Artificial Intelligence and Robotics Research 人工智能与机器人研究, 2025, 14(4), 885-892
Published Online July 2025 in Hans. https://www.hanspub.org/journal/airr
https://doi.org/10.12677/airr.2025.144084

Hans汉斯

基于改进JPS算法的路径规划方法研究

孙伟博*，贾丹平#

沈阳工业大学信息科学与工程学院，辽宁 沈阳

收稿日期：2025年5月26日；录用日期：2025年7月9日；发布日期：2025年7月24日

摘　要

针对跳点搜索(Jump Point Search, JPS)算法搜索方向不明确、搜索范围过大和存在冗余节点等问题，提出一种改进JPS算法。首先，对启发式函数添加权重系数提升搜索效率；其次，设置椭圆形动态扩展边界限制搜索范围；最后，通过提取关键跳点优化拐点数量和路径长度。实验结果表明，改进算法与传统算法相比，拐点和计算节点数更少，路径长度更短，搜索效率更高。

关键词

移动机器人，路径规划，跳点搜索算法，启发式函数，动态扩展边界

Research on Path Planning Method Based on Improved JPS Algorithm

Weibo Sun*, Danping Jia#

School of Information Science and Engineering, Shenyang University of Technology, Shenyang Liaoning

Received: May 26th, 2025; accepted: Jul. 9th, 2025; published: Jul. 24th, 2025

Abstract

To address the issues of unclear search direction, excessive search range, and redundant nodes in the Jump Point Search (JPS) algorithm, an improved JPS algorithm is proposed. Firstly, a weight coefficient is added to the heuristic function to enhance search efficiency. Secondly, an elliptical dynamic expansion boundary is established to limit the search range. Finally, key jump points are extracted to optimize the number of inflection points and path length. Experimental results demonstrate

*第一作者。
#通讯作者。

文章引用: 孙伟博, 贾丹平. 基于改进 JPS 算法的路径规划方法研究[J]. 人工智能与机器人研究, 2025, 14(4): 885-892.
DOI: 10.12677/airr.2025.144084

that compared with traditional algorithms, the improved algorithm achieves fewer inflection points and computational nodes, shorter path length, and higher search efficiency.

Keywords

Mobile Robot, Path Planning, Jump Point Search Algorithm, Heuristic Function, Dynamic Expansion Boundary

1. 引言

路径规划是机器人导航领域的核心问题之一，其目的是通过规划算法在给定环境中高效地找到从起点到终点的安全路径[1]。随着智能系统的快速发展，对路径规划的要求日益增加。传统的路径规划算法虽然在理论上能够保证最优解，但在处理大规模环境时往往面临计算效率低下的问题[2]。因此，研究高效的路径规划算法具有重要的理论意义和应用价值。

JPS 算法是一种基于栅格地图的路径规划方法，相较于 A*算法，JPS 算法能够跳过大量无关的中间节点，直接定位路径中的关键点，使搜索复杂度大大降低[3]。这一特性使其在大规模地图和动态环境中表现出优越的性能，成为近年来路径规划领域的研究热点之一。鉴于 JPS 算法的独特优势，国内外很多学者对其进行了不同的改进和应用。程擎等[4]运用几何航迹碰撞检测法改进 JPS 算法，提升了无人机在全局环境中的航迹规划能力；赵晓东等[5]提出一种基于蜂窝栅格地图的 JPS 算法，解决了规划过程中穿越墙角的不安全行为；陈芹等[6]使用双向 JPS 算法并与 B 样条曲线结合，提升了路径的平滑性。周熙栋等[7]将 JPS 算法与 A*算法结合，提出一种基于分层栅格地图的融合算法，解决了非结构化场景的全局规划问题。

上述改进算法在不同程度上提高了 JPS 算法的性能，但是并没有很好地解决其需要计算冗余节点的问题，其搜索效率仍有大量提升空间。针对 JPS 算法的不足，本文提出三种优化策略，分别对搜索方向、搜索范围和路径长度进行优化，使算法能够高效地规划路径。

2. JPS 算法

JPS 算法的搜索流程和 A*算法基本相同，需要维护一个待扩展节点列表 Openlist 和一个已扩展节点列表 Closelist [8]，但不同的是 JPS 算法仅维护跳点节点，最终得到的路径为一个跳点坐标的集合，具体流程如下：

1) 初始化并将起始节点加入 Openlist。

2) 从 Openlist 中选择代价最低的节点 x 作为当前节点，并将该节点从 Openlist 中移除。节点 x 的代价为 $f(x)=g(x)+h(x)$，其中 $g(x)$为初始节点到节点 x 的实际路径长度，$h(x)$为节点 x 到目标节点的预估路径长度。

3) 若当前节点 x 是目标节点，则搜索成功，回溯父节点得到路径。否则，检查节点 x 是否有强迫邻居，若有，则将强迫邻居加入 Openlist，设置它们的父节点为 x，并将 x 加入 Closelist。

4) 以先水平，再垂直，后斜向的顺序进行方向跳跃性搜索。

若搜索到目标节点，则将目标节点的父节点设为 x，并将 x 加入 Closelist，搜索结束，返回路径。若

搜索到跳点，则将跳点加入 Openlist，将跳点的父节点设为 x，并将 x 加入 Closelist，停止搜索。若搜索到障碍物，则停止当前方向的搜索。

5) 重复步骤 2)到步骤 4)，直到 Openlist 为空。

一个完整的 JPS 算法搜索过程如图 1 所示，其中黑色栅格为障碍物，蓝色栅格为跳点，红色路径即为算法得到的最优路径。

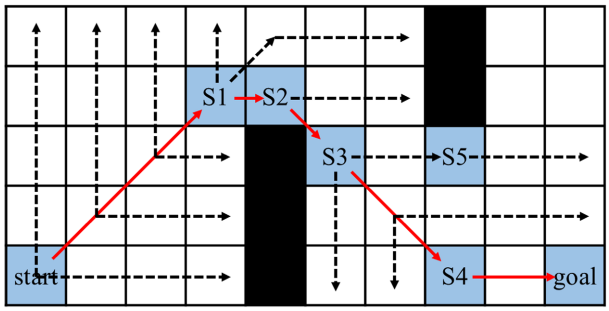


Figure 1. Schematic diagram of the JPS algorithm
图 1. JPS 算法示意图

3. 改进 JPS 算法

3.1. 优化启发式函数

传统 JPS 算法的启发式函数为 $f(x)=g(x)+h(x)$，$g(x)$为初始节点到节点 x 的实际路径长度，$h(x)$为节点 x 到目标节点的预估路径长度。通过对 $h(x)$进行优化，可以使算法在搜索过程中的不同时期表现出不同的倾向性。

常见的 $h(x)$估计函数有 4 种，如图 2 所示，这 4 种表示方法分别适用于不同场景。图中蓝色线条为欧几里得距离，适用于连续空间或自由移动场景；橙色线条为 Octile 距离，适用于八方向移动场景，即只能水平、垂直和 45˚斜向移动的栅格地图；绿色线条为曼哈顿距离，适用于只能水平或垂直移动的栅格地图；紫色线条为切比雪夫距离，适用于八方向移动的栅格地图，但往往会低估代价。

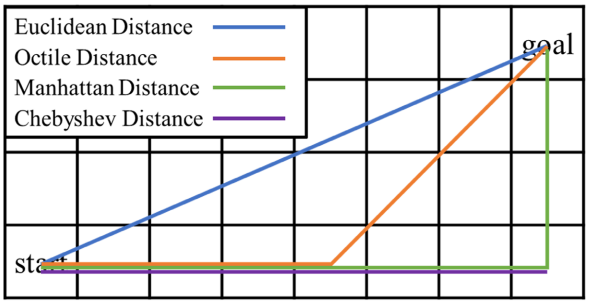


Figure 2. Common heuristic functions
图 2. 常见的启发式函数

由于 JPS 算法的搜索方向为水平、垂直和斜向的共八个方向，故选取 Octile 距离最能贴合实际需求。针对 $h(x)$，在前方没有障碍物的情况下，Octile 距离即为实际的代价距离，在有障碍物的情况下，利用 Octile 距离也能很好地规避无效节点并估计代价距离。Octile 距离的数学表达为：

$$h(x)=\max\left(|x_2-x_1|,|y_2-y_1|\right)+\left(\sqrt{2}-1\right)\min\left(|x_2-x_1|,|y_2-y_1|\right) \tag{1}$$

为了使算法在搜索过程中，尤其是距离目标节点较远的初期时，能够迅速朝着目标节点的方向进行探索，可以对 $h(x)$进行优化，让 $h(x)$距离目标节点越远，其在 $f(x)$中的占比越大，使路径充分考虑搜索方向，从而提升搜索效率。为优化启发式函数，提出启发式权重影响因子 α，并将原启发式函数优化为：

$$f(x)=g(x)+\alpha h(x) \tag{2}$$

$$\alpha=1+\frac{h(x)}{D} \tag{3}$$

其中，D 为起始节点到目标节点的 Octile 距离，取 O 等于 $1/D$，即起始节点到目标节点 Octile 距离的倒数，最终的启发式函数可以表示为式(4)。针对一次路径规划过程，该函数的 O 为定量，故新函数没有引入过多的计算量，并且充分考虑了搜索方向对搜索效率的影响。

$$f(x)=g(x)+h(x)+O\cdot h^2(x) \tag{4}$$

3.2. 限制搜索范围

为了进一步降低 JPS 算法的计算量，减少无效跳点，引入动态扩展边界来限制算法的搜索区域，在路径规划过程中通过双焦点定位动态椭圆作为扩展边界，从而大大减小搜索范围。

(1) 运用动态扩展边界进行搜索的流程：

1) 将起始节点和目标节点作为椭圆的两个焦点，确定初始状态下的动态扩展边界。

2) 在限定的搜索范围内(包含边界)，结合优化后的启发式函数进行路径搜索。

3) 在搜索过程中，若当前限制范围内无可行路径，则以当前节点和目标节点为焦点，重新确定动态扩展边界。

4) 经过步骤 3)后，若在新的限制范围内仍无可行路径，可能此时已经足够接近目标节点并且目标节点附近存在大量障碍物，则放弃扩展边界的限制，直接在原始地图下进行搜索，直到搜索到下一跳点后以此节点和目标节点为焦点，重新确定动态扩展边界。

5) 重复步骤 2)到步骤 4)，直到搜索到最优路径，返回最优路径。

(2) 动态扩展边界的确定：

将当前节点和目标节点设为焦点，绘制椭圆形动态扩展边界。椭圆的焦距 $2c$ 即为两点间的距离，为充分利用已有的计算结果、减轻计算压力，两点间距离的计算依旧采用前文中的 Octile 距离，即当前节点的 $h(x)$值。椭圆形动态扩展边界的数学表达式为：

$$\frac{(x-m)^2}{a^2}+\frac{(y-n)^2}{b^2}=1,(a>b>0) \tag{5}$$

其中，m 和 n 分别表示椭圆中心的横坐标和纵坐标，可由两个焦点的坐标计算得出。$2a$ 为椭圆的长轴长，$2b$ 为椭圆的短轴长。提出动态扩展边界膨胀系数 β，定义 $a=\beta h$ 。

根据椭圆的几何性质可知焦距与长短轴之间的关系为：

$$c^2=a^2-b^2 \tag{6}$$

其中，c 为焦距的一半，即两点间 Octile 距离的一半，可由式(1)得出，即 $c=h/2$。那么：

$$\left(\frac{h}{2}\right)^2=(\beta h)^2-b^2 \tag{7}$$

进一步整理，则 b 为：

$$b=h\sqrt{\beta^2-\frac{1}{4}}=\frac{h\sqrt{4\beta^2-1}}{2} \tag{8}$$

由此可见，扩展边界的椭圆可以完全由当前节点坐标、目标节点坐标、当前节点到目标节点的预估代价值 h、和动态扩展边界膨胀系数 β 得到。

扩展边界膨胀系数 β 应该与局部地图的复杂度正相关，以当前节点和目标节点作为对角点的矩形为边界，以矩形框内占用栅格占总栅格的比例来衡量地图的复杂程度。为尽可能避免无可行解情况的发生，局部地图复杂度越高，动态扩展椭圆应设置的越大。这种动态设置扩展边界的方式，既可以防止因搜索范围过大而增加计算压力，又可以避免因搜索范围过小而出现无解的情况。图 3 为局部地图复杂度的确定，其中黑色栅格为障碍物，红色虚线矩形框内为确定复杂度的局部地图。左图中局部地图共有栅格 36 个，其中障碍物占 5 个，复杂度为 0.139；右图中局部地图共有栅格 15 个，其中障碍物占 3 个，复杂度为 0.200。为简化计算，当复杂度小于 0.1 时，β 取 0.6；当复杂度大于等于 0.1 小于 0.2 时，β 取 0.8，当复杂度大于等于 0.2 时，β 取 1.0。

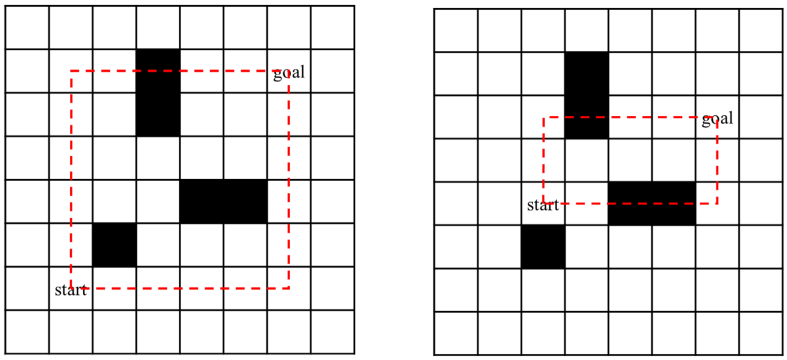
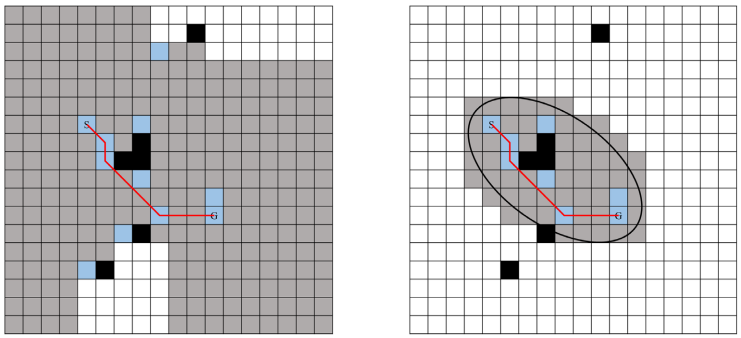


Figure 3. Determination of local map complexity
图 3. 局部地图复杂度的确定

动态扩展边界用于限制当前迭代下的地图搜索范围。图 4 展示了利用一个迭代周期内进行搜索时，有、无扩展边界对搜索范围以及跳点数的影响。其中，蓝色栅格代表跳点，灰色栅格代表搜索范围。右图运用椭圆形动态扩展边界，剔除了非核心节点，在边界内(包含边界)进行搜索。与左图所示的传统搜索方式相比，动态扩展边界的运用使得算法的搜索方向更明确、搜索节点(灰色栅格)和计算节点(蓝色栅格，跳点)更少，大大降低了计算量。图 4 中当前的局部地图复杂度为 0.063，β 取 0.6。椭圆形动态扩展边界的焦距 $2c$ 为 9.070，长轴长 $2a$ 为 10.884，短轴长 $2b$ 为 6.016。运用动态扩展边界进行搜索后，搜索节点栅格数从 271 减少到 62，减少了 77.1%；计算节点栅格数从 11 减少到 8，减少了 27.3%。



(a) (b)

Figure 4. Comparison of search processes with and without dynamic boundary expansion. (a) Without dynamic boundary expansion; (b) With dynamic boundary expansion
图 4. 有无动态扩展边界搜索过程对比。(a) 不使用动态扩展边界；(b) 使用动态扩展边界

3.3. 提取关键跳点

在优化启发式函数并限制扩展边界后，所规划出的路径依旧存在个别跳点是冗余跳点，这些跳点并不会影响搜索速度，但是会使路径变长、拐点变多。针对该问题，通过提取关键跳点来进一步优化路径，从而缩短路径的实际长度，减少拐点的数量。具体步骤如下：

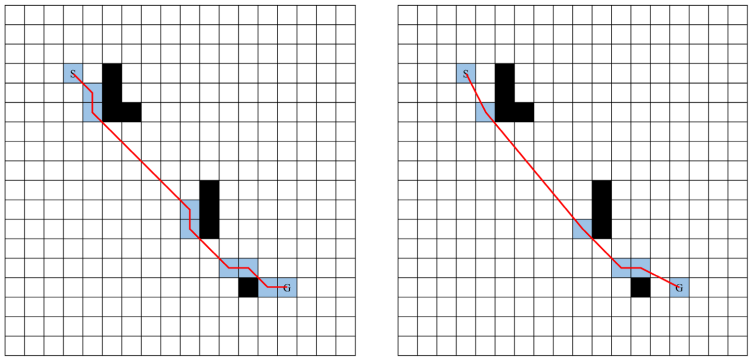
1) 取出原路径列表中的第一个跳点 p 和它的后方跳点 p + 1 和 p + 2。

2) 连接 p 和 p + 2，判断该连线是否经过障碍物。若未经过障碍物，则 p + 1 为冗余跳点，删掉该跳点，令原路径中的 p + 2 和 p + 3 作为新一轮的 p + 1 和 p + 2，重复操作直到 p 与 p + 2 的连线经过障碍物为止。

3) 令当前的 p + 1 为新的 p。

4) 判断 p + 2 是否为终点。若否，返回步骤 2)，若是，跳出流程并返回全局路径。

是否提取关键跳点的最终路径对比如图 5 所示。



(a) (b)

Figure 5. Comparison of final paths with and without key jump point extraction. (a) Without key jump point extraction; (b) With key jump point extraction
图 5. 是否提取关键跳点最终路径对比。(a) 不提取关键跳点；(b) 提取关键跳点

4. 仿真和实际场景验证

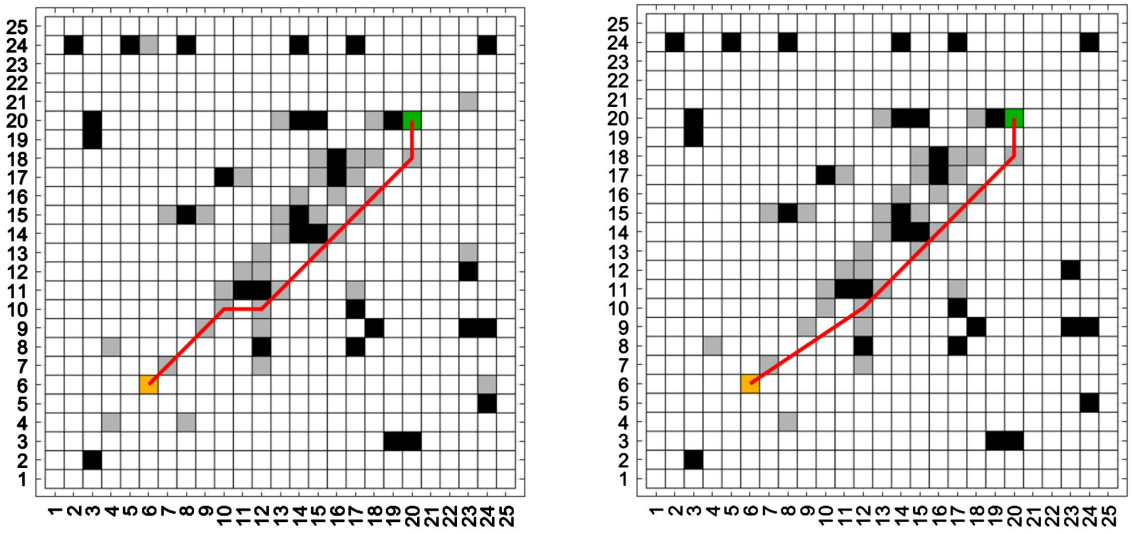
4.1. 仿真实验

为验证改进 JPS 算法的性能，在栅格地图上进行对比仿真实验，分别运用 JPS 算法和改进 JPS 算法进行路径规划。实验基于 matlab R2021a 平台，结果如图 6 所示。其中，橙色栅格为起始节点，绿色栅格为目标节点，黑色栅格为障碍物，灰色栅格为计算节点，红色线条为最优路径。

由表 1 可知，改进 JPS 算法相比传统 JPS 算法在各项指标上均有提升。仿真实验结果表明，启发式函数的改进和动态扩展边界的引入使 JPS 算法的遍历节点数和计算节点数大幅下降，计算量减少；提取关键跳点策略的运用使传统 JPS 算法拐点数减少、路径长度缩短。其中，路径长度缩短了 2.15%，规划时间缩短了 12.96%，规划效率显著提升。

Table 1. Comparison of simulation experiment results
表 1. 仿真实验结果数据对比

算法	拐点数	遍历节点数	计算节点数	路径长度	规划时间/s
JPS	3	320	41	20.97	0.0162
改进 JPS	2	206	36	20.52	0.0141

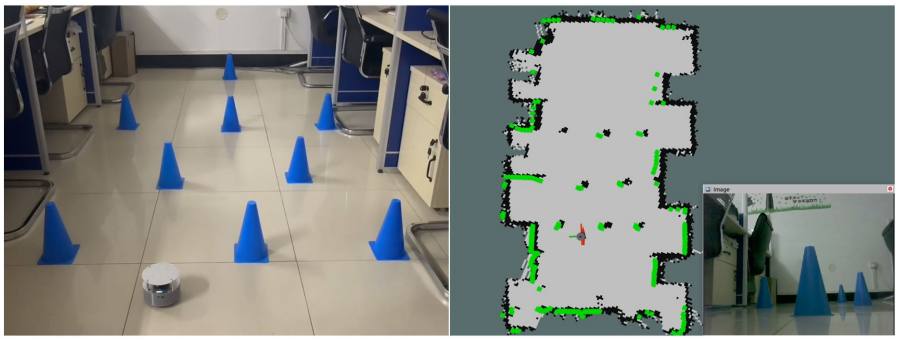


(a) (b)

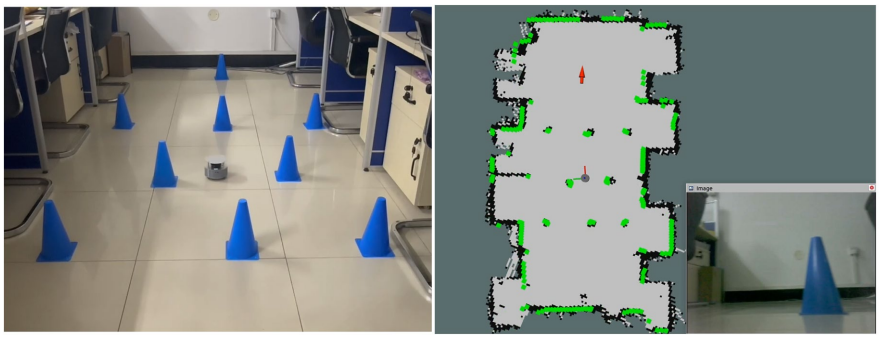
Figure 6. Comparison of simulation experiment results. (a) Planning result of baseline JPS; (b) Planning result of enhanced JPS
图 6. 仿真实验结果对比。(a) JPS 算法规划结果；(b) 改进 JPS 算法规划结果

4.2. 实际场景验证

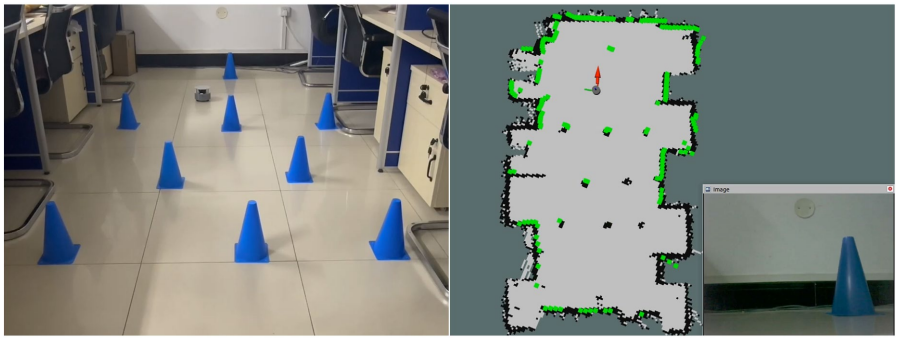
为进一步验证改进 JPS 算法在实际应用中的可行性，在 ROS 机器人上进行了实验。实验基于 Ubuntu20.04 系统、ROS Noetic 平台，将 ROS 导航系统 GlobalPlanner 中的算法替换为改进 JPS 算法，在建立好的地图上运行，运行结果如图 7 所示。



(a)



(b)



(c)

Figure 7. Field test validation results. (a) Initial state; (b) During operation; (c) Final state
图 7. 实际场景验证结果。(a) 初始状态；(b) 运行过程中；(c)运行结束

经过实际测试，改进 JPS 算法在复杂环境中表现出良好的实时性和稳定性，能够满足实际应用的需求。

5. 总结

传统 JPS 算法存在搜索方向不明确、搜索范围过大和存在冗余节点等问题，为解决这些问题，本文提出三种改进措施分别对算法进行优化。对启发式函数添加权重系数，使算法在规划初期能迅速朝目标节点的方向进行搜索；引入动态扩展边界限制搜索范围，减少遍历节点数量以提升效率；通过提取关键跳点优化拐点数量和路径长度。实验结果表明，改进算法与传统算法相比，规划效率更高，可以满足实际应用需求，并且优化效果明显。

参考文献

[1] 崔炜, 朱发证. 机器人导航的路径规划算法研究综述[J]. 计算机工程与应用, 2023, 59(19): 10-20.

[2] 杨姝慧, 杨姝慧, 李彬. 机器人路径规划算法研究分析与综述[J]. 齐鲁工业大学学报, 2024, 38(5): 37-46.

[3] Wang, F., Sun, W., Yan, P., Wei, H. and Lu, H. (2024) Research on Path Planning for Robots with Improved A* Algorithm under Bidirectional JPS Strategy. *Applied Sciences*, **14**, Article 5622. https://doi.org/10.3390/app14135622

[4] 程擎, 王圣淳, 李云飞, 等. 耦合改进 JPS 与 DWA 的无人机航迹规划[J]. 电光与控制, 2023, 30(9): 52-55, 67.

[5] 赵晓东, 侯坤, 王建超, 等. 面向蜂窝栅格地图的改进跳点搜索算法研究[J]. 计算机工程与应用, 2025, 61(8): 100-107.

[6] 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.

[7] 魏博闻, 严华. 一种面向非结构化环境的改进跳点搜索路径规划算法[J]. 科学技术与工程, 2021, 21(6): 2363-2370.

[8] 杨聿壬, 郭江宇, 董晓峰, 等. 基于改进 A*算法的安全路径规划[J]. 电脑知识与技术, 2024, 20(9): 1-4, 11.