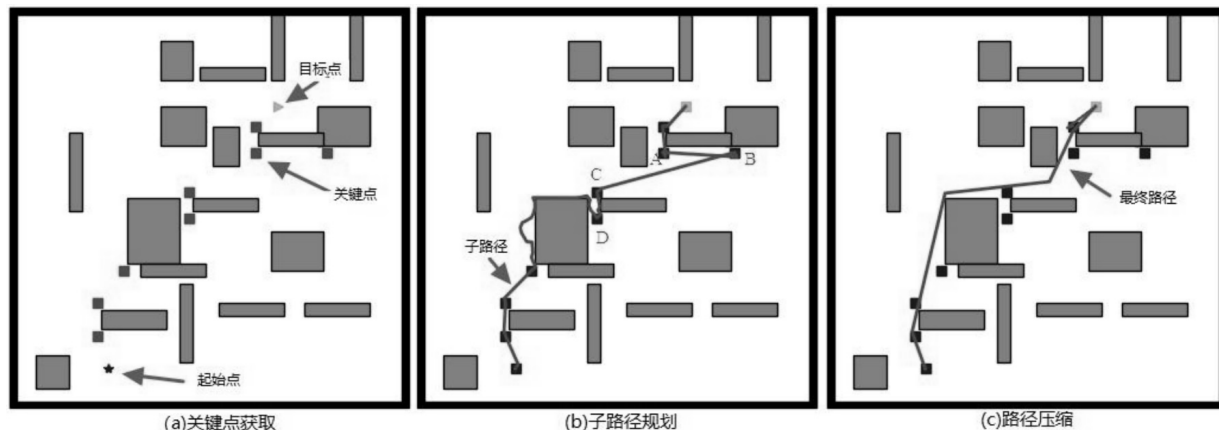


图 1 KPP 处理过程

关键点路径规划算法的处理过程如图 1 所示。第一步:找出起始点到目标点之间的关键点(见图 1(a))。这些关键点是通过将起始点和目标点连线上的障碍物的某些顶点外推得到的。第二步:选用某种路径规划算法规划出各个关键点之间的子路径;再以关键点为连接点,将这些子路径连接在一起,形成起始点到目标点间的初始路径。第三步:对第二步中得到的初始路径进行压缩,删除其中的冗余路径段,得到最终路径(见图 1(c))。

## 2.1 获取关键点

关键点是指绕过某一障碍物的最短路径上,与该障碍物的距离小于某一设定阈值的点。假设障碍物

## 1 路径规划问题

路径规划就是在给定空间中,规划出一条从起始点到目标点的无碰撞路径,使得该路径的代价尽可能小。记空间  $X \subset R^n$  为  $P$ ,  $X_{obs}$  表示  $P$  上被障碍物占据的空间;  $X_{free} = X \setminus X_{obs}$  表示  $P$  上可以通行的空间;  $D_{start}$  表示起始点;  $D_{goal}$  表示目标点;  $\sigma: [0, 1] \mapsto X$  表示连接  $D_{start}$  和  $D_{goal}$  之间的路径;  $\sum$  表示  $D_{start}$  和  $D_{goal}$  之间所有的路径组成的集合;  $S: \sum \mapsto R$  表示一条路径的代价函数;则路径规划问题可以描述为:

$$\min_{\sigma \in \sum} \{ S(\sigma) \mid \sigma(0) = D_{start}, \sigma(1) = D_{goal}, \forall t \in [0, 1], \sigma(t) \in X_{free} \}$$

## 2 关键点路径规划算法

关键点路径规划算法基于“两点之间直线最短”的原理,首先找出阻碍起始点与目标点直线连通的障碍物 KObses;然后找出绕过 KObses 的最近路线必须经过的点作为关键点;接着以这些关键点为引导,将路径规划从随机搜索转换为启发式搜索,从而提高规划效率和路径质量;最后通过路径压缩来减少路径长短,进一步提高规划出的路径的质量。

$Obs_i$  的外边界为  $boundry_i$ , 从起始点到目标点的路径中,一条绕过  $Obs_i$  的最短路径为  $path_i$ , 则称集合  $\{ point_i \mid point_i \in X_{free} \wedge point_i \in path_i \wedge distance(point_i, boundry_i) < \varepsilon \}$  中的点为关键点。这些关键点是距离障碍物较近的可通行节点,所以经过关键点的路径是绕过某障碍物的近似最优路线。关键点的获取要经过两步完成,第一步是获取所有潜在关键点,第二步是从潜在关键点中选出相对较好的点作为最终的关键点。

### 2.1.1 获取潜在关键点

为了便于讨论,该文以实际障碍物的正外接矩形来代表障碍物本身,文中的障碍物均指实际障碍物的

正外接矩形。在寻找可能的关键点时,首先找到所有与起始点和目标点的连线相交的障碍物。然后对这些障碍物的四个顶点分别向远离中心点的方向外推,如对左上点进行左推和上推,对右下点进行右推和下推。通过这种外推得到的点就是可能的关键点。左上点的外推过程如下:首先完成对左上点 A 的左推,即以步长  $step$  向左寻找障碍物的左侧边界,若在地图边界内找到了该障碍物的左侧边界则把当前点拉回到障碍物内,并以步长  $step$  向上寻找障碍物的上侧边界;若在地图边界内找到了该障碍物的上侧边界,则对该点进行一步左推,并返回左推后的点,即 A 的外推点。具体过程如图 2 所示。障碍物的其他顶点的外推过程与此类似,不再赘述。需要说明的是,为了便于后续操作,起始点和目标点也分别被视为可能的关键点。

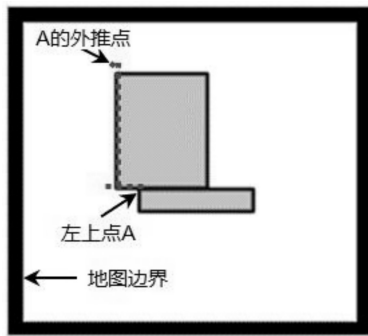


图2 左上点的外推过程

### 2.1.2 关键点选择

在一次路径规划中,可能的关键点数量较多,其中有一些点能够在路径规划过程中起到正向的启发作用,也就是说经过这些关键点的路径代价会比较小,另一些则不能。关键点选择的目的是从可能的关键点中,选出那些具有正向启发作用的点作为最终的关键点。

在进行关键点选择时,从起始点开始,在可能的关键点组成的集合  $SP$  中查找距离最近的无碰撞连接点  $p$ ,若找到则将  $p$  加入到关键点集合  $SKP$  中,并将  $p$  点从  $SP$  中删除,然后从  $p$  开始在  $SP$  中继续查找距离最近的无碰撞连接点。重复上述过程,直至找不到无碰撞连接点或找到的点是目标点。如果目标点不在  $SKP$  中,则从目标点开始,在  $SP$  中查找距离最近的无碰撞连接点  $p$ ,若找到则将  $p$  加入到关键点集合  $SKP$  中,并将  $p$  点从  $SP$  中删除,然后从  $p$  开始在  $SP$  中查找距离最近的无碰撞连接点。重复上述过程,直至找不到无碰撞连接点或找到的点已经存在于  $SKP$  中。简单来说,关键点选择的过程,就是分别从起始点和目标点开始,在  $SP$  中查找最近的可以无碰撞连接的点的过程。

## 2.2 实施路径规划

通过关键点的选择,得到了一个起始点到目标点

之间的关键点数组。数组中相邻两个关键点之间的路径称为子路径。接下来根据需要选用某种路径规划算法完成子路径的规划。最后以关键点为连接点,将这些子路径连接在一起,形成初步路径。由于难以找到适合于任何情况的最优关键点选择策略,所以选出的关键点并不都是对路径规划具有正向的启发作用,如图1(b)中的关键点 B;同时,因为所选择的路径规划算法不一定是最优算法,甚至可能不是渐近最优的算法,所以初步规划出来的路径通常不是最短路径。为了解决这一问题,需要对初始路径做进一步优化,也就是路径压缩。

路径规划的具体过程如下:假设经过关键点的生成与选择后,得到了关键点数组  $arrayKeypoint$ 。对于  $arrayKeypoint$  中的每一个点  $arrayKeypoint[i]$ ,使用选定的路径规划算法,规划以  $arrayKeypoint[i]$  和  $arrayKeypoint[i+1]$  为起始点和目标点的路径,得到子路径  $subPath_i$ ,并将  $subPath_i$  加入到初始路径  $Path$  中。最后对  $Path$  进行压缩,以提高路径的质量。路径压缩是为了减少路径长度,提高路径质量,其实质就是删除一条路径中的冗余路径点。冗余路径点的定义如下:

冗余路径点:对于一条路径  $Path$  上的一段子路径  $sPath$ ,令  $sPath$  的起止点分别是  $PointStart$  和  $PointEnd$ ,如果由  $PointStart$  和  $PointEnd$  两个点构成的子路径也是无碰撞的可行路径,则称  $sPath$  上除起止点之外的其他路径点为冗余路径点。

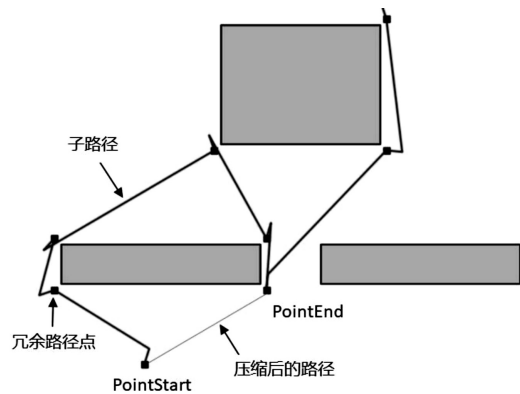


图3 冗余路径点

如图3所示,因为由  $PointStart$  和  $PointEnd$  两点构成的路径是无碰撞的可行路径,所以以  $PointStart$  和  $PointEnd$  为起止点的子路径  $sPath$  上的其他点均为冗余路径点,子路径  $sPath$  就可以被压缩成  $compressed path$ ,由此来降低路径长度。路径压缩从路径的目标点开始,逐一查找距离当前点  $curr\_point$  最近的无障碍连接点  $farthest\_collision\_free\_point$ ,并删除  $curr\_point$  与  $farthest\_collision\_free\_point$  之间的冗余路径点。路径压缩算法首先将目标点赋值给当前点  $curr\_point$ 。



point,并将 curr\_point 加入到压缩后的路径中;然后查找路径中距离当前点最远的无碰撞点 farthest\_collision\_free\_point,并将该点加入到压缩后的路径中;最后以 farthest\_collision\_free\_point 为当前点,继续寻找。循环执行上述过程,直到找到的最远无碰撞点为起始点。

### 2.3 算法的总流程

关键点路径规划算法首先获取可能的关键点,潜在关键点的获取是通过对起始点与目标点连线上的障碍物的顶点进行外推实现的。然后进行关键点选择,依据步长最小且可连通为原则,在可能的关键点中选出路径规划所需的关键点。接下来进行初步的路径规划,选用某种路径规划算法在两个相邻的关键点之间规划子路径,并将这些子路径合并成总路径。最后进行路径压缩,采用删除总路径中的冗余路径点的方式,对原始路径进行优化,得到最终路径。

### 2.4 算法分析

关键点路径规划算法的优点是效率较高,规划的路径质量也较好。与其他路径规划算法相比,关键点路径规划算法多了两个步骤,一个是关键点的获取,另一个是路径压缩。通过关键点获取,算法找到了绕过从起始点到目标点之间的障碍物的关键点,并使用这些关键点来引导路径规划算法完成规划。这些关键点将一条未知路径分割成多条子路径。通过分割将规划空间切分成更小的子空间,从而减小了问题规模,提高了规划效率。另一方面,由于大多数关键点之间是可以无障碍连通的,所以这些子路径的规划就变得非常简单,只要将两个关键点连接起来就行了,路径规划的效率得到了较大的提高。同时,关键点的引入将随机搜索转换为启发式搜索,能够使算法朝着目标点的方向进行搜索,所以算法规划出了更短的路径。尽管关键点的引入极大地提高了算法的效率和路径的质量,然而,由于地图环境复杂多样,起始点和目标点的位置又不确定,导致难以找到一个最优的关键点选择策略,

障碍物的一个顶点在某种情况下是最优路径经过的点,在另外一种情况下又不是了。所以照固定的关键点选择策略选出的关键点中,会出现少量坏点,这些坏点降低了路径的质量。另外用于完成关键点间路径规划的算法不一定是最优算法,规划出来的子路径也可能会出现较多的冗余路径点。对此,路径压缩很好地解决了这两个问题。路径压缩能够以较高的效率找出路径中的冗余部分,并将其删除,从而提高最终路径的质量。所以通过关键点和路径压缩两个操作,可以提高路径规划的效率和路径的质量,特别是在狭窄通道环境和“Z”字形通道环境下。

关键点路径规划算法的缺点是需要进行频繁的碰撞检测。在进行关键点获取和路径压缩时,都需要多次进行碰撞检测。当地图上障碍物数量过多时,会在一定程度上影响规划效率。

## 3 实验

为了验证关键点路径规划算法的性能,进行了多组对比实验。实验中选用 RRT-connect 完成关键点之间的路径规划。

### 3.1 与 RRT-connect 算法的对比实验

#### 3.1.1 实验过程

实验模拟了四种场景(scenario),如图 4 所示。分别是离散障碍物场景、窄通道场景、Z 字形场景和迷宫场景。实验中选用规划效率和规划出来的路径质量作为评价算法优劣的标准。每个场景选择五组典型的起始点和目标点进行路径规划,图 4 中的  $S[i]$  表示第  $i$  组的起始点,  $G[i]$  表示第  $i$  组的目标点。每组运行 50 次,记录每次运行的时间和路径长度。通过比较平均运行时间和平均路径长度来对比算法的规划效率和规划出来的路径质量。实验中 RRT-connect 算法的迭代次数设置为 50 000。

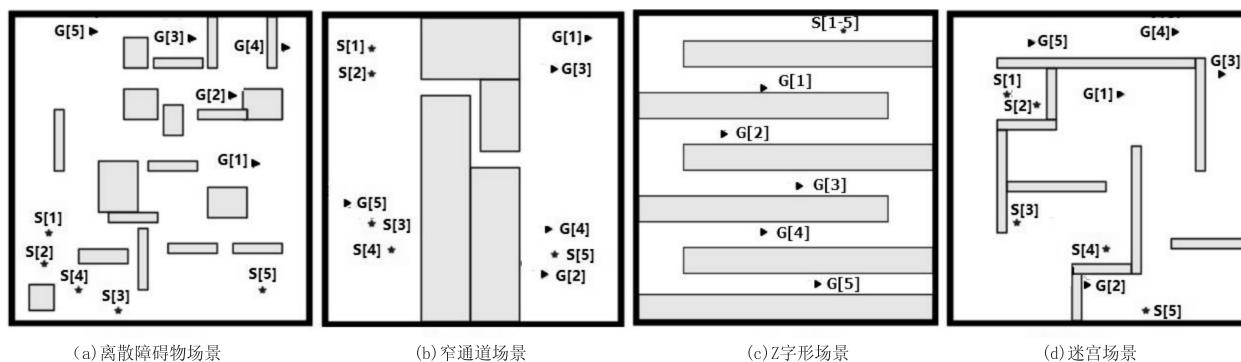


图 4 四种场景

#### 3.1.2 实验结果

离散障碍物场景实验的结果如图 5 所示。图 5 (a) 是路径规划所用时间的对比图,其纵坐标是规划

路径所用的平均时长,单位为毫秒;横坐标是路径起止点的组序,如横坐标的“1”就表示规划的是以图 4(a) 中的  $S[1]$  为起始点、 $G[1]$  为目标点的路径;图例中的

KPP 代表关键点路径规划算法的运行结果,RRT-C 是 RRT-connect 算法的运行结果。图 5(b) 是规划出来的路径长度对比图,其纵坐标表示算法规划出来的平

均路径长度,横坐标及图例所表示的意义与图 5(a) 相同。如无特殊说明,后续实验结果图中各元素的意义均与图 5 相同,不再赘述。

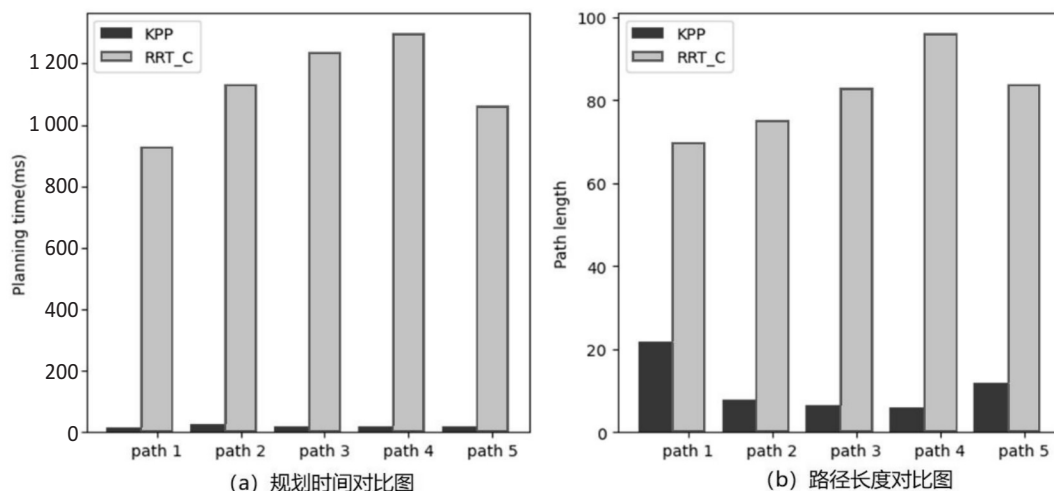


图5 离散障碍物场景实验结果

由图 5 不难看出,在对图 4(a) 中的 5 组路径规划中,KPP 算法在规划效率和路径质量方面都比 RRT-connect 算法好很多。KPP 算法的平均耗时仅占 RRT-connect 算法的 1.66%,KPP 算法规划的平均路径长度仅占 RRT-connect 算法的 13.22%。

窄通道场景的实验结果如图 6 所示。在窄通道场景下,KPP 算法在效率方面的提高更加明显,KPP 算

法的平均耗时仅占 RRT-connect 算法的 0.16%。KPP 算法规划的平均路径长度占 RRT-connect 算法的 21.67%。这一比例有所升高,其原因是窄通道限制了路径的大致方向。不管哪种算法规划的路径,都要从窄通道中通过,而该通道的长度是固定的,所以在通道内部,这些路径的长度差就被限定在一个较小的范围内了。

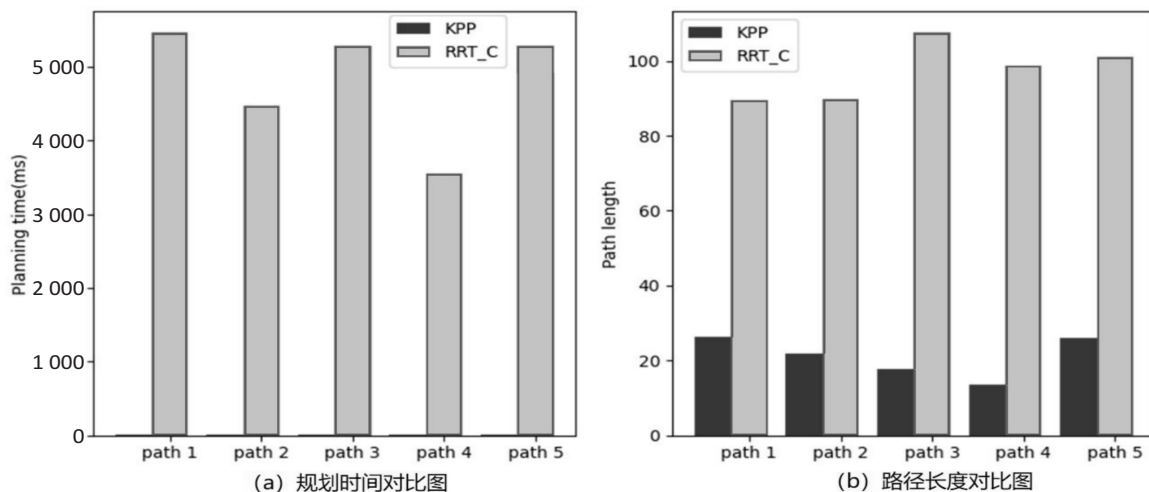


图6 窄通道场景实验结果

Z 字形场景的实验结果如图 7 所示。在 Z 字形场景实验中,图 4(c) 中的 5 组起止点中,RRT-connect 算法仅成功规划出了两组,分别是起止点相距较近的 S[1]到 G[1]的路径和 S[2]到 G[2]的路径。这两组路径规划过程中,KPP 算法的平均耗时仅占 RRT-connect 算法的 0.09%;KPP 算法规划的平均路径长度占 RRT-connect 算法的 23.18%。与窄通道相比,Z 字形通路更多地限制了路径的走向,所以 Z 字形场景中,KPP 算法规划的平均路径长度与 RRT-connect 算

法规划的平均路径长度进一步提高。

迷宫场景的实验结果如图 8 所示。KPP 算法的平均耗时仅占 RRT-connect 算法的 0.08%;KPP 算法规划的平均路径长度占 RRT-connect 算法的 12.32%。

由实验结果可以看出,在四种不同的场景中,KPP 算法的规划效率和规划出来的路径长度均比 RRT-connect 算法好很多。特别是在窄通道场景和 Z 字形场景中,当以高效著称的 RRT-connect 算法都难以成功规划处路线时,KPP 仍然可以在较短时间内规划出

一条质量较高的路径。

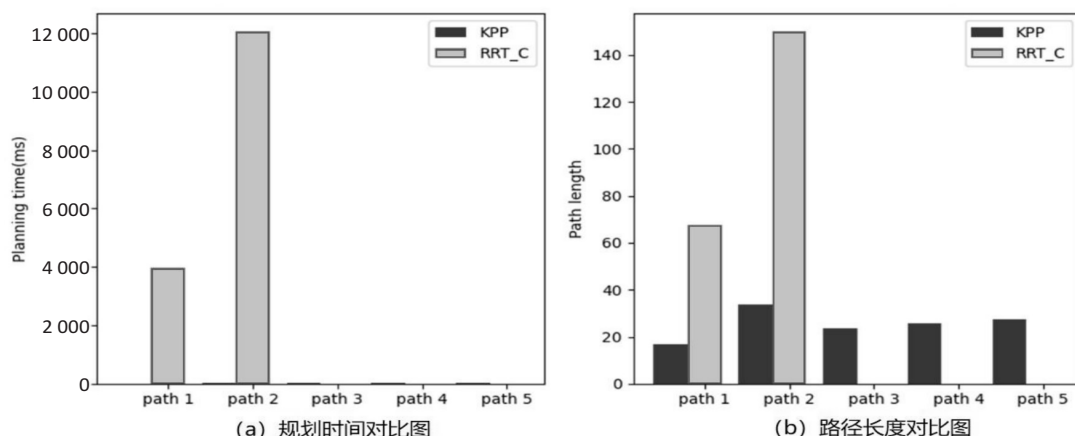


图 7 乙字形场景实验结果

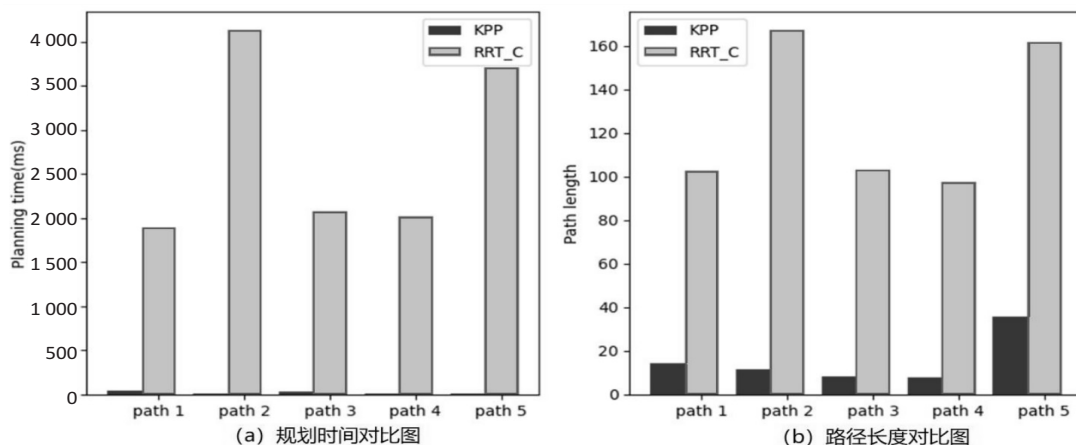


图 8 迷宫场景实验结果

### 3.2 与 BIT \* 算法的对比实验

为了进一步检验 KPP 算法的性能,与 BIT \* 算法做了对比实验,实验选用了 3 组起止点。每组起止点完成 50 次规划,通过对比平均规划用时和平均路径长度来评价算法性能。

实验结果如下:进行第一组路径规划时,BIT \* 算法的平均耗时为 27 343.75 毫秒,平均路径长度为 26.976 321;KPP 算法的平均耗时为 0.27 毫秒,平均路径长度为 26.131 984。进行第二组路径规划时,BIT \* 算法的平均耗时为 25 312.5 毫秒,平均路径长度为 23.484 871;KPP 算法的平均耗时为 0.21 毫秒,平均路径长度为 23.332 335。进行第三组路径规划时,BIT \* 算法的平均耗时为 29 484.375 毫秒,平均路径长度为 33.495 748;KPP 算法的平均耗时为 15.625 毫秒,平均路径长度为 34.640 041。三组路径规划中,KPP 算法平均耗时明显要小于 BIT \* 算法;在路径长度方面,除第三组外,KPP 规划的路径长度均略小于 BIT \* 算法规划的路径长度。在第三组路径规划时,由于关键点选择时,以最小步长为原则,选取距离最近的节点,所选节点并不是最好的,从而导致 KPP 算法所规

划的路径略长。

## 4 结束语

KPP 算法通过引入关键点,将较复杂的长路径规划转化为规模较小的子路径规划;同时将随机搜索转化为启发式搜索,所以其规划效率较高,特别是在窄通道、Z 字形或障碍物较密集的场景中,当其他算法无法成功规划路径时,KPP 算法仍然可以较快地完成规划。同时,路径压缩的操作也减少了 KPP 算法规划出来的路径长度。尽管 KPP 算法是用于解决 2D 问题路径规划问题的,可以很容易地将其扩展到立体空间中,求解 3D 问题路径规划的问题。

由于选出的关键点不一定是最合理的,部分关键点可能会给规划带来错误的引导,进而导致 KPP 规划出来的路径质量降低,所以 KPP 算法不是最优的。

### 参考文献:

- [1] EELE A J, RICHARDS A. Path-planning with avoidance using nonlinear branch-and-bound optimization[J]. Journal of Guidance Control & Dynamics, 2009, 32(2): 384-394.
- [2] DIJKSTRA E W. A note on two problems in connexion with

- graphs[J]. *Numerische Mathematik*, 1959, 1(1): 269–271.
- [3] HART P E, NILSSON N J, RAPHAEL B. A formal basis for the heuristic determination of minimum cost paths[J]. *IEEE Transactions on Systems Science and Cybernetics*, 1968, 4(2): 100–107.
- [4] KAVRAKI L E, SVESTKA P, LATOMBE J C, et al. Probabilistic roadmaps for path planning in high-dimensional configuration spaces[J]. *IEEE Transactions on Robotics & Automation*, 1996, 12(4): 566–580.
- [5] LAVALLE S M, KUFFNER JR J J. Randomized kinodynamic planning[J]. *The International Journal of Robotics Research*, 2001, 20(5): 378–400.
- [6] HSU D, KINDEL R, LATOMBE L C, et al. Randomized kinodynamic motion planning with moving obstacles[J]. *International Journal of Robotics Research*, 2002, 21(3): 233–255.
- [7] JR J J K, LAVALLE S M. RRT-connect: an efficient approach to single-query path planning[C]//2000 IEEE international conference on robotics and automation. San Francisco: IEEE, 2000: 995–1001.
- [8] KARAMAN S, FRAZZOLI E. Sampling-based algorithms for optimal motion planning[J]. *International Journal of Robotics Research*, 2011, 30(7): 846–894.
- [9] SALZMAN O, HALPERIN D. Asymptotically near-optimal RRT for fast, high-quality, motion planning[J]. *IEEE Transactions on Robotics*, 2016, 32(3): 473–483.
- [10] URMSON C, SIMMONS R. Approaches for heuristically biasing RRT growth[C]//IEEE/RSJ international conference on intelligent robots and systems. [s. l.]: IEEE, 2003: 1178–1183.
- [11] FERGUSON D, STENTZ A. Anytime RRTs[C]//IEEE/RSJ international conference on intelligent robots and systems. Beijing: IEEE, 2006: 5369–5375.
- [12] AKGUN B, STILMAN M. Sampling heuristics for optimal motion planning in high dimensions[C]//IEEE/RSJ international conference on intelligent robots and systems. San Francisco: IEEE, 2011: 2640–2645.
- [13] OTTE M, CORRELL N. C-FOREST: parallel shortest path planning with super linear speedup[J]. *IEEE Transactions on Robotics*, 2013, 29(3): 798–806.
- [14] 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 international conference on intelligent robots and systems. Chicago: IEEE, 2014: 2997–3004.
- [15] GAMMELL J D, SRINIVASA S S, BARFOOT T D. Batch informed trees (BIT\*): sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs[C]//2015 IEEE international conference on robotics and automation (ICRA). Seattle: IEEE, 2015: 3067–3074.
- [16] STRUB M P, GAMMELL J D. Adaptively informed trees (AIT\*): fast asymptotically optimal path planning through adaptive heuristics[C]//2020 IEEE international conference on robotics and automation (ICRA). Paris: IEEE, 2020: 3191–3198.
- [17] 王洪斌, 尹鹏衡, 郑维, 等. 基于改进的 A\* 算法与动态窗口法的移动机器人路径规划[J]. *机器人*, 2020, 42(3): 346–353.
- [18] 余星宝, 杨慧斌, 周玉凤, 等. 改进 A\* 的 4 阶贝塞尔曲线路径规划[J]. *轻工机械*, 2020, 38(6): 64–67.
- [19] 葛志远, 肖本贤. 使用改进蚁群算法的 AGV 路径规划研究[J]. *机械设计与制造*, 2020(6): 248–251.