DOI: 10.19650/j.cnki.cjsi.J2513933

基于改进 JPS 的智能车路径规划策略研究*

朱振东,姚强强,石艺恒,谢麒麟
(青海大学机械工程学院 西宁 810016)

摘 要:跳点搜索是一种应用于路径规划的快速的图搜索算法。针对现有基于改进 JPS 算法的路径规划过程中,跳点搜索过程复杂、节点扩展数量较多导致搜索效率较低,为此提出一种基于改进 JPS 的全局路径规划策略,包括减少冗余节点扩展的方法以及提出安全平滑路径生成策略。首先,进行搜索方向的优先级排序,调整各个可能移动方向的优先级,优先探索指向目标的方向,依次按照该优先级排序扩展其他节点寻求跳点。其次,结合路径优化策略,包括安全节点更新、冗余节点消除及路径平滑等策略,确保路径的安全性与平滑性。安全节点更新策略减少危险路径,冗余节点消除策略有效降低路径长度,路径平滑策略通过三次准均匀 B 样条曲线处理改善路径的平滑度。最后,通过仿真与真实场景的试验验证改进算法的性能,仿真结果表明,改进的 JPS 算法在复杂环境中相较于传统的 JPS 算法和 A* 算法,分别减少了 19.0% 和 99.92% 的搜索时间,Improved-JPS 算法的扩展节点数相较于 JPS 减少了 56.9%,相较于 A* 算法减少了 98.9%。在更复杂的实际环境中,ROS 智能车在实际环境中的实验结果表明,搜索时间相较于 A* 算法提高了约 20.5%。相较于 JPS 提高了约 28.0%,有效提高复杂场景下车辆路径规划效率和安全性。验证了 Improved-JPS 算法的有效性和优越性。

关键词: 智能车路径规划;跳点搜索;优先级排序;路径优化;安全策略
中图分类号: TH242 TN96 **文献标识码**: A **国家标准学科分类代码**: 510.80

Research on intelligent vehicle path planning strategy based on improved JPS

Zhu Zhendong, Yao Qiangqiang, Shi Yiheng, Xie Qilin
(*School of Mechanical Engineering, Qinghai University, Xining 810016, China*)

Abstract: Jump point search is a fast graph search algorithm widely used in path planning. However, the complexity of the hopping point search process and the excessive number of node extensions can lead to low search efficiency. To address these limitations, this paper proposes a global path planning strategy based on an improved JPS algorithm. The proposed approach includes a method to reduce redundant node extensions and introduces a safe and smooth path generation strategy. First, a directional priority ranking is introduced, where the search directions are adjusted to prioritize movement toward the goal. Nodes are then expanded sequentially according to this ranking to search for jump points more efficiently. Second, several path optimization strategies are combined, including safe node updating, redundant node elimination and path smoothing to ensure the safety and smoothness of paths. The safe node update strategy reduces the dangerous paths, the redundant node elimination strategy effectively reduces the path length, and the path smoothing strategy improves the smoothness of the paths by three times quasi-uniform B-spline curve processing. Finally, the performance of the improved algorithm is verified through simulation and real scenarios. The simulation results show that the improved JPS algorithm reduces the search time by 19.0% and 99.92% in complex environments compared to the traditional JPS algorithm and A* algorithm, respectively, and the number of expansion nodes of the improved-JPS algorithm is reduced by 56.9% compared to the JPS and 98.9% compared to the A* algorithm. 98.9%. In more complex real-world environments, tests conducted on a ROS-based intelligent vehicle demonstrate search time improvements of approximately 20.5% over A* and 28.0% over JPS. These results confirm that the proposed Improved-JPS algorithm significantly enhances the efficiency and safety of path planning in complex scenarios, validating its effectiveness and superiority.

Keywords: intelligent vehicle path planning; jump point search; priority ranking; path optimization; safety strategy

收稿日期:2025-04-16 Received Date: 2025-04-16
*基金项目:青海大学中青年科研基金(2023-QGY-15)、青海大学大学生科研训练计划(SRT202541)项目资助

0 引　　言

路径规划作为无人驾驶技术领域的重要组成部分,是全球无人驾驶领域研究的热点之一[1]。在静态环境中,路径规划的主要目标是为某一移动主体找到一条从起点到目标点的路径,有现有的地图,规划一条通过固定障碍物的全局路径[2]。

根据不同的场景或目标,路径规划分为全局路径规划和局部路径规划。目前,学者针对不同行驶场景,研究了全局路径规划方法主要有 RRT(rapidly exploring random trees, RRT)[3]、RRT*[4-5]、概率路线图 PRM(probabilistic roadmap, PRM))[6]、Dijkstra 算法[7]、A*算法[8]、D*算法[9]等被广泛应用。基于节点的算法主要包括 Dijkstra、A*算法等。标准的 A*算法需要从起点开始,不断扩展相邻节点,进而利用目标代价函数搜索目标节点[10]。然而,当环境复杂、地图规模大时,A*算法可能会需要大量的内存和时间来进行搜索[11]。Harabor 等[12]在保留维持 A*算法框架的基础上,建立在 A*算法的评价函数的基础上,忽略无用节点,只保留关键节点,提高了 A*算法的搜索效率,这种算法称为跳点搜索(jump point search, JPS)算法。可以大大减少搜索过程的时间消耗,在复杂地图环境中的最短路径求解中得到广泛应用。然而,该算法存在过多的转折点,导致理论优化未能在实践中体现,路径平滑性较差。

针对上述问题,Zheng 等[13]通过新的跳点定义和节点域矩阵,改进了 JPS 算法,增强了机器人在路径执行过程的安全性。陈奕虎等[14]设计了一种由对角线距离和方向信息结合的启发式函数用于提高寻路效率,并且进一步对规划路径进行平滑处理以有效兼顾规划路径的安全可靠性。孙凌宇等[15]提出跳点优化蚁群算法(jump point optimization ant colony algorithm, JPOACA),通过引入跳点搜索筛选低成本邻域并扩展搜索范围,同时在跳点处补充信息素,以提升路径质量并加快收敛速度。Zhou 等[16]将 JPS 算法应用于复杂城市场景中下自动驾驶汽车的局部路径规划,能够以较低的时间复杂度搜索最优且安全的路径。周熙栋等[17]在结合 A*搜索的基础上改进 JPS 算法,并建立搜索策略切换规则,结构层与非结构层使用不同的搜索策略,从而有效缩短搜索时间。Chen 等[18]将 JPS 算法与蚁群算法结合,提升了大量障碍物场景下路径搜索的效率和准确性,在获得最优路径的同时保证了搜索效率。Hou 等[19]引入人工势场算法,为 JPS 算法提供参考方向,减少了无用跳点的数量,提高搜索效率。Su 等[20]考虑到 APF 的影响和方向图引导,计算路径规划的扩展方向优先级,这指导和改善了后续跳跃点的搜索。刘荣华等[21]提出一种改进双向动态 JPS 算法,引入了动态约束椭圆,限制节点扩展范围,加快收敛速度,在路径搜索时间和内存消耗方面均优于原始 JPS 算法。Huang 等[22]改进了跳点选择和障碍扩展规则,获得了安全性能更高的路径。田雅琴等[23]结合了跳点搜索和改进遗传算法,利用 JPS 的局部搜索和遗传算法的全局搜索能力,提升路径规划的效率。结果显示跳点搜索—遗传算法(jump point search-genetic, JPSG)在准确性、速度和稳定性方面优于传统算法。刘晓茜等[24]通过采用双向交替搜索策略提升了算法效率,并利用位运算快速识别路径中的关键点。同时,在评价函数中引入了高度参数,以便更准确地评估下一节点,从而显著减少了搜索过程中的冗余节点。人们已经完成了大量基于 JPS 的路径规划的研究[25-28]。然而,以上基于改进 JPS 算法的路径规划研究,跳点搜索过程较为复杂,扩展节点仍然较多,导致搜索效率较低。其次,现有大部分研究针对的环境较为简单,而在复杂环境中,障碍物可能迫使算法绕道,这意味着即使某个方向当前看似不优,后续可能成为唯一有效的路径。因此,仍需考虑有效路径的扩展,优先处理看似更好的方向,同时不遗漏潜在的有效路径。

针对上述问题,本研究提出一种新的基于改进 JPS 的路径规划策略,研究了跳点搜索方向和冗余节点优化方法,提出安全平滑路径生成策略,设计了基于 JPS 的智能车高效安全的全局路径规划算法,提高智能车全局路径规划的效率和安全性能。本研究的主要贡献在于:1)提出了一种基于方向优先级调整的路径搜索策略,JPS 的跳点搜索能够避免一些冗余的搜索方向,尤其是在大规模网格或复杂地形中,能够显著减少不必要的搜索节点和跳点扩展次数,使搜索更加高效、精准。2)提出路径规划安全节点更新策略,从路径数据中剔除冗余节点,大大减少控制点和路径总长度,保证规划路径的安全性。3)提出一种新的基于改进 JPS 的路径规划策略,提高复杂场景下车辆路径规划效率和安全性。开展试验研究,验证了算法的有效性和优越性。

1 跳点搜索逻辑

1.1 环境建模

环境建模是为路径规划算法提供一个准确的、可操作的环境表示,以便车辆能够根据这个模型作出决策,本文选用栅格法进行环境建模,如图 1 所示。格栅地图中白色方块表示可以自由通过的区域,黑色方块表示不可通行的障碍物。

只要被障碍物涉及的区域,无论大小,均用黑色方块表示。利用栅格法,车辆行驶环境模型可以描述为:

$$\begin{cases} x_i = a \cdot (\mathrm{mod}(i-1, N_x) + 0.5) \\ y_i = a \cdot (N_y - \mathrm{ceil}(i/N_x) + 0.5) \end{cases} \tag{1}$$

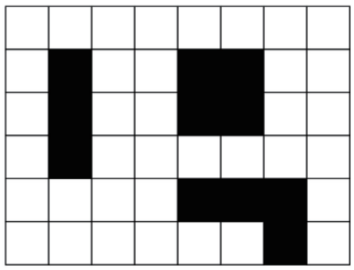


图 1 栅格地图

Fig. 1 Raster map

式中：(x_i, y_i) 表示第 i 个栅格的位置坐标，x_i 是该栅格在水平方向上的坐标，y_i 是该栅格在垂直方向上的坐标；mod( ) 表示取余运算，ceil( ) 表示取整运算，这些运算通常用于将实际坐标映射到栅格地图中的栅格索引；N_x 是栅格地图的行数；N_y 是栅格地图的列数；a 栅格的边长，表示每个栅格在实际地图中占据的物理空间长度。

1.2 路径搜索过程

跳点搜索算法的主要思路是先通过搜索确定地图中的跳点，再利用 A* 算法找到最优路径。图 2 展示了 JPS 算法的路径搜索过程示意图。根据裁剪得到的自然节点和强迫节点，JPS 算法通过直线跳跃和对角跳跃来加速路径搜索。在探索路径的过程中，JPS 会沿着某一方向递归地寻找跳点，直到发现下一个关键的跳跃点为止。每当发现跳点时，该跳点就会被加入开放列表，算法随后会从这个新发现的跳点继续递归地沿所有可能的方向寻找其他跳点。

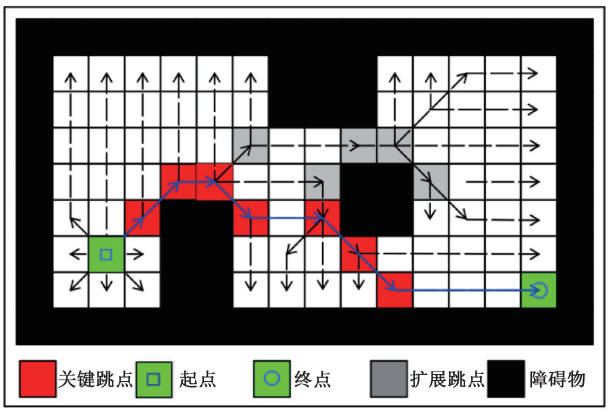


图 2 JPS 算法路径搜索过程

Fig. 2 Schematic diagram of the path search process of JPS algorithm

JPS 算法根据启发式函数 $F(n)$ 的值来选择下一个扩展节点，启发式函数的表达式为：

$$H(n) = \sqrt{2} \cdot \min(|x_g - x_n|, |y_g - y_n|) + \max(|x_g - x_n|, |y_g - y_n|) \quad (2)$$

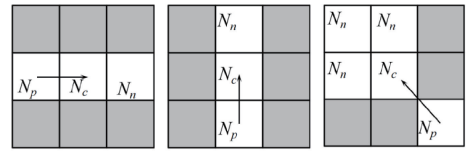
式中：$F(n)$ 由 2 部分组成；$G(n)$ 是指从起始位置经过一系列中间跳点到达当前跳点 N_c 的实际距离，如式(5)所示；$H(n)$ 是指从跳点 N 到达目标点 $N_g(x_g, y_g)$ 的估计距离，$H(n)$ 愈趋近于空间中两点的真实距离，启发式函数的效率愈高。所以选取的是对角线距离，如式(6)所示。

$$G(n) = \sum_{i=1}^{n} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \quad (3)$$

$$H(n) = \sqrt{2} \cdot \min(|x_g - x_n|, |y_g - y_n|) + \max(|x_g - x_n|, |y_g - y_n|) \quad (4)$$

1) 邻居裁剪规则

邻居裁剪规则用于筛选出需要拓展的节点，具体规则如图 3 所示，N_p 表示父节点(nodeparent)，N_c 表示当前节点(nodecurrent)，N_n 白色栅格为自然节点(nodenature)，拓展查询自然节点。



(a) 横向移动 (a) Lateral movement　(b) 纵向移动 (b) Longitudinal movement　(c) 斜方向移动 (c) Diagonal movement

图 3 无障碍物时搜索示意图

Fig. 3 Schematic diagram of searching when there is no obstacle

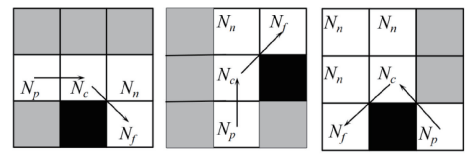
如图 3 所示，当节点周围不存在障碍物时的裁剪规则可用式(5)表示。

$$Len(\langle N_p, N_c, N_n \rangle) < Len(\langle N_p, \cdots, N_n \rangle \mid N_c) \quad (5)$$

其中，不等式的左侧表示从父节点 N_p 通过节点 N_c 到达节点 N_n 的代价距离，而右侧表示从父节点 N_p 经过其他节点到达节点 N_n 的代价距离。如果节点 N_n 满足式(5)，则 N_n 被定义为自然节点。

图 3(a)和(b)分别为横、纵向的移动，图 3(c)为斜方向(对角线)方向的移动。

强迫邻居是指在当前节点附近存在障碍物时(如图 4 所示)，由于障碍物的影响，通往邻居节点 N_f 的路径发生了改变。在这种情况下，显然是最短路径中的关键节点。因此，在没有障碍物的情况下，除了自然邻居之外，还需要额外添加一个邻居节点，这个节点被称为强迫邻居。



(a) 横向移动 (a) Lateral movement　(b) 纵向移动 (b) Longitudinal movement　(c) 斜方向移动 (c) Diagonal movement

图 4 有障碍物时搜索示意图

Fig. 4 Schematic diagram of searching when there is an obstacle

若节点 N_f 是 N_c 的邻居,节点 N_f 的邻居有障碍物,并且从父节点 N_p 经过 N_c 到 N_f 的路径长度比其他任何从 N_p 到 N_f 且不经过 N_c 的路径短,则为 N_c 的强迫邻居,强制邻居裁剪条件为如式(6)所示。

$$Len(\langle N_p, N_c, N_f\rangle) < Len(\langle N_p, \cdots, N_n\rangle \mid N_c) \tag{6}$$

2)跳点剪枝规则

保留了邻居节点中的最短路径节点。然而,并不是所有的自然邻居都是最终的路径节点。对于目标路径,只有更改路径方向的节点才具有展开的值。修剪跳点的目的是将没有展开跳点的中间跳点替换为远程跳点。跳点的剪枝规则为:

(1)当自然邻居是起点或目标点时,必须是跳点;

(2)自然邻居至少有一个强制邻居。此时,自然邻居和强制邻居是跳点;

(3)在对角线方向搜索时,如果自然邻居的水平或垂直方向上有一个跳点,对角线上的自然邻居也是一个跳点。

2　改进的跳点搜索策略

2.1　路径搜索方向优化

根据高优先级方向的搜索结果,动态决定是否需要扩展次优先级方向,提高搜索效率。优化搜索方向是为了使算法更加高效地朝着目标前进。这种调整是通过改变搜索过程中考虑的移动方向的优先级来实现的。具体来说,根据当前节点与目标节点之间的相对位置,调整各个可能移动方向的优先级,优先探索指向目标的方向。如图 5 所示。

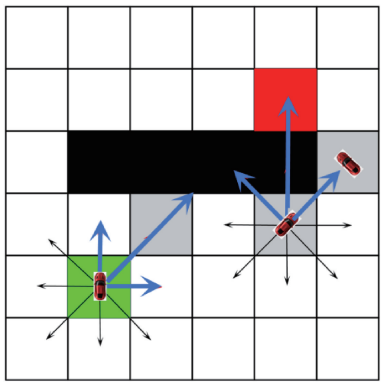


图 5　方向优先级搜索示意图

Fig. 5　Schematic diagram of direction prioritization search

计算目标方向向量后,对于每个可能的移动方向,计算该方向与目标方向向量的点积。点积的结果与两个向量长度的乘积相关,该值可以用来确定两个向量之间夹角的余弦值。

$$\cos\theta = \frac{(x_n - x_f)\times(x_g - x_f) + (y_n - y_f)\times(y_g - y_f)}{\sqrt{(x_n - x_f)^2 + (y_n - y_f)^2}\times\sqrt{(x_g - x_f)^2 + (y_g - y_f)^2}} \tag{7}$$

式中:(x_n, y_n)、(x_f, y_f)、(x_g, y_g)分别为当前节点 n、父节点 f、目标节点 g 的坐标。

对于每一个可能的移动方向,算法计算该方向与目标方向的点积。根据计算出的角度(即每个方向与目标方向的接近程度),对矩阵中的所有可能移动方向进行排序。角度最小的方向(即最接近目标方向的)被放置在列表的前面。

$$v_{move} = \underset{v_{move}}{\operatorname{argmin}}\left(\arccos\left(\frac{v_{target}\cdot v_{move}}{|v_{target}|\cdot|v_{move}|}\right)\right) \tag{8}$$

式中:v_{move} 从小到大排序,选择前 3 个方向作为高优先级方向,其余方向为次优先级方向,选择的 3 个最优移动方向;arccos 表示计算方向之间的夹角;$v_{target}\cdot v_{move}$ 表示计算目标方向与移动方向之间的内积;$|v_{target}|\cdot|v_{move}|$ 表示向量的模(长度)。

改进的 JPS 算法迭代过程如算法 1 所示。

算法 1:改进的 JPS 算法迭代过程

```
while Open ≠ Ø do
    n ← argmin_{n' ∈ Open} f(n')//f(n')=g(n')+h(n')
    Open ← Open \ {n}
    if n = Goal then
        构建路径并终止
    end if
    Closed ← Closed ∪ {n}
    D_priority ← Sort_Directions(n, Goal)
    for each d ∈ D_priority do
        J ← Jump(n, d)
        if J 是有效跳点 且 J ∉ Closed then
            g(J) ← g(n)+distance(n, J)
            f(J) ← g(J)+h(J)
            if J ∉ Open or g(J) < g_旧(J) then
                Open ← Open ∪ {J}
                came_from(J) ← n
            end if
            if d 为高优先级方向 then
                停止扩展低优先级方向
                break
            end if
        end if
    end for
end while
```

进行搜索方向的优先级排序,依次按照该优先级排序扩展其他节点寻求跳点,最优的 3 个方向为高优先级,其余方向为次优先级。如果高优先级方向的扩展找到有效跳点(有效跳点定义为,扩展的跳跃点周围至少 5 个方向是畅通的,则为有效跳点)。则停止其他次优先级方向的扩展,并进行后续跳点的扩展,当高优先级所有方向都没有有效跳点时,则切换到第 2 优先级方向。重复该过程,直到找到目标点为止。通过上述方法,可以增强跳点的目标方向性,避免在搜索过程中部分跳点向远离目标点的方向扩展,从而有效减少扩展跳点的数量,提高路径规划的效率。

为了确保算法的高效性和完备性,同时避免陷入循环或局部陷阱。加入闭集管理机制,通过记录已扩展的节点,防止算法重复处理同一节点,节省计算资源。闭集确保每个节点仅被扩展一次,避免陷入无意义的循环或被障碍物包围的局部区域。

通过识别并跳过网格中的冗余节点,JPS 仅扩展必要的跳点,极大地减少了需要处理的节点数量。这在处理大规模或高密度的网格图时,显著提升了搜索速度和效率。在扩展节点时,改进的 JPS 仅选择启发式距离最小的前 3 个方向进行跳跃。这一策略进一步减少了不必要的节点扩展,避免了在不利方向上的资源浪费,从而优化了整体搜索过程。

2.2 路径优化策略

1)安全节点更新策略

为了降低路径中的危险性,当搜索时存在危险路径,对这个危险路径的 2 个路径点进行更新。路径优化需要保证所有点位于自由空间 S_{free},定义为:

$$S_{free} = \{p \in \mathrm{R}^2 \mid \mathrm{dist}(p,\mathrm{Obstacle}) \geqslant \delta\} \tag{9}$$

式中:δ 为安全距离,表示点 p 与障碍物的最小容许距离;$\mathrm{dist}(p,\mathrm{Obstacle})$ 点 p 到障碍物的最近距离。

安全节点更新策略的核心在于对路径中各个节点及其邻居节点进行分析,以寻找更“安全”的替代节点。具体而言,算法检查路径中相邻的节点对,并尝试找到二者的共同邻居。只有在这些共同邻居远离障碍物时,才会将原来的两节点替换为一个更安全的邻居点。对路径中的每对相邻点 (p_i,p_{i+1}) 检查是否存在一个共同安全点 p_c,满足式(10)。

$$N_{common} = N(p_i) \cap N(p_{i+1}) \tag{10}$$

式中:$N(p) = \{q \mid q$ 是 p 的直接邻居,且 $q \in S_{free}\}$,直接邻居即为对 4 个基本方向进行单位移动后的点集,以寻找路径中的共同安全点。

如图 6 所示,当路径可能直接接触障碍物时,为优化这些危险点,算法通过检查路径中每对存在危险路径的相邻点,判断是否可以替换相邻点为共同的安全邻居点并确保该移动不会碰到障碍物。

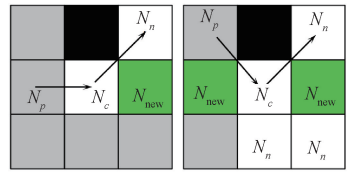


图 6 安全更新节点示意图

Fig. 6 Schematic diagram of security update node

这种安全节点更新策略通过替换相邻点为共同的安全邻居点,能够显著降低路径在拐角或障碍物附近的碰撞风险,从而提高路径规划的稳健性和安全性。

2)冗余节点消除策略

在路径规划的过程中进一步优化和简化路径的工具,通过减少路径上的转折和点数,使路径更加直接和高效,同时保持路径的安全性和可行性。主要作用是通过检查路径上连续的点是否可以直接连接,从而继续简化路径。

$$n = \max(|x_{i+1} - x_{i-1}|, |y_{i+1} - y_{i-1}|) \tag{11}$$

在路径生成的过程中,检查式(11)各个点满足下面条件:通过生成直线上的点,检查直线上的点是否与障碍物相撞。

$$\begin{cases} \Delta x = \dfrac{x_{i+1} - x_{i-1}}{n} \\ \Delta y = \dfrac{y_{i+1} - y_{i-1}}{n} \\ x_i = x_{i-1} + i \cdot \Delta x \\ y_i = y_{i-1} + i \cdot \Delta y \end{cases} \tag{12}$$

式中:$i=0,1,\cdots,n$;(x_{i-1},y_{i-1}) 是当前检查 p_{i-1} 的坐标。

如图 7 所示,考虑一条初始路径包含如下节点,如果节点 A 和 C 之间可以直接连线且不碰到障碍物,那么节点 B 就是冗余的,可以直接删除。通过这种方式,最终路径可能只包含 A → C → F→ G→ I,避免了不必要的转折和冗余的中间节点。

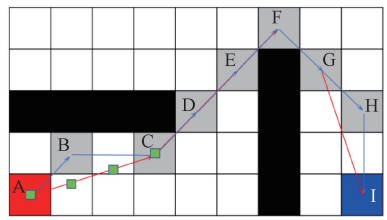


图 7 冗余节点消除示意图

Fig. 7 Schematic diagram of redundant node elimination

这个过程会递归地检查路径中的所有节点,最终生成一条更加简洁、平滑的路径安全节点更新策略与冗余跳点删除策略的结合,使得该方法不仅适用于简单环境,也能在复杂障碍环境中规划出安全且高效的路径。

3)路径平滑策略

在跳点处存在尖锐拐点,得到的路径为连续线段,曲率突变严重,很可能影响车辆的实际运行稳定性,不符合车辆实际路径规划作业。三次 B 样条曲线具有二阶导数连续性,避免了曲率的突变,并提供了足够的光滑性。k 阶 B 样条曲线函数表达式为:

$$C(x)=\sum_{i=0}^{n}P_iN_{i,k}(x) \tag{13}$$

式中:$p(i=0,1,2,3,\cdots,n)$ 为控制点,这些控制点用于定义样条曲线的走向、界限范围;$N_i,k(x)$ 为第 i 个 k 阶 B 样条基函数。基函数具有如下德布尔-考克斯递推式如式(14)所示。

$$N_{i,k}(x)=\begin{cases}\begin{cases}1, & x_i\leqslant x<x_{i+1}\\ 0, & \text{其他}\end{cases}, & k=1\\ [2ex]\dfrac{x-x_i}{x_{i+k-1}-x_i}N_{i,k-1}(x)+\\ \dfrac{x_{i+k}-x}{x_{i+k}-x_{i+1}}N_{i+1,k-1}(x), & k\geqslant 2\end{cases} \tag{14}$$

由于 k 值表示曲线的平滑程度,其值越大,曲线的平滑程度越高,但计算复杂度也随之增加。为平衡路径的平滑度和计算复杂度,选择 $k=3$,从而得到三次 B 样条曲线的基函数如使(15)所示。

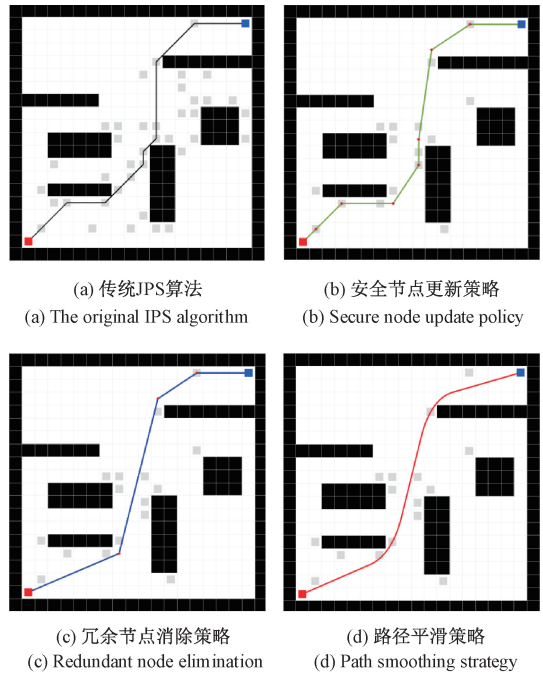
$$\begin{cases}N_{0,3}(x)=\dfrac{1}{6}(-x^3+3x^2-3x+1)\\ N_{1,3}(x)=\dfrac{1}{6}(3x^3-6x^2+4)\\ N_{2,3}(x)=\dfrac{1}{6}(-3x^3+3x^2+3x+1)\\ N_{3,3}(x)=\dfrac{1}{6}x^3\end{cases} \tag{15}$$

则三次 B 样条曲线段为:

$$C_{0,3}(x)=C_0N_{0,3}(x)+C_1N_{1,3}(x)+C_2N_{2,3}(x)+C_3N_{3,3}(x) \tag{16}$$

利用改进算法找到的初始路径,将路径跳点作为控制点集,通过式(16)计算出三次 B 样条曲线点。经过分段拟合后,将分段曲线拼接为完整的 B 样条曲线,生成最终平滑路径。

如图 8 所示,展示了 4 种路径规划策略的比较,每个子图代表不同的算法策略。图 8(b)中的安全节点更新策略这种策略确保路径的安全性,以避免潜在的危险路径。图 8(c)中的冗余节点删除策略删除路径中的冗余节点,通常会使路径更短且更直接。图 8(d)中的路径平滑策略采用平滑算法来优化路径,使得路径更为平直和平滑,减少了急剧的转弯和不必要的路径变化。每个子图展示了不同的路径规划方法如何在同样的地图环境中产生不同的结果。



(a) 传统JPS算法 (a) The original IPS algorithm
(b) 安全节点更新策略 (b) Secure node update policy
(c) 冗余节点消除策略 (c) Redundant node elimination
(d) 路径平滑策略 (d) Path smoothing strategy

图 8　改进 JPS 过程示意图

Fig. 8　Schematic diagram of the improved JPS process

3　仿真与实验分析

3.1　仿真验证

为验证改进 JPS 算法的有效性,分别采用不同环境的栅格地图进行仿真实验,栅格地图展示了 3 种算法的的路径规划结果对比(上排为简单地图,下排为复杂地图),分别是 A* 算法、JPS 算法和本文改进 Improved-JPS 算法对比。其中黑色为障碍物,白色为可通行区域。图 9 和 10 分别显示了在规模在 20×20 栅格地图上的路径规划结果。由于简单地图没有陷阱与局部最优的问题,A* 算法和 JPS 算法搜索速度还是比较快的,但依然生成了大量无用的扩展节点。

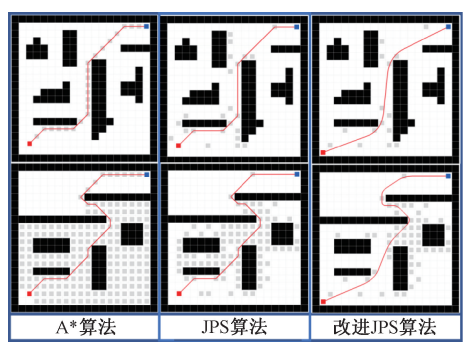


图 9　不同算法在 20×20 地图上的路径规划结果

Fig. 9　Planning results of different algorithms on 20×20 map

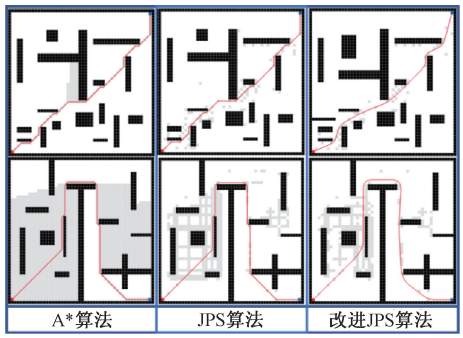


图 10 不同算法在 50×50 地图上的路径规划结果

Fig. 10 Planning results of different algorithms on 50×50 map

以下不同场景中,均是进行 100 次的仿真试验,取 100 次搜索时间平均值作为评价指标,其余路径长度,扩展节点数等指标均是一些路径规划的常用的评价指标。如表 1 所示,在 20×20 规模的复杂栅格地图(下排)中,在搜索时间上 Improved-JPS 算法的平均搜索时间为 0.113 0 s,相较于 JPS 减少了约 20.6%,相较于 A* 算法减少了 47.1%,展现了卓越的效率。Improved-JPS 算法的平均路径长度降低为 30.26,比前两种算法略短。改进的 JPS 算法扩展的节点数大幅减少,说明其在搜索空间上的优化效果显著,避免了大量冗余计算。

表 1 20×20 地图下仿真搜索数据

Table 1 Simulated search data under 20×20 map

20×20 规模	算法	搜索时间/s	路径长度	扩展节点数	路径点数
简单地图	A* 算法	0.021 7	27.54	24	24
	JPS 算法	0.023 1	27.54	26	7
	Improved-JPS 算法	0.034 5	26.20	16	4
复杂地图	A* 算法	0.213 8	31.79	193	27
	JPS 算法	0.142 4	31.79	150	14
	Improved-JPS 算法	0.113 0	30.26	36	8

如表 2 所示,随着地图规模的增大,在 50×50 规模的栅格地图中,A* 算法与 JPS 算法性能逐渐下降,Improved-JPS 在搜索时间上降低到 0.135 4 s,比 JPS 减少了 62.0%,比 A* 算法减少了 97.9%,改进的 JPS 算法在搜索时间上有显著优势。Improved-JPS 的平均路径长度为 108.85,略短于前两种算法,路径优化了约 0.53%。Improved-JPS 算法扩展节点数为 322,比 JPS 减少了 50.8%,比 A* 算法减少了 74.2%。改进的 JPS 显著减少了扩展的节点数,表明其在搜索空间上的效率得到了大幅优化,有效避免了冗余计算。

表 2 50×50 地图下仿真搜索数据

Table 2 Simulated search data under 50×50 map

50×50 规模	算法	搜索时间/s	路径长度	扩展节点数	路径点数
简单地图	A* 算法	0.112 1	73.49	87	60
	JPS 算法	0.082 3	73.49	104	29
	Improved-JPS 算法	0.044 5	71.85	61	9
复杂地图	A* 算法	6.549 1	109.43	1 248	98
	JPS 算法	0.356 2	109.43	654	22
	Improved-JPS 算法	0.135 4	108.85	322	8

在图 11 中,对于大规模有陷阱的地图,由于改进跳点搜索策略的优化,减少了大量的无用扩展节点。此外,Improved_JPS 算法的安全策略与冗余节点剔除策略也是十分明显。表明 Improved-JPS 算法在保持高效率和高安全性的同时,路径质量同样出色,展现了其在综合性能上的全面优势。

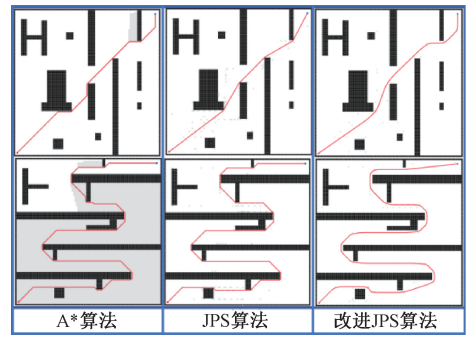


图 11 不同算法在 100×100 地图上的路径规划结果

Fig. 11 Planning results of different algorithms on 100×100 map

如表 3 所示,在大规模的复杂地图中,Improved-JPS 平均搜索时间为 0.067 4 s,相较于 JPS 减少了 19.0%,相较于 A* 算法减少了 99.92%,改进的 JPS 算法在搜索时间上表现出压倒性的优势。Improved-JPS 算法的平均路径长度为 340.56,比前两种算法缩短了约 1.66%。Improved-JPS 算法的扩展节点数进一步减少到 66,相较于 JPS 减少了 56.9%,相较于 A* 算法减少了 98.9%,改进的 JPS 算法在搜索空间上进一步优化,大幅减少了扩展节点数,有效提升了算法的效率。

JPS、Improved-BJPS 和 Improved-JPS 算法路径对比如图 12 所示。Improved-JPS 算法充分体现了在减少冗余节点方面的高效性。算法实现了零危险边保证了路径规划的安全性。试验也表明 Improved-JPS 算法在保持高效率和高安全性的同时,路径质量同样出色,不仅显著优于 A* 算法,也优于 JPS,展现了其在综合性能上的全面优势。

表 3　100×100 地图下仿真搜索数据

Table 3　Simulated search data under 100×100 map

100×100 规模	算法	搜索时间/s	路径长度	扩展节点数	路径点数
简单地图	A* 算法	0.141 6	144.30	218	113
	JPS 算法	0.041 2	144.30	56	11
	Improved-JPS 算法	0.035 6	137.41	48	6
复杂地图	A* 算法	85.352 0	346.30	6 244	315
	JPS 算法	0.083 2	346.30	153	28
	Improved-JPS 算法	0.067 4	340.56	66	22

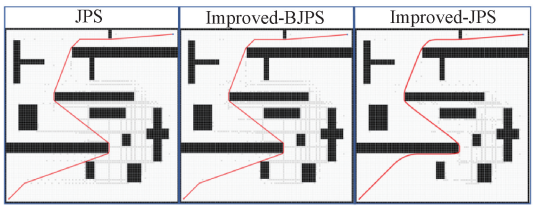


图 12　与改进双向 JPS 对比

Fig. 12　Comparison with improved bidirectional JPS

文献[29]中以最小实际代价节点中间无障碍物结束搜索的方式能提高搜索效率,搜索时间为 0.065 6 s,而传统的 JPS 的搜索时间为 1.084 1 s,Improved-BJPS 算法在减少冗余扩展节点方面的高效性,而对于有陷阱的大规模地图,Improved-JPS 算法表现更优,搜索时间仅为 0.054 2 s。试验也表明 Improved-JPS 算法在保持高效率和高安全性的同时,路径质量同样出色。

3.2　智能小车试验

为验证运动规划算法在真实环境中的有效性和可行性,在 Ubuntu 20.04 操作系统的 ROS noetic 系统通过相应传感器获取周围环境信息。实验设备如图 13 所示。用户的 PC 通过 Wi-Fi 实现对控制计算机的远程连接,利用工控机来操作 ROS 指令,构成小车上位机平台。

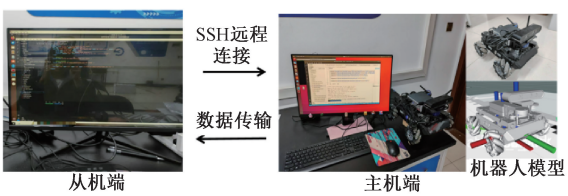
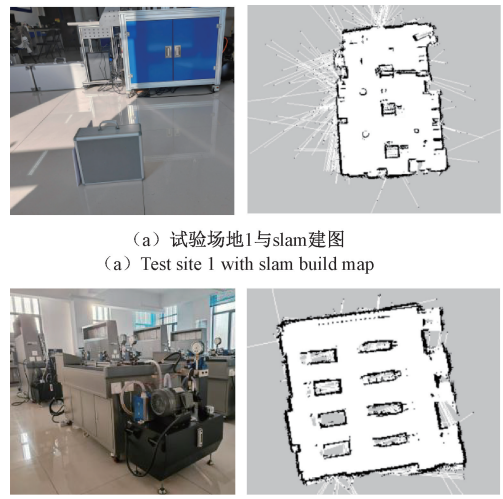


图 13　实验设备

Fig. 13　Experimental equipment

利用 Gmapping 建图算法与 AMCL 定位算法构建环境地图。试验场地如图 14 所示。

在 ROS 系统中,为实现全局路径规划的算法移植,改进后的 JPS 算法被以插件形式与 move-base 功能包进行整合,并重新构建全局路径规划器。实验中通过 Rviz 可视化平台展示全局路径规划路线,结果如图 15 和 16 所示。



(a)试验场地1与slam建图

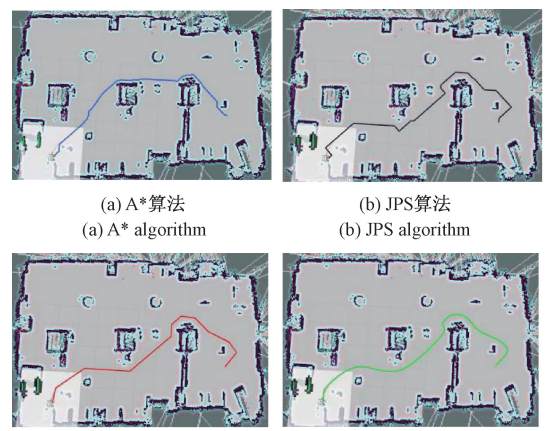
(a) Test site 1 with slam build map

(b)试验场地2与slam建图

(b) Test site 2 with slam build map

图 14　实验场地

Fig. 14　Experimental site



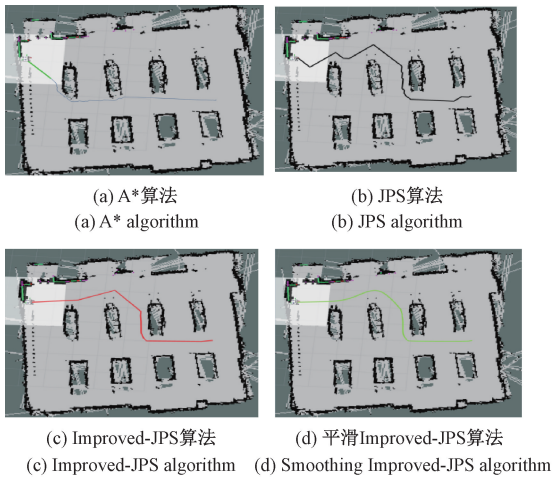
(a) A*算法　(b) JPS算法

(c) Improved-JPS算法　(d) 平滑Improved-JPS算法

图 15　试验场景 1 下不同算法效果图(map_global)

Fig. 15　Effect of different algorithms under test scenario 1

在图 15 和 16 中,对比了 A* 算法、JPS 算法和 Improved-JPS 算法的路径规划效果。A* 算法(图 15(a)与 16(a))尽管在寻找最短路径方面表现出色,但生成的路径包含了较多急转弯和不连续节点,整体路径较为生硬,缺乏平滑性。原始 JPS 算法的路径(图 15(b)与 16(b))虽然提升了计算效率,但依然存在大量急转弯和拐角,不利于需要平稳和连续路径的应用场景,难以满足高效行驶的需求。相比之下,改进后的 JPS 算法(图 15(c)与 16(c))在路径规划上进行了显著优化,路



(a) A*算法 (b) JPS算法
(a) A* algorithm (b) JPS algorithm

(c) Improved-JPS算法 (d) 平滑Improved-JPS算法
(c) Improved-JPS algorithm (d) Smoothing Improved-JPS algorithm

图 16 试验场景 2 下不同算法效果图

Fig. 16 Effect of different algorithms under test scenario 2

径相较于原始 JPS 更加平滑,使得路径在保持较优长度的情况下更加连续,提升了行驶的顺畅性和效率。此外,该算法有效减少了频繁的方向调整,简化了控制系统的设计和操作,特别适用于复杂场景。改进算法在路径规划中保持合理的安全距离,降低了潜在的碰撞风险,显著提升了与障碍物交互的安全性,确保了行驶的稳定性和可靠性。经过平滑处理后的改进 JPS 算法(图 15(d)与 16(d))进一步提升了路径的连续性和平滑性,能够为实际应用中对平稳行驶要求更高的场景提供更优的路径选择。

如表 4 和 5 所示,为 A* 算法、改进的 JPS 算法以及引入平滑策略的 Improved-JPS 算法在相同环境及相同条件下的实验数据对比。1)在试验场地 1 中,改进后的 Improved-JPS 算法平均搜索时间相较于 A* 算法提高了约 20.5%。相较于 JPS 提高了约 28.0%。这表明 Improved-JPS 算法在搜索效率上的优化对路径规划性能具有显著优势。Improved-JPS 算法的路径长度为 13.58 m,较 A* 算法缩短了 10.6%,较 JPS 算法缩短了 5.3%。表明在更复杂的环境中 Improved-JPS 算法依然能够保持更优的路径长度,使得导航路径更加流畅,Improved-JPS 算法的路径点数显著降低,这进一步表明改进后的算法在复杂环境中能更好地平滑路径,提高行驶稳定性。2)在试验场地 2 中,改进后的 JPS 算法展现出显著优势,Improved-JPS 算法的搜索时间比 JPS 算法减少了约 54%,比 A* 算法减少了约 35%。表明 Improved-JPS 算法拥有更加高效路径规划性能。路径长度与 A* 算法相比几乎一致,但比 JPS 减少了约 12%。这表明 Improved-JPS 算法在不增加路径长度的情况下实现了平滑化和优化路径,实现更高质量路径生成,更适合车辆连续平稳行驶的应用。

表 4 场地 1 不同算法的性能对比

Table 4 Performance comparison of different algorithms for site 1

算法	搜索时间/s	路径长度/m	路径点数
A* 算法	4.227 3	15.20	452
JPS 算法	4.664 2	14.35	225
Improved-JPS 算法	3.358 6	13.58	12

表 5 场地 2 不同算法的性能对比

Table 5 Performance comparison of different algorithms for site 2

算法	搜索时间/s	路径长度/m	路径点数
A* 算法	1.323 5	9.53	54
JPS 算法	1.865 7	10.89	46
Improved-JPS 算法	0.855 9	9.58	12

4 结 论

改进后的 Improved-JPS 算法在不同环境下展现了路径规划的高效性和可靠性。首先,扩展优先级调整实现了搜索方向的优化,使算法能够更高效地朝着有效路径前进。其次,路径优化策略进一步提升了路径的质量,包括安全节点更新策略、冗余节点消除策略和路径平滑策略。安全节点更新策略有效地规避了危险路径,冗余节点消除策略减少了不必要的控制点和路径总长度,而路径平滑策略则利用三次准均匀 B 样条曲线对路径进行平滑处理,生成了更流畅的行驶轨迹。

实验结果证明了路径优化策略的必要性,特别是在复杂环境下实现高效、安全、平滑路径的重要性。安全节点更新和冗余节点消除的组合,使得路径规划过程更为稳定和安全,避免了不必要的急转弯和路径重复。在复杂地图的实际实验中,Improved-JPS 算法平均搜索时间相较于 A* 算法提高了约 20.5%。相较于 JPS 提高了约 28.0%。这表明 Improved-JPS 算法在搜索效率上的优化对路径规划性能具有显著优势。Improved-JPS 算法的路径长度为 13.58 m,较 A* 算法缩短了 10.6%,较 JPS 算法缩短了 5.3%。表明 Improved-JPS 算法在搜索效率上具有显著优势。也展现了路径规划的安全、可靠性和有效性。虽然 Improved-JPS 算法展现了出色的路径规划能力,但在实际复杂环境下仍可能存在局限性。本文的实际环境比较简单,未来会在更加复杂的环境验证改进的算法的性能。

参考文献

[1] TANG B, YANG ZH Y, JIANG H B, et al. Stability-

considered lane keeping control of commercial vehicles based on improved apf algorithm[J]. Chinese Journal of Mechanical Engineering, 2024, 37(1): 19.

[2] CAO D P, WANG X, LI L X, et al. Future directions of intelligent vehicles: Potentials, possibilities, and perspectives[J]. IEEE Transactions on Intelligent Vehicles, 2022, 7(1): 7-10.

[3] YUAN CH R, LIU G F, ZHANG W Q, et al. An efficient RRT cache method in dynamic environments for path planning[J]. Robotics and Autonomous Systems, 2020, 131: 103595.

[4] 宋俊辉, 刘宇庭, 郭世杰. 动态环境下 AIP-RRT* 与 DGF-APF 融合的机器人路径规划[J]. 仪器仪表学报, 2025, 46(3): 51-64.
SONG J H, LIU Y T, GUO SH J. Robot path planning by fusion of AIP-RRT* and DGF-APF in dynamic environments[J]. Chinese Journal of Scientific Instrument 2025, 46(3): 51-64.

[5] KATIYAR S, DUTTA A. Dynamic path planning over CG-Space of 10DOF rover with static and randomly moving obstacles using RRT* rewiring[J]. Robotica, 2022, 40(8): 2610-2629.

[6] CAO K, CHENG Q, GAO S, et al. Improved PRM for path planning in narrow passages[C]. 2019 IEEE International Conference on Mechatronics and Automation, 2019: 45-50.

[7] SHANG E, DAI B, NIE Y M, et al. A guide-line and key-point based A-star path planning algorithm for autonomous land vehicles[C]. 2020 IEEE 23rd International Conference on Intelligent Transportation Systems, 2020: 1-7.

[8] GONZÁLEZ D, PÉREZ J, MILANÉS V, et al. A review of motion planning techniques for automated vehicles[J]. IEEE Transactions on Intelligent Transportation Systems, 2016, 17(4): 1135-1145.

[9] MA X L, GONG R, TAN Y B, et al. Path planning of mobile robot based on improved PRM based on cubic spline[J]. Wireless Communications and Mobile Computing, 2022, 2022(1): 1632698.

[10] MAGELE C, KÖSTINGER A, JAINDL M, et al. Niching evolution strategies for simultaneously finding global and pareto optimal solutions[J]. IEEE Transactions on Magnetics, 2010, 46(8): 2743-2746.

[11] 孙岩霆, 王荣杰, 蒋德松. 融合 A* 与 DWA 算法的水面船艇动态路径规划[J]. 仪器仪表学报, 2024, 45(1): 301-310.
SUN Y T, WANG R J, JIANG D S. Dynamic path planning of surface ship by combining A* and dynamic window algorithm[J]. Chinese Journal of Scientific Instrument, 2024, 45(1): 301-310.

[12] HARABOR D, GRASTIEN A. Online graph pruning for pathfinding on grid maps[C]. Proceedings of the AAAI Conference on Artificial Intelligence, 2011, 25(1): 1114-1119.

[13] ZHENG X, TU X W, YANG Q H. Improved JPS algorithm using new jump point for path planning of mobile robot[C]. 2019 IEEE International Conference on Mechatronics and Automation, 2019: 2463-2468.

[14] 陈奕虎, 苏珉, 闭金杰, 等. 基于改进 JPS 的移动机器人路径规划算法[J]. 电子测量技术, 2023, 46(7): 88-93.
CHEN Y H, SU M, BI J J, et al. Path planning algorithm of mobile robot based on improved JPS algorithm[J]. Electronic Measurement Technology, 2023, 46(7): 88-93.

[15] 孙凌宇, 王威, 秦红亮, 等. 跳点优化蚁群算法的移动机器人路径规划[J]. 电子测量技术, 2023, 46(9): 48-53.
SUN L Y, WANG W, QIN H L, et al. Mobile robot path planning based on jump point optimization ant colony algorithm[J]. Electronic Measurement Technology, 2023, 46(9): 48-53.

[16] ZHOU K J, YU L L, LONG Z W, et al. Local path planning of driverless car navigation based on jump point search method under urban environment[J]. Future Internet, 2017, 9(3): 51.

[17] 周熙栋, 张辉, 陈波. 非结构化场景下基于改进 JPS 算法的移动机器人路径规划[J]. 控制与决策, 2024, 39(2): 474-482.
ZHOU X D, ZHANG H, CHEN B. Path planning for mobile robots based on improved JPS algorithm in unstructured environments[J]. Control and Decision, 2024, 39(2): 474-482.

[18] CHEN T, CHEN S F, ZHANG K R, et al. A jump point search improved ant colony hybrid optimization algorithm for path planning of mobile robot[J]. International Journal of Advanced Robotic Systems, 2022, 19(5): 17298806221127953.

[19] HOU Y X, GAO H B, WANG Z J, et al. Mobile robot path planning based on improved jump point search algorithm[J]. Modular Machine Tool & Automatic Manufacturing Technique, 2023, 589(3): 54-58.

[20] SU Q H, MA SH B, WANG L Y, et al. Artificial potential field guided JPS algorithm for fast optimal path planning in cluttered environments[J]. Journal of the Brazilian Society of Mechanical Sciences and

Engineering, 2022, 44(12): 602.

[21] 刘荣华, 王欣, 吴迪, 等. 改进双向动态 JPS 算法的移动机器人全局路径规划[J]. 计算机应用研究, 2024, 41(4): 1117-1122.
LIU R H, WANG X, WU D, et al. Improved bidirectional dynamic JPS algorithm for mobile robot global path planning[J]. Application Research of Computers, 2024, 41(4): 1117-1122.

[22] HUANG ZH B, HU L K, ZHANG Y, et al. Research on security path based on improved hop search strategy[J]. Computer Engineering and Applications, 2021, 57(1): 56-61.

[23] 田雅琴, 胡梦辉, 刘文涛, 等. 基于跳点搜索-遗传算法的自主移动机器人路径规划[J]. 工程设计学报, 2023, 30(6): 697-706.
TIAN Y Q, HU M H, LIU W T, et al. Path planning for autonomous mobile robots based on jump point search and genetic algorithm[J]. Chinese Journal of Engineering Design, 2023, 30(6): 697-706.

[24] 刘晓茜, 刘海鹏, 毛剑琳, 等. 基于双向 JPS 算法的室外移动机器人 2.5D 路径规划[J]. 控制工程, 2025, 32(2): 233-241.
LIU X Q, LIU H P, MAO J L, et al. 2.5D path planning for outdoor mobile robots based on bidirectional JPS algorithm[J]. Control Engineering, 2025, 32(2): 233-241.

[25] LIU L S, YAO J X, HE D W, et al. Global dynamic path planning fusion algorithm combining jump-A* algorithm and dynamic window approach[J]. IEEE Access, 2021, 9: 19632-19638.

[26] YANG ZH, LI J L, YANG L W, et al. A smooth jump point search algorithm for mobile robots path planning based on a two - dimensional grid model[J]. Journal of Robotics, 2022, 2022(1): 7682201.

[27] FAN B B, GUO L L. An improved JPS algorithm for global path planning of the seabed mining vehicle[J]. Arabian Journal for Science and Engineering, 2024, 49(3): 3963-3977.

[28] MI Z ZH, XIAO H J, HUANG CH H. Path planning of indoor mobile robot based on improved A* algorithm incorporating RRT and JPS[J]. AIP Advances, 2023, 13(4): 045313.

[29] 陈芹, 李燕, 樊新宇. 改进跳点搜索算法的移动机器人路径规划[J]. 组合机床与自动化加工技术, 2024(9): 81-85.
CHEN Q, LI Y, FAN X Y. Path planning of mobile robot with improved JPS[J]. Modular Machine Tool & Automatic Manufacturing Technique, 2024 (9): 81-85.

作者简介

朱振东,2023 年于辽宁石油化工大学获得学士学位,现为青海大学硕士研究生,主要研究方向为智能车路径规划与跟踪控制。
E-mail:18212584744@163.com

Zhu Zhendong received his B. Sc. degree from Liaoning Petrochemical University in 2023. He is currently pursuing a M. Sc. degree at Qinghai University. His research interests include path planning and tracking control of intelligent vehicles.

姚强强(通信作者),2022 年于北京交通大学获得博士学位,现为青海大学副教授,主要研究方向为智能车辆路径规划与路径跟踪控制以及新能源汽车能量优化控制等方向的研究。
E-mail:yaoqiang@bjtu.edu.cn

Yao Qiangqiang (Corresponding author) received his Ph. D. degree from Beijing Jiaotong University in 2022. He is currently an associate professor at Qinghai University. His research interests include path planning and path tracking control for intelligent vehicles, as well as energy optimization control for new energy vehicles.