doi:10.19562/j.chinasae.qcgc.2024.07.012

一种基于简化可视图的建图和规划方法*

范晓临[1],张旭东[1],邹 渊[1],尹 鑫[2],刘颖群[1]
(1. 北京理工大学机械与车辆学院,北京 100081; 2. 上海涵润汽车电子有限公司,上海 201601)

[摘要] 当前车辆路径规划大部分是基于栅格地图的规划方法,这种方法在搜索面积较大时计算量也会大幅增加。相比之下,基于可视图的方法能够在路径搜索时减小计算量,但是受到障碍物复杂程度的影响较大。针对这一问题,本文结合SLAM和可视图的方法,提出了一种简化可视图的建图和规划方法。首先使用改进的SLAM算法生成点云地图,并进行动态障碍物的剔除。接着生成可视图,并基于障碍物的大小和顶点处内凹角的大小对可视图中多边形的复杂边缘进行简化,剔除冗余的顶点。最后通过仿真和实车实验证明,该方法相对原有的算法,在保证建图精度的情况下,可视图中多边形的顶点数量减少20%~30%,地图更新时间和整体算法的运行时间减少30%以上。这表明本文方法能够有效减小建图和规划过程的计算量和算法的运行时间。

关键词:可视图;路径规划;SLAM;智能车辆

A Mapping and Planning Method Based on Simplified Visibility Graph

Fan Xiaolin[1], Zhang Xudong[1], Zou Yuan[1], Yin Xin[2] & Liu Yingqun[1]
1. *School of Mechanical Engineering, Beijing Institute of Technology, Beijing* 100081;
2. *Shanghai Hanrun Automotive Electronics Co., Ltd., Shanghai* 201601

[**Abstract**] Most of the current vehicle route planning is based on the grid map planning method, which will greatly increase the amount of calculation when the search area is large. In contrast, the method based on visibility graph can reduce the amount of calculation during path search, but is greatly affected by the complexity of obstacles. For this problem, combining the SLAM and visibility graph methods, a simplified visibility graph construction and planning method is proposed in this paper. Firstly, the improved SLAM algorithm is used to generate point cloud maps, and dynamic obstacles are removed. Then a visibility graph is generated, and the complex edges of polygons in the visibility graph are simplified based on the size of the obstacle and the size of the concave angle at the vertex to eliminate redundant vertices. Finally, through simulation experiments and real vehicle experiments, it is proved that compared with the original algorithm, this method can reduce the number of polygon vertices in the visibility graph by 20%-30% while ensuring the accuracy of mapping. The map update time and the running time of the overall algorithm are also reduced by more than 30%. It shows that the method in this paper can effectively reduce the amount of calculation and the running time of the algorithm in the mapping and planning process.

Keywords: visibility graph; path planning; SLAM; intelligent vehicle

前言

随着汽车智能化水平的不断提高,无人驾驶技术成为汽车领域的重要研究方向,而路径规划是其中的关键技术之一。智能车辆的路径规划是指在一定环境模型的基础上,给定起点与目标点后,按照性能指标规划出一条无碰撞、能安全到达目标点的有

*北京市科协金桥工程、北京市自然科学基金(3212013)和中国汽车工程学会青年托举人才项目(YESS20200301)资助。
原稿收到日期为2023年09月24日,修改稿收到日期为2023年12月20日。
通信作者:张旭东,副教授,博士,E-mail:Xudong.zhang@bit.edu.cn。

效路径[1]。

当前车辆路径规划主要分为传统算法、智能仿生算法、基于采样的算法等。其中传统算法又分为人工势场法[2]、bug法[3]、基于图搜索的算法。智能仿生算法分为蚁群算法[4]、遗传算法[5]、神经网络法[6]等。基于采样的算法分为PRM算法[7]、RRT算法[8]等。其中基于图搜索的算法中,有比较经典的基于栅格地图的算法,比如Dijkstra算法[9]、A*算法[10]、D*算法[11]等,也有基于Voronoi图的规划方法[12]和基于可视图的规划方法[13]。本文主要研究基于可视图的建图和规划算法。

基于栅格地图的规划方法,当搜索范围较大时,基于栅格地图搜索算法的计算量随着变大。而基于可视图的规划方法将障碍物转化为多边形,在地图面积较大的路径搜索过程中能够极大减小计算量。但是可视图在建图过程中生成的多边形的复杂程度受到实际障碍物的复杂程度影响非常大,路径搜索的速度也依赖于可视图中障碍物多边形的复杂程度。当障碍物的形状简单,顶点和边较少时,可视图的复杂程度很小。但是当障碍物的形状较复杂,具有较多的顶点和边时,生成可视图也会具有较多的顶点和边,会极大地增加可视图建图时间和路径搜索的时间。

对基于可视图的建图和路径规划方法,国内外研究也较多。黎萍等[14]利用矩形包络障碍物生成路径点,结合可视图和A*搜索算法提出一种新的路径规划算法,但此方法仅适用于二维空间。李霜琳等[15]提出融合鸽群优化算法与可视图法,提高了获得轨迹的平滑性,但是对于无顶点型障碍物采用贴壁绕行的方法可能导致最终规划路径长度较长。Ou等[16]提出了一种新的初始化方法,将自适应能见度图和Dijkstra算法相结合,显著提高了初始路径和探索的多样性、优化精度和计算速度。Lv等[17]提出了一种并行可见性图构建和路径优化的方法,提高了路径规划的时间效率和空间效率。Yang等[18]将可视图应用到真实世界的车辆导航中,但是由于真实世界中障碍物的复杂性,建图和路径规划过程依然需要非常大的计算成本。Li等[19]在文献[18]的基础上提出了一种基于障碍物多边形最长边的过滤方法,能够有效简化可视图,提高建图和规划效率。但是在真实的环境下,障碍物外围形状非常不规则,所提取的多边形并没有较长的边。这种情况下,基于最长边的简化方法效果并不理想。

本文中结合SLAM算法,并在文献[18]的基础上进行改进,提出一种简化可视图的建图和规划方法。首先使用改进的SLAM算法生成点云地图,并进行动态障碍物的剔除,接着生成可视图,并基于障碍物的大小和顶点处内凹角的大小对可视图中多边形的复杂边缘进行简化,剔除冗余的顶点。最后通过仿真和实车实验,证明了算法的可行性以及优化效果。

这种方法的好处在于,使用基于障碍物大小的自适应系数进行简化,既能保证可视图的精度,又能极大地减少计算成本。对于较小的障碍物,所对应多边形的顶点较少,同时自适应简化系数较小,会倾向于保留更多的顶点,保证了可视图的精度。对于较大的多边形,对应的顶点较多,同时自适应简化系数较大,会剔除冗余的顶点,提高可视图的简化程度。

1 基于SLAM的点云地图获取

本文的算法主要由点云地图获取、可视图建图以及路径规划3部分组成,算法的系统整体框架图如图1所示。

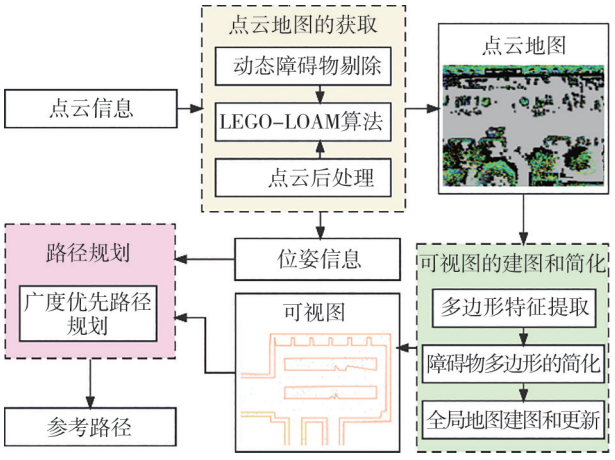

图1 系统整体架构图

在进行可视图建图之前,需要先获取当前车辆周围环境的点云地图,以及车辆在环境中的位姿信息。在本文的研究中,首先使用激光雷达获取点云信息,并使用比较成熟的LEGO-LOAM算法对激光雷达原始数据进行处理,之后在原有算法的基础上,使用贝叶斯滤波的方法进行动态障碍物的剔除,再对最后得到的点云进行进一步的后处理,以此提高建图的精度和效果。图2所示为点云地图获取过程

的算法架构图。

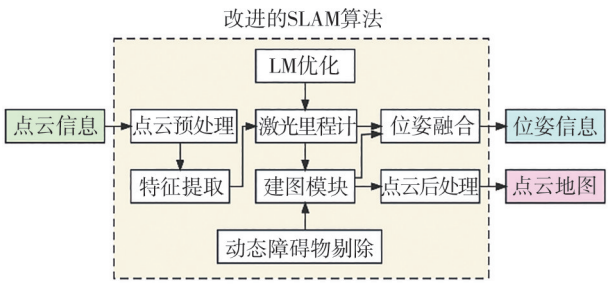

图 2　点云地图获取算法架构图

1.1　基于LEGO-LOAM的改进SLAM算法

本文使用目前比较成熟的LEGO-LOAM算法[20],并进行进一步的优化。同步定位与建图算法主要包括4个步骤,分别是点云预处理、特征提取、运动估计和回环检测。这里采用稀疏和密集的点云匹配技术,同时保证点云地图的实时性和精度,在无人驾驶领域得到比较广泛的应用。算法主要由两个部分组成。首先使用激光雷达和IMU获取机器人的位姿及环境中机器人周围的稀疏点云,并通过多帧点云对机器人进行运动估计以及激光点云的配准,得到机器人在三维环境中的行驶轨迹和周围环境的点云地图。然后使用点云地图优化的算法对机器人行驶轨迹和局部点云地图进行进一步优化,得到最终的机器人位姿信息和环境点云地图。

这种方法的优点是可以较快速处理环境中大量的点云信息,且对不同的环境有较好的适应性。但是这种方法也存在一些问题,比如无法去除环境中的动态障碍物,以及点云地图中还包含大量对机器人行驶无关的无用信息。

1.2　基于贝叶斯滤波的动态障碍物剔除

本文使用基于贝叶斯滤波的动态障碍物剔除方法,对点云中动态障碍物的点云进行剔除。以此解决上述算法无法剔除环境中的动态障碍物的问题。

贝叶斯滤波是一种概率推断的方法,用于估计系统状态随时间推移的变化情况。它基于贝叶斯定理和递归贝叶斯估计的概念,通过使用已经得到的先验概率和观测数据,更新系统状态的后验概率。基于贝叶斯滤波的基本原理,系统状态一般被描述为一个随机变量,其取值的随机性通常用概率分布来表示。观测量也被描述为随机变量,其测量噪声也是用一个概率分布来表示。

将机器人周围环境中的动态障碍物所得到的点云信息表示为测量噪声,通过计算概率分布来识别动态障碍物所对应的点云。由于机器人和激光雷达的模型是确定的,在机器人周围方向上出现的点云应该也是确定的,因此在建图和定位过程中,如果点云地图中某个点在前后几帧出现的位置存在较大差异,那么此点有可能是动态障碍物对应的点。在机器人建图和定位的过程中,对每个点进行多帧比对,通过识别与当前激光雷达观测到的点在角度、距离上最接近的点。通过距离比、角度比、航向差等参数计算其为动态障碍物的概率。当计算得到的数值大于设定的阈值时认为其是动态障碍物,然后在点云地图中将其剔除。

通过该方法,能够有效识别环境中的动态障碍物,得到去除动态障碍物的点云地图。

1.3　基于L-M非线性优化的点云重定位

本文采用L-M非线性优化算法,将原始点云地图作为重定位的特征点云地图,其中分为角特征点云地图以及面特征点云地图。为能够在后续构建的可视图中获取车辆的定位信息,需要基于点云地图进行重定位来获取车辆在可视图中的相对坐标。

SLAM本质上是对$[x,y,z,yell,roll,pitch]$这6自由度变量进行优化,采用L-M非线性优化算法迭代求解变换矩阵,若需要进行重定位,需要优化实时点云特征点与特征点云地图特征点间的距离,当优化问题收敛后,便得到了相对准确的定位信息。

首先须获取车辆相对于特征点云地图之间的粗略定位信息得到初始位姿,该定位信息可以通过GPS或设定车辆定位起点获得,相当于提供了一个最优解附近的粗略解,在这个条件下可以将问题近似看作一个凸优化问题进行求解。然后根据该初始定位点来计算当前点云中角点与平面点和特征点云地图中的角点与平面点间的距离得到$[dl1,dl2,\cdots]$与$[ds1,ds2,\cdots]$,基于L-M非线性优化算法,将两个距离矩阵都优化到最小值,便可以得到变化矩阵$\boldsymbol{T}=[\mathrm{d}x,\mathrm{d}y,\mathrm{d}z,\mathrm{d}yell,\mathrm{d}roll,\mathrm{d}pitch]$,通过旋转矩阵可以将初始位姿转换到相对准确的定位坐标下,再基于这个坐标进行SLAM定位,可以实现鲁棒的定位效果。

1.4　激光点云后处理

对点云地图进行进一步的后处理,使用kdtree算法去除冗余离散点,使用直通滤波剔除对车辆行驶不会产生影响的障碍物冗余点云,以此提高建图的精度和效果。

在获得去除动态障碍物后的点云地图,还不能

直接作为地形计算的参考点云地图,原因如下:

(1)SLAM建图以及剔除动态障碍物过程中会引入干扰点,这些点会对后续地图计算产生较大的影响,需要在计算之前进行滤波。

(2)对于高度远高于车辆、对车辆行驶不会产生影响的障碍物,如车道上方的树枝等物体,在地图构建的过程中会被判定为不可通行区域,这些对地形描述影响不大的部分也需要进行滤波。

一些对地形有影响的冗余离散点,选择使用半径滤波的方式去除,使用kdtree算法寻找每个点最近的k个点,对这些点按照距离建立排序,若距离小于0.4 m的点少于5个,则认为是离散点,需要去除。图3所示为滤波原理示意图,蓝色点为正在判定的点,红色点为周围的离散点。经过滤波后有效减少了离散点的数量,能有效提高代价地图的计算质量。

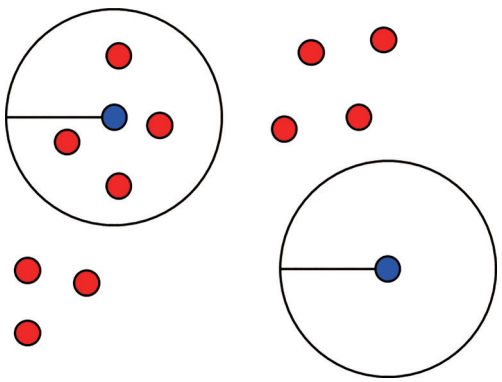

图3 半径滤波原理示意图

对于不需要区域的点云,由于构建点云地图是将位姿$[x,y,z,yell,roll,pitch]$与每一帧特征点云进行匹配融合就可以得到点云地图,故在融合前对每一帧点云进行直通滤波,滤掉不需要的点云部分再依据位姿进行融合便可以得到本文计算所需的点云地图S。

2 可视图的建图和简化

2.1 局部地图多边形提取

本节主要基于上节中生成的点云地图,生成可视图,并对可视图中的多边形进行简化,获得简化后的可视图。

对点云地图进行处理,通过二值化投影,均值滤波、边缘点提取、顶点降采样等获得最初的障碍物多边形。

在获取点云地图之后,须通过点云地图S提取障碍物多边形。首先将点云地图S投影到二值化地图I上,在二值化地图中,黑色像素点表示障碍物,白色像素点表示可通行区域。同时以车辆的尺寸为参考,对S中的点进行膨胀处理。获得二值化地图I之后,用均值滤波器进行处理,获得灰度图像I_{blur}。之后,提取I_{blur}中的边缘点,并分析边缘点的拓扑分布,以此获得一组封闭多边形$\{P^k_{\text{coutour}} \subset Q | k \in Z^+\}$,对于每个$\{P^k_{\text{coutour}}\}$轮廓,使用RDP方法[21]来减少顶点数量。

检查每个顶点的轮廓上的两个连接边之间的内角,以推断障碍物的局部曲率,消除内角小于阈值的顶点。最终生成局部地图中的多边形$\{P^k_{\text{local}}\}$,具体伪代码如算法1所示。

算法1

输入:点云地图S

输出:多边形$\{P^k_{\text{local}}\}$

1 通过点云地图S获取二值化地图I

2 使用均值滤波器,获取灰度图I_{blur}

3 基于注释中的方法,提取一组多边形$\{P^k_{\text{coutour}}\}$

4 **for** 每个$\{P^k_{\text{coutour}}\}$ **do**:

5 使用RDP方法减少顶点数量

6 检查每个顶点的内角,剔除小于阈值的顶点

7 生成局部地图中的多边形$\{P^k_{\text{local}}\}$

8 **end for**

2.2 障碍物多边形简化

使用基于障碍物多边形大小的方法对前文获得的多边形进行进一步的处理,减少多边形的顶点数量,以简化多边形。

在复杂场景中,对于1.2节中得到的多边形的轮廓顶点$\{P^k_{\text{local}}\}$,还存在以下几个问题:

(1)多边形的顶点数量非常多,尤其是较大障碍物对应的多边形中,在形状复杂处存在较多的冗余点和冗余边。

(2)在一些顶点处,内凹角较小。对于车辆来说,虽然是可通行区域,但是通行意义不大。

以上涉及的两种多边形顶点,在实际建图和规划过程中意义不大,但是却增加了建图过程和规划过程中的计算量,造成大量计算资源的浪费,须对其进行剔除,以简化多边形。在简化多边形的过程中,使用重构多边形的思路,即根据保留的点,重新生成多边形。

在简化大型复杂障碍物多边形的同时,保证小

型障碍物的特征点,设置简化阈值n_{limit},对顶点数量大于n_{limit}的多边形进行简化。对每个符合简化条件的多边形,首先获得多边形的最小外接矩形的对角线长度,并乘以一个系数k,获得参考简化系数d_0。

$$d_0 = \sqrt{\left(\max\left(x_i\right) - \min\left(x_i\right)\right)^2 + \left(\max\left(y_i\right) - \min\left(y_i\right)\right)^2} \tag{1}$$

式中:d_0为参考简化系数;x_i、y_i分别为当前多边形第i个点的横坐标和纵坐标。

考虑到一些障碍物过于庞大,得到参考简化系数d_1可能过大,导致地图精度下降,这里设置一个最大简化系数$d_{\max}$,在d_0和$d_{\max}$中取较小值得到最终的简化系数d_{limit}。

然后计算障碍物多边形每个顶点之间的欧式距离,若第$i-1$点和第i点之间的欧氏距离、第i点和第$i+1$点之间的欧氏距离都小于d_{limit},那么此点不保留。

$$d_{\text{limit}} = \min\left(d_0, d_{\max}\right) \tag{2}$$

式中:d_{limit}为简化系数;$d_{\max}$为最大简化系数。

对顶点处内凹角过小的顶点,此时设多边形的顶点按照逆时针进行排序和存储,则对于第$i-1$、i、$i+1$个顶点,首先计算第i个顶点的位置是否为内凹角,根据向量叉积定义,有:

$$\overrightarrow{P_i^k P_{i-1}^k} \times \overrightarrow{P_i^k P_{i+1}^k} = \left(x_{i-1} - x_i\right)*\left(y_{i+1} - y_i\right) - \left(x_{i+1} - x_i\right)*\left(y_{i-1} - y_i\right) \tag{3}$$

式中:P_i^k为第k个多边形的第i个顶点,×表示向量叉乘;* 表示数量乘积;x_i、y_i分别为第i个顶点的横坐标和纵坐标。

若式(3)结果大于0,则$\overrightarrow{P_i^k P_{i-1}^k}$在$\overrightarrow{P_i^k P_{i+1}^k}$顺时针方向,即点$i$对应的角为外凸角;若式(3)结果小于0,则$\overrightarrow{P_i^k P_{i-1}^k}$在$\overrightarrow{P_i^k P_{i+1}^k}$逆时针方向,即点$i$对应的角为内凹角;设阈值$ang_1$为内凹角的最大阈值,若此时内凹角的余弦值$\cos(angle_i)$大于$\cos(ang_{\text{limit}})$,则说明内凹角过小,此点不保留。

经过简化之后得到的多边形集合$\{P'^k_{\text{local}}\}$,相比原来的多边形有更少的顶点和边。

上述两个简化的流程具体如图4所示。图4中圆点为多边形的顶点,实线为多边形的边。在经过简化之后,蓝色顶点表示的冗余顶点和蓝色实线表示的多边形冗余边将被剔除。而红色顶点和红色实线将被保留。蓝色虚线表示不同多边形顶点之间相连生成的边,在相关顶点被剔除后也随之被剔除。

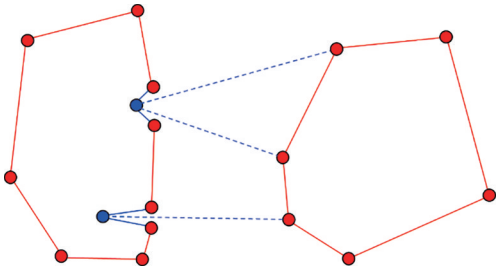

图4　多边形简化原理示意图

上述过程具体算法如算法2所示。

算法2

输入:多边形$\{P^k_{\text{local}}\}$

输出:简化后的多边形$\{P'^k_{\text{local}}\}$

```
for 每个 P'^k_local do:
    if 多边形的顶点数量 > n_limit
        for 多边形的每个顶点 do
            x_max = max(x_max, x_i)
            y_max = max(y_max, y_i)
            x_min = min(x_min, x_i)
            y_min = min(y_min, y_i)
            d_0 = sqrt((x_max - x_min)^2 + (y_max - y_min)^2)
            d_limit = min(d_0, d_max)
        end for
        for 多边形的每个顶点 do
            dist_{i-1,i} = sqrt((x_i - x_{i-1})^2 + (y_i - y_{i+1})^2)
            dist_{i,i+1} = sqrt((x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2)
            if dist_{i-1,i} < d_limit and dist_{i,i+1} < d_limit
                break
            end if
        if (x_{i-1} - x_i)*(y_{i+1} - y_i) - (x_{i+1} - x_i)*(y_{i-1} - y_i) < 0
        if (x_{i-1} - x_i)*(x_{i+1} - x_i) + (y_{i-1} - y_i)*(y_{i+1} - y_i) >
          cos(ang_limit)
                break
            end if
            end if
            将该点放入新多边形 P'^k_local 的点集中(该点不满足任
            一剔除条件)
        end for
    end if
end for
```

2.3　全局地图的建图和更新

使用包含局部层M_{local}和全局层M_{global}的双层可视图动态更新结构。局部层中包含车辆周围的地图信息,全局层包含全部地图信息。传感器获取到车

辆周围的障碍物点云信息,通过改进的SLAM算法中获得点云地图S,之后基于点云地图进行障碍物多边形的提取,获得可视图局部层,再对局部层中的障碍物多边形进行简化,得到最终的局部层地图,最后通过特征点匹配更新到全局层中。由于每次新处理的点云数据只是局部层中的信息,能够有效减少计算量。

在局部层的构建中,首先通过之前获得的多边形顶点构建可视图当中的可见边。设E_{local}为局部层中的可见边集合,E_{global}为全局层中的可见边集合。可见边主要由两部分组成,一部分是由障碍物生成的多边形的边,另一部分是这些多边形的顶点连线形成的边。一方面,在多边形集合$\{P'^{k}_{\text{local}}\}$中,由于多边形的边可以理解为现实中车辆绕过障碍物的可通行路径,所以将多边形集合中每个多边形的边作为可见边加入到局部层可见边集合E_{local}中。另一方面,不同多边形的顶点连线可以理解为可通行路径,所以根据顶点的连线生成可见边并加入到局部层可见边集合E_{local}中。另外,由于在规划过程中,基于路径最短的基本原则,可通行路径需要尽可能地在外围绕过障碍物,而一部分可见边的方向是指向障碍物的。这部分冗余边在建图和规划过程中会增加计算量,所以须剔除这些可见边。

如图5所示,红色顶点为多边形的顶点,红色边为多边形的边。蓝色实线和蓝色虚线为不同多边形顶点相连生成的边。蓝色实线所表示的边从外部绕过多边形,所以保留。灰色虚线表示为蓝色虚线的延长线和方向,可以看出蓝色虚线所表示的边是指向多边形内部的,所以剔除这部分边。

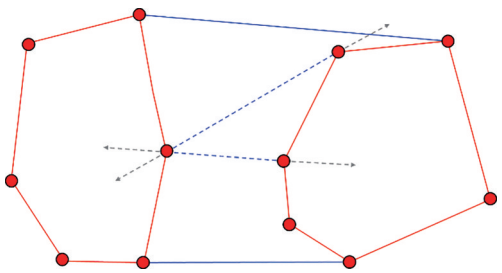

图 5 剔除冗余可见性边原理示意图

在全局层的更新中,首先对局部层和全局层的顶点进行匹配,当局部层和全局层的两个顶点的欧氏距离小于阈值时,将这两个顶点关联,全局层中的点更新位置信息。如果全局层中的某个顶点在局部层中无关联点,那么删除这个顶点。如果局部层中的某个顶点在全局层中无关联点,那么就在全局层中添加这个顶点。在关联局部层和全局层的过程中,为提高鲁棒性,须剔除异常值。对全局层的某个顶点,如果其位置数据能够在连续几帧局部层的迭代中保持不变,或达到迭代次数,那么更新此点坐标,坐标数据为这几帧局部层中坐标的平均值。如果多帧局部层中的某个顶点一直未在全局层中找到对应的关联点,那么将这个点添加到全局层中。在顶点的更新结束之后,相应地,将局部层中的可见边更新到全局层中。

3 基于可视图和广度优先算法的路径规划

基于上节已经生成的可视图,使用广度优先算法(breadth-first search, BFS)进行最短路径寻找。

广度优先搜索算法是一种用于图(或树)的遍历和搜索的基本算法。它从图的起始节点开始,逐层地探索邻接节点,然后再探索邻接节点的邻接节点,以此类推,直到达到目标节点或遍历完整个图。BFS通常用于图的遍历、最短路径查找、连通性检查、拓扑排序等。

对于给定的目标点坐标$position_{\text{goal}}$和当前车辆坐标$position_{\text{veh}}$,首先将这两个点作为顶点添加到全局层中。此时会将两个点分别与全局层中的最近的顶点连接,并生成可见性边。此时的可视图中多边形的顶点可以看作BFS中的节点,而多边形的边和顶点连接生成的边可以看作BFS中的边。在可视图上运行广度优先算法,找到最短通行路径。

具体原理如图6所示。A为起点,F为终点,对节点进行依次遍历,节点的访问顺序为{A,B,C,D,E,F},最终得到的最短路径为{A,B,D,F}。

具体算法如算法3所示。

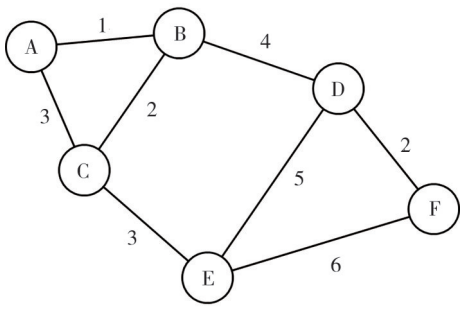

图 6 广度优先算法示意图

```
算法3
输入:可视图、起点、终点
输出:最短路径
1 创建一个空队列Q
2 将起始节点加入队列Q
3 标记起始节点为已访问
4 while Q非空且终点未被访问时 do:
5     当前节点从队列Q中移出
6     处理当前节点
7     if 当前节点等于终点 do
8         结束算法
9     for 当前节点的每个邻居节点 do
10        if 邻居节点未被访问 do
11            将邻居节点加入队列Q
12            标记邻居节点为已访问
13 end for
```

4　实验

实验分为仿真实验和实车实验两部分,其中仿真实验使用与文献[18]相同的仿真平台和实验参数,在一个室内地图中进行实验。实车实验中,使用阿卡曼转向遥控小车,搭载Velodye16线激光雷达。软件平台为Ubuntu20.04和ROS-Noetic。

对于点云地图获取这部分,实验进行定性分析,使用可视化工具RVIZ来观察动态障碍物和地面冗余点剔除优化效果。

对于可视图建图和路径规划这两部分,通过实验进行定量研究。主要通过局部地图顶点数量、全局地图顶点数量、地图更新时间、算法运行时间这4个评价指标进行优化效果的验证。

局部地图顶点数量表示在可视图的局部层中障碍物多边形的顶点数量。全局地图顶点数量表示在可视图的全局层中障碍物多边形的顶点数量。地图更新时间表示可视图建图过程中,从收到的新一帧点云地图到可视图全局层中相应部分更新完成的时间。算法运行时间表示可视图建图和路径规划这两部分算法的总体运行时间。

以上涉及的评价指标,将在仿真实验和实车实验中对程序的优化效果进行验证。

在仿真实验中,通过对照实验选择合适的简化系数。此后,通过“全局地图的顶点数量”,“算法运行时间”这两个评价指标对程序的优化效果进行验证。

在实车实验中,使用RVIZ验证动态障碍物和地面冗余点剔除的效果。此后,通过局部地图顶点数量、全局地图顶点数量、地图更新时间这3个评价指标对程序的优化效果进行验证。

4.1　仿真实验

仿真实验中,首先通过对照实验,选取合适的简化系数,既能对可视图中的多边形进行简化,又能保留多边形的重要特征点。最终的评价标准为遍历整个地图的过程中,全局地图中顶点的数量和算法的运行时间。

在仿真环境的地图中,车辆会按顺序到达地图的一些固定点,以此遍历整个地图。如图7所示,例如在室内环境中,车辆将从起点开始,逐次抵达一些事先设置好的点,以此遍历整个地图,获取整个地图的环境信息。车辆的初始状态为处于未知环境中,通过探索陌生环境收集地图中的障碍物信息。

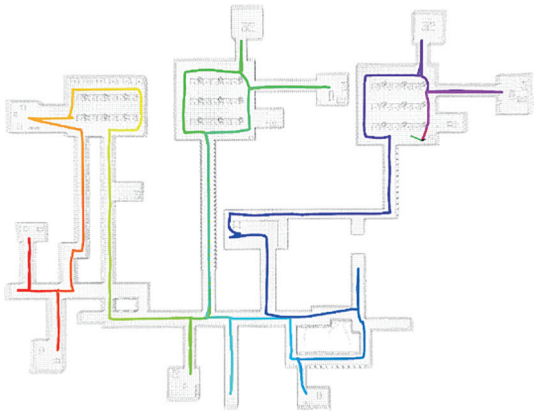

图7　仿真环境遍历示意图

在对照实验中,局部地图和全局地图中的多边形都使用RVIZ进行可视化。通过对比不同简化系数的简化效果,确定最终的简化系数。如图8所示,当没有简化过程时,局部地图中多边形顶点数量非常多,极大增加了计算量。当简化系数为0.005时,进行简化处理得到的地图与原始地图基本一致,并没有得到明显的简化效果。当简化系数为0.01时,局部地图中顶点数量有所下降,全局地图中多边形复杂度也有所下降,但是仍然比较复杂。当简化系数为0.02时,局部地图中顶点数量明显下降,全局地图中的多边形简化效果明显。当简化系数为0.05时,局部地图中顶点数量过少,全局地图中的多边形已经无法正常表示障碍物的轮廓。经过综合对比,取最终的简化系数为0.02,既能对多边形进行简化,又能保留多边形的重要特征点。

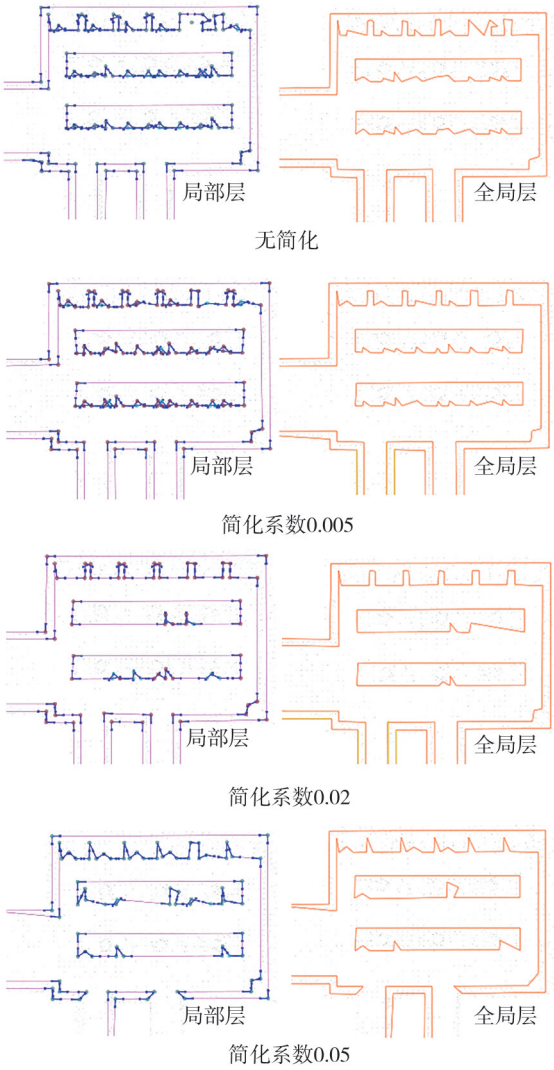

图 8 不同简化系数下的效果图

车辆在遍历地图的标志点时,随着车辆当前位置和目标点位置的变化,局部地图和全局地图不断更新,同时规划算法不断更新从当前位置点到目标点的最优路径,整个过程可以看作若干段的动态路径规划过程。以原始程序和开启简化系数为0.02的程序分别进行实验,并记录算法运行时间。

在建图过程中,通过全局地图顶点数量来评价算法的优化效果。如图9所示,黑色三角形点线表示使用原始程序的多边形顶点数量,在开启简化之后,红色圆形点线全局地图中多边形的顶点数量明显减少。

在可视图建图和路径规划整体过程中,主要使用算法运行时间来评价算法的优化效果。如图10所示,黑色三角形点线表示未经过简化的算法运行时间,红色圆形点线表示开启简化的算法运行时间。

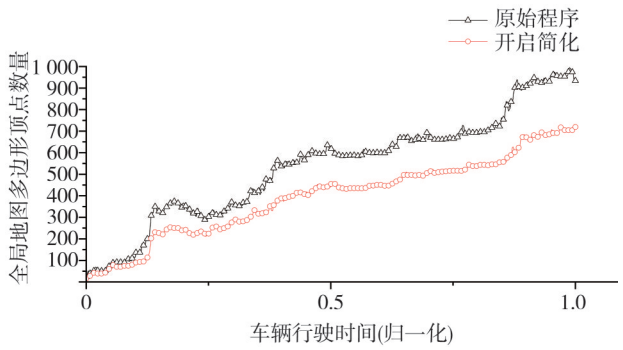

图 9 全局地图顶点数量对比

经过对比可以看出,在开启简化功能之后,整体算法运行时间明显减少,也就是经过多边形简化之后,整个算法的计算量显著减少。

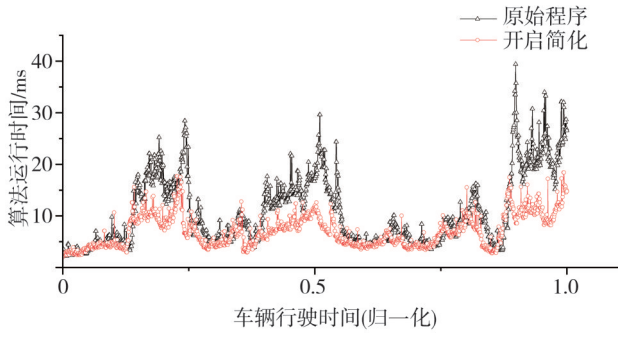

图 10 算法运行时间对比

如表1所示,对于全局地图中的最终顶点数量,使用原始程序的方法平均为934,使用本文方法为719,减少了23.0%。可见本文算法在建图过程中的优化效果明显。

由表1可见,对于可视图建图和路径规划的算法运行时间,使用原始程序的方法为10.68 ms,使用本文方法为7.02 ms,减少了34.3%。本文算法的整体计算量相对于原始算法明显减小。

表 1 实车实验数据对比

项目	原始程序	本文方法	优化效果
全局地图顶点数量	934	719	23%
平均算法运行时间/ms	10.68	7.02	34%

4.2 实车实验

实车实验使用阿卡曼转向遥控小车,搭载Velodye16线激光雷达。图11为本文所使用的实验平台。实验环境为校园环境,障碍物有建筑物、静止的车辆、树木等静态障碍物,另外还有行人、行车等动态障碍物。

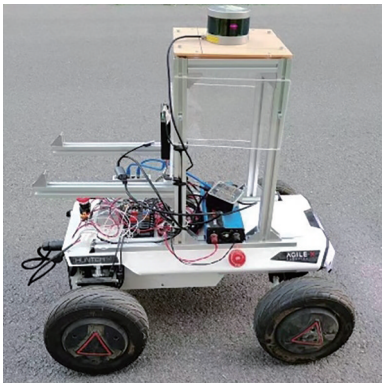

图 11　实验平台

在实车实验中,可以通过 rviz 可视化工具直观地观察到动态障碍物和地面点剔除的效果。最终的评价标准为建图过程中,局部地图顶点数量、全局地图顶点数量和地图的更新时间。

首先使用改进的SLAM算法进行激光雷达点云地图构建,并对动态障碍物和地面冗余点进行剔除,如图 12所示。矩形框中是行人轨迹对应的点云,圆形框中是地面冗余点对应的点云,在经过剔除之后,只保留了静态障碍物的点云。通过对比可以看出,本文方法可以对点云地图中的行人轨迹进行有效剔除,生成可靠的点云地图。

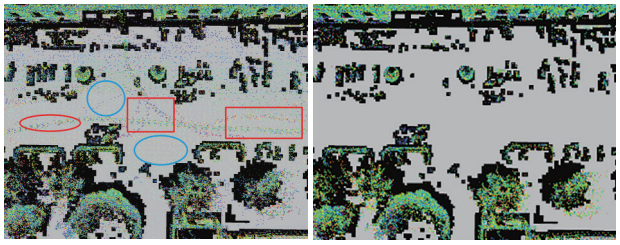

图 12　动态障碍物和地面冗余点剔除

在剔除动态障碍物之后,对点云地图进行多边形的提取和简化,如图 13~图 15 所示。黑色三角形点线表示原始程序的数值,红色圆形点线表示本文算法的数值。可以看出,经过本文方法的简化处理之后,局部地图中顶点数量和全局地图中的顶点数量明显减少,另外地图更新时间也显著减少。

如表2所示,对于局部地图中的平均顶点数量,使用原始程序的方法平均为84.28,使用本文方法为51.42,减少了39.9%。对于全局地图中的最终顶点数量,使用原始程序的方法平均为957,使用本文方法为675,减少了29.5%。对于全局地图的更新时间,使用原始程序的方法为12.04 ms,使用本文

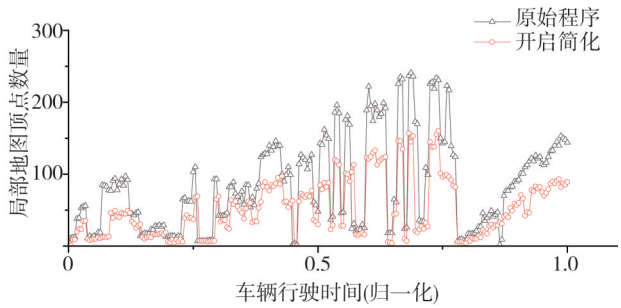

图 13　局部地图顶点数量对比

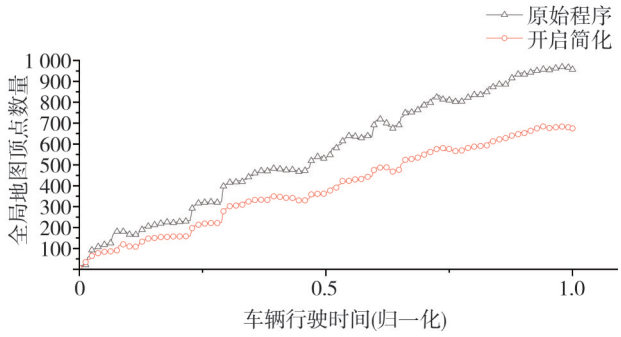

图 14　全局地图顶点数量对比

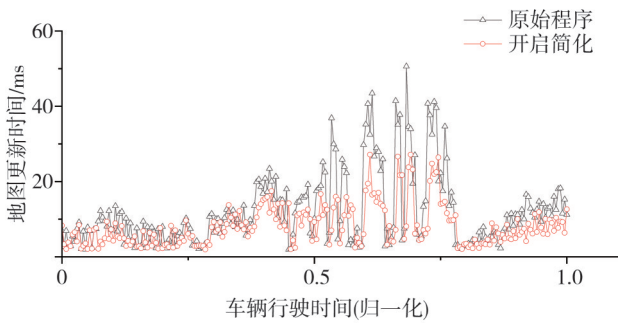

图 15　地图更新时间对比

方法为8.18 ms,减少了32.0%。可以看出,在实际环境的建图过程中,本文算法相对于原始算法优化效果明显。

表 2　实车实验数据对比

项目	原始程序	本文方法	优化效果
平均局部地图顶点数量	84.28	51.42	39.9%
全局地图顶点数量	957	675	29.5%
平均地图更新时间/ms	12.04	8.18	32.0%

5　结论

针对复杂环境中使用可视图方法计算量大、地图复杂的问题,结合SLAM和可视图的方法,提出了

一种简化可视图的建图和规划方法。首先使用改进的SLAM算法生成点云地图,并进行动态障碍物的剔除。接着生成可视图,并基于障碍物的大小和顶点处内凹角的大小对可视图中多边形的复杂边缘进行简化,剔除冗余的顶点。最后通过仿真实验和实车实验证明该方法的可行性和相对原始算法的优化效果。结果表明,本文方法能够有效剔除动态障碍物,获得可靠的点云地图,并在保证建图精度的情况下,对可视图中的障碍物多边形进行简化,减小了建图和规划过程的计算量。在保持搜索最优路径的基础上,减少了路径规划的时间。相对于原始算法,本文方法更有利于建图和规划。

参考文献

[1] 陈慧岩,熊光明,龚建伟,等. 无人驾驶汽车概论[M].北京:北京理工大学出版社,2014.
CHEN H Y, XIONG G M, GONG J W, et al. Introduction to self-driving car [M]. Beijing: Beijing Institute of Technology Press, 2014.

[2] KHATIB O. Real-time obstacle avoidance for manipulators and mobile robots [J]. The International Journal of Robotics Research, 1986, 5(1): 90-98.

[3] LUMELSKY V, STEPANOV A. Dynamic path planning for a mobile automaton with limited information on the environment [J]. IEEE Transactions on Automatic Control, 1986, 31(11): 1058-1063.

[4] COLORNI A, DORIGO M, MANIEZZO V. Distributed optimization by ant colonies[C]. Proceedings of the first European Conference on Artificial Life, 1991: 134-142.

[5] BREMERMANN H J. The evolution of intelligence: the nervous system as a model of its environment[M]. University of Washington, Department of Mathematics, 1958.

[6] CHAN H, TAM K, LEUNG N. A neural network approach for solving the path planning problem[C]. 1993 IEEE International Symposium on Circuits and Systems (ISCAS), 1993: 2454-2457.

[7] 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 and Automation, 1996, 12(4): 566-580.

[8] LAVALLE S M, KUFFNER J J, DONALD B. Rapidly-exploring random trees: progress and prospects[J]. Algorithmic and Computational Robotics: New Directions, 2001, 5: 293-308.

[9] JOHNSON D B. A note on Dijkstra's shortest path algorithm[J]. Journal of the ACM (JACM), 1973, 20(3): 385-388.

[10] 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.

[11] STENTZ A. Optimal and efficient path planning for partially-known environments[C]. Proceedings of the 1994 IEEE International Conference on Robotics and Automation, 1994: 3310-3317.

[12] WANG B, LIU Z, LI Q, et al. Mobile robot path planning in dynamic environments through globally guided reinforcement learning[J]. IEEE Robotics and Automation Letters, 2020, 5(4): 6932-6939.

[13] LOZANO-PéREZ T, WESLEY M A. An algorithm for planning collision-free paths among polyhedral obstacles[J]. Communications of the ACM, 1979, 22(10): 560-570.

[14] 黎萍, 朱军燕, 彭芳, 等. 基于可视图与A*算法的路径规划[J]. 计算机工程, 2014, 40(3): 193-195,200.
LI P, ZHU J Y, PENG F, et al. Path planning based on visibility graph and A* algorithm [J]. Computer Engineering, 2014, 40(3): 193-195,200.

[15] 李霜琳, 何家皓, 敖海跃, 等. 基于鸽群优化算法的火星飞行器智能可视图法[J]. 飞行力学, 2020, 38(5): 90-94.
LI S L, HE J H, AO H Y, et al. Intelligent visibility graph algorithm of Mars aircraft based on pigeon-inspired optimization[J]. Flight Dynamics, 2020, 38(5): 90-94.

[16] OU J, HONG S H, SONG G, et al. Hybrid path planning based on adaptive visibility graph initialization and edge computing for mobile robots[J]. Engineering Applications of Artificial Intelligence, 2023, 126: 107110.

[17] LV T, ZHAO C, BAO J. A global path planning algorithm based on bidirectional SVGA[J]. Journal of Robotics, 2017, 2017.

[18] YANG F, CAO C, ZHU H, et al. FAR planner: fast, attemptable route planner using dynamic visibility update [C]. 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022: 9-16.

[19] LI Q, XIE F, ZHAO J, et al. FPS: fast path planner algorithm based on sparse visibility graph and bidirectional breadth-first search[J]. Remote Sensing, 2022, 14(15): 3720.

[20] SHAN T, ENGLOT B. LeGO-LOAM: lightweight and ground-optimized lidar odometry and mapping on variable terrain [C]. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018: 4758-4765.

[21] DOUGLAS D H, PEUCKER T K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature[J]. Cartographica: the International Journal for Geographic Information and Geovisualization, 1973, 10(2): 112-122.