DOI: 10.13382/j.jemi.B2508321

连续障碍物环境中安全高效的 A* 路径规划算法

黄家博[1,2]　陈春梅[1,2]　龚渔民[1,2]　刘桂华[1,2]　徐　敏[1,2]
(1. 西南科技大学信息与控制工程学院　绵阳　621010;2. 特殊环境机器人技术四川省重点实验室　绵阳　621010)

摘　要:针对 A* 算法在大范围连续障碍物环境中存在时间效率低、遍历节点数过多、安全性低及路径不平滑的问题,提出一种改进的 A* 算法。首先,提出 8 种双层 5 领域来提高时间效率和路径平滑度,并设计边界拓展和满障碍物拓展的方法来解决使用小邻域搜索陷入死锁的问题;其次,提出一种启发函数分层策略,该策略将搜索区域分层并根据分层阈值赋予不同层启发函数不同权重,从而减少遍历节点数及进一步提高时间效率;最后,提出一种安全探测的方法来使路径与障碍物保持安全距离。与不同环境下的 5 种算法相比,仿真实验表明改进 A* 算法的运行时间平均减少了 30.9%,路径安全性平均提高了 14.7%。此外,改进 A* 算法的路径平滑度较高,综合性能优于其他 5 种算法。所提出的改进 A* 算法不仅能在大范围连续障碍物环境中满足移动机器人安全高效的路径规划需求,还在不同环境中表现出更强的鲁棒性。在不对路径进行二次规划的情况下改进 A* 算法的路径也能具有较高的平滑度。

关键词:路径规划;A* 算法;移动机器人;小邻域搜索;连续障碍物环境
中图分类号:TP242.6;TN96　　**文献标识码**:A　　**国家标准学科分类代码**:510.80

Safe and efficient A* path planning algorithm for continuous obstacle environments

Huang Jiabo[1,2]　Chen Chunmei[1,2]　Gong Yumin[1,2]　Liu Guihua[1,2]　Xu Min[1,2]
(1. School of Information and Control Engineering, Southwest University of Science and Technology, Mianyang 621010, China;
2. Robot Technology Used for Special Environment Key Laboratory of Sichuan Province, Mianyang 621010, China)

Abstract: To address the problems of low computational efficiency, excessive node exploration, poor safety, and non-smooth path in large-scale continuous obstacle environments, this paper proposes an improved A* algorithm. Firstly, eight types of two-layer 5-neighborhood are proposed to improve computational efficiency and enhance path smoothness, and boundary expansion and full-obstacle expansion methods are designed to resolve the deadlock caused by small neighborhood search. Secondly, a hierarchical heuristic function strategy is proposed. The strategy divides the search space into multiple layers and assigns different weights to the heuristic functions of each layer based on predefined thresholds, thereby reducing the number of explored nodes and further improving computational efficiency. Finally, a safety detection method is proposed to ensure that the generated paths maintain a safe distance from obstacles. Compared with five algorithms in different environments, simulation experiments demonstrate that the improved A* algorithm reduces running time by an average of 30.9%, increases path safety by an average of 14.7%. In addition, the improved A* algorithm generates paths with moderate smoothness and achieves better overall performance than the five compared algorithms. The proposed improved A* algorithm not only satisfies the requirements for safe and efficient path planning for mobile robots in large-scale continuous obstacle environments but also shows greater robustness in different environments. Even without secondary path optimization, the improved A* algorithm generates paths with relatively high smoothness.

Keywords: path planning; A* algorithm; mobile robots; small neighborhood search; continuous obstacle environments

收稿日期: 2025-04-17　　Received Date: 2025-04-17

0　引　言

随着人工智能和智能制造的发展,移动机器人已广泛应用到服务业、工业和农业等。移动机器人能够代替人类完成一些艰巨和危险的工作,这不仅提高了工作效率还减少了人类的伤亡。因此,移动机器人的发展对人类和社会具有重要的意义。移动机器人大致可分为 3 个模块,感知、路径规划和运动控制[1]。其中,路径规划是指在静态或动态环境中移动机器人规划出一条无碰撞的路径[2]。路径规划是感知和运动控制之间沟通的桥梁,为移动机器人安全高效到达目标点发挥着重要的作用。因此,提高路径规划的安全性和高效性对移动机器人的发展具有重要的意义。

路径规划算法可分为全局路径规划算法和局部路径规划算法。全局路径规划算法从已知环境中规划出一条全局路径。全局路径规划算法可分为 3 类,基于图搜索的方法[3-6]、基于采样的方法[7-9]和基于仿生的方法[10-12]。而局部路径规划算法从未知或部分已知的环境中规划出一条局部路径。局部路径规划算法主要有动态窗口法(dynamic window approach, DWA)[13]、时间弹性带算法(timed elastic band, TEB)[14]、人工势场法(artificial potential field, APF)[15]等。在这些路径规划算法中,A* 算法因其简单高效且能够高效地找到最短路径,而被广泛应用于移动机器人的静态地图路径规划。不过,传统的 A* 算法对环境的适应能力较差,从而导致时间效率低、遍历节点数过多。此外,传统的 A* 算法采用传统的 8 邻域搜索产生的路径不够平滑,并且与障碍物没有保持安全距离。

针对以上 A* 算法的不足,国内外的学者们提出了许多改进。文献[16]提出加权 A* 算法(Weighted A*),通过给启发函数赋予一个大于 1 的权重来改进启发函数,从而使 A* 算法提高搜索效率和减少遍历节点数。但 Weighted A* 不能够确保找到的路径最优。文献[17]提出一种多启发函数的 A* 算法(multi-heuristic A*, MHA*)来提高启发函数的启发性能。MHA* 虽然能提高搜索效率,但依然存在陷入次优解的问题。文献[18]提出混合 A* 算法(Hybrid A*),通过考虑运动学约束和障碍物提高了搜索效率和路径的平滑度。文献[19]增加节点到起点与目标点连线的代价到启发函数中来提高搜索效率和减少遍历节点数,但这种方法对于有大范围连续障碍物阻挡的环境搜索效率并不高。文献[20]通过优化邻域避免了斜穿障碍物,并通过二次规划使路径与障碍物保持距离。但这种二次规划的方法依然没有从根本上解决 A* 算法的缺陷,导致时间效率不高。在搜索邻域优化方面,文献[21]提出一种改进 8 邻域的方法使移动机器人与障碍物保持安全距离,同时还设计了垂距限值法来删除冗余节点,从而使路径能用 B 样条曲线平滑路径,但依然存在二次规划的问题,导致时间效率低。文献[22]排除与目标点相反 3 个邻域来提高搜索效率,减小了邻域,提高了搜索效率,但没有解决小邻域搜索陷入死锁的问题。文献[23]使用双层 9 邻域搜索的方法来平滑路径和提高搜索效率,但也存在陷入死锁的问题。双层邻域通常以增加邻域来优化路径,而单层邻域往往减小邻域以提高搜索效率。

在大范围连续障碍物阻挡的环境中,由于连续障碍物的阻挡,地图会存在大量对称且代价相同的节点,从而导致遍历节点过多、时间效率低。现有的改进 A* 算法主要通过优化启发函数和搜索邻域来提高算法的性能。在大范围连续障碍物阻挡的环境中,这两个方面的改进仅能提高安全性、平滑性,并没有解决对称且代价相同节点多的问题以提高时间效率。要解决在大范围连续障碍物阻挡的环境中大量对称且代价相同的节点的问题,A* 算法需要在搜索策略上进行改进。现有方法中,只有少数改进 A* 算法能较高效地生成路径。如跳点搜索策略(jump point search, JPS)[24]通过 8 个方向寻找跳点作为父节点的邻居节点,这极大减少了大量对称且代价相同的节点。但 JPS 无法生成平滑的路径以及与障碍物保持安全距离。文献[25]改进的 JPS 使用双向搜索有效提高了搜索效率,但在极端的环境下搜索效率不如 JPS 生成的路径,且依然存在平滑性和安全性的问题。文献[26]在 JPS 的启发函数上增加了一个自适应权重,虽然极大提高了搜索效率,但如加权 A* 一样会存在全局路径不是最优的问题。所提 JPS 算法及 JPS 的变体都聚焦于快速寻找最优路径,没有考虑路径的安全性和平滑性。除 JPS 及 JPS 的变体之外,文献[27]设计了一种基于分层解耦的 A* 算法以处理大范围障碍物阻挡环境,该方法提高了搜索效率,但是聚焦于将大范围障碍物分块并没有聚焦于连续障碍物。文献[28]使用安全时间间隔策略的 A* 算法解决了在大范围连续障碍物端点处动态障碍物的存在造成路径阻塞的问题,目的主要是为了降低由于动态碍物存在高路径成本,而不是优化该环境下的搜索效率。大语言模型的 A* 算法(large language model A*, LLM-A*)[29]在大范围连续障碍物阻挡环境中提高了搜索效率,但 LLM-A* 依赖目标节点的生成,具有不稳定性且推理成本过高,部署成本比 A* 算法高。

因此,针对现有改进存在小邻域搜索陷入死锁、不能同时兼顾时间效率与安全性以及在大范围连续障碍物阻挡环境中搜效率低的问题,本文提出一种改进的 A* 算法。

改进 A* 算法主要有 3 点改进:1)提出 8 种双层搜索邻域以提高路径平滑度和搜索效率,并设计一种边界拓

展的方法以解决小邻域搜索死锁;2)提出一种分层策略以解决在大范围连续障碍物阻挡环境中搜索效率低的问题;3)提出一种安全探测的方法以避免与障碍物发生碰撞。改进 A* 算法不仅为大范围连续障碍物环境下的路径规划提供了一种安全高效的方法,还为小邻域搜索的不足提供了新的解决思路。

1　传统 A* 算法

A* 算法是一种基于图搜索的算法,因其启发搜索的思想而被广泛应用于路径规划。A* 算法的路径规划主要包括地图建模、搜索邻域、代价函数 3 部分。

A* 算法在进行路径规划前需要将地图栅格化以构建地图模型。每一个栅格代表一个节点,方便了移动机器人的状态表示和路径规划。栅格地图如图 1 所示。

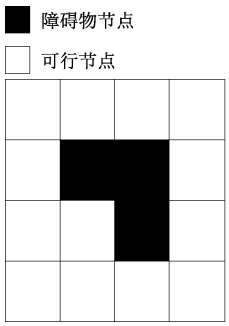


图 1　栅格地图

Fig. 1　Grid map

构建地图后,A* 算法需要确定搜索领域。传统 A* 算法的搜索邻域是一层,如 4 邻域和 8 邻域。搜索邻域越小,时间效率就越高。但小邻域搜索牺牲了路径平滑度,因此 A* 算法大多采用 8 邻域搜索路径。A* 算法常用搜索邻域如图 2 所示,图 2(a)为 4 邻域,图 2(b)为 8 邻域。

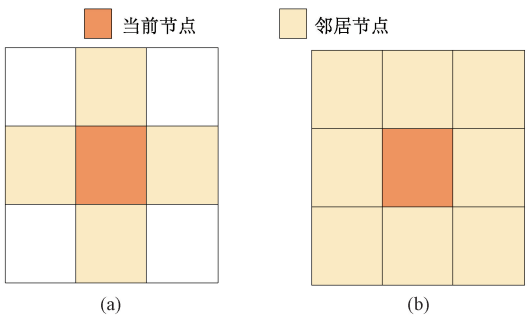


图 2　搜索邻域

Fig. 2　Search neighborhood

A* 算法的关键之处在于代价函数中的启发函数。通过启发函数,A* 算法能够有目的地寻找路径。A* 算法的代价函数 $f(n)$ 的定义为:

$$f(n) = g(n) + h(n) \tag{1}$$

式中:n 表示当前节点;$f(n)$ 表示节点 n 的总代价;$g(n)$ 表示起点到当前节点 n 的实际代价;$h(n)$ 是启发函数,用以预估当前节点 n 到目标节点的预估代价。

启发函数的设计决定了 A* 算法的搜索效率。目前常用的启发函数有曼哈顿距离(式(2))、欧氏距离(式(3))和切比雪夫距离(式(4))。这 3 个启发函数都具备可采纳性和一致性,能够确保找到最优解。

$$h(n) = |x_n - x_G| + |y_n - y_G| \tag{2}$$

$$h(n) = \sqrt{(x_n - x_G)^2 + (y_n - y_G)^2} \tag{3}$$

$$h(n) = \max(|x_n - x_G|, |y_n - y_G|) \tag{4}$$

式中:(x_n, y_n) 为当前节点坐标;(x_G, y_G) 为目标节点坐标。

确定地图、搜索邻域和代价函数后,A* 算法即可从起点向目标点搜索路径。A* 算法的大致流程如图 3 所示。

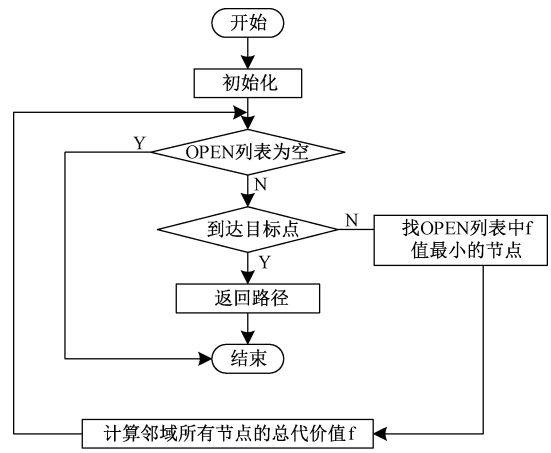


图 3　A* 算法流程

Fig. 3　Flowchart of A* algorithm

2　改进 A* 算法

改进 A* 算法有 3 部分改进,分别为双层小邻域搜索、启发函数分层、安全探测。改进 A* 算法的主体框架如图 4 所示。

2.1　双层小邻域搜索策略

针对单层 8 邻域时间效率较低和路径不平滑的问题,双层 5 邻域能有效解决该问题。提出的 8 种双层 5 邻域如图 5 所示。8 种双层 5 邻域可分为两类:直线邻域(Neighborhood 1-4)和非直线邻域(Neighborhood 5-8)。根据夹角 α,改进的 A* 算法可以确定图 5 中的一种邻域作为搜索区域内的搜索邻域。$\alpha = 0$,则搜索邻域为 Neighborhood 5;$\alpha \in (0,90)$,则搜索邻域为

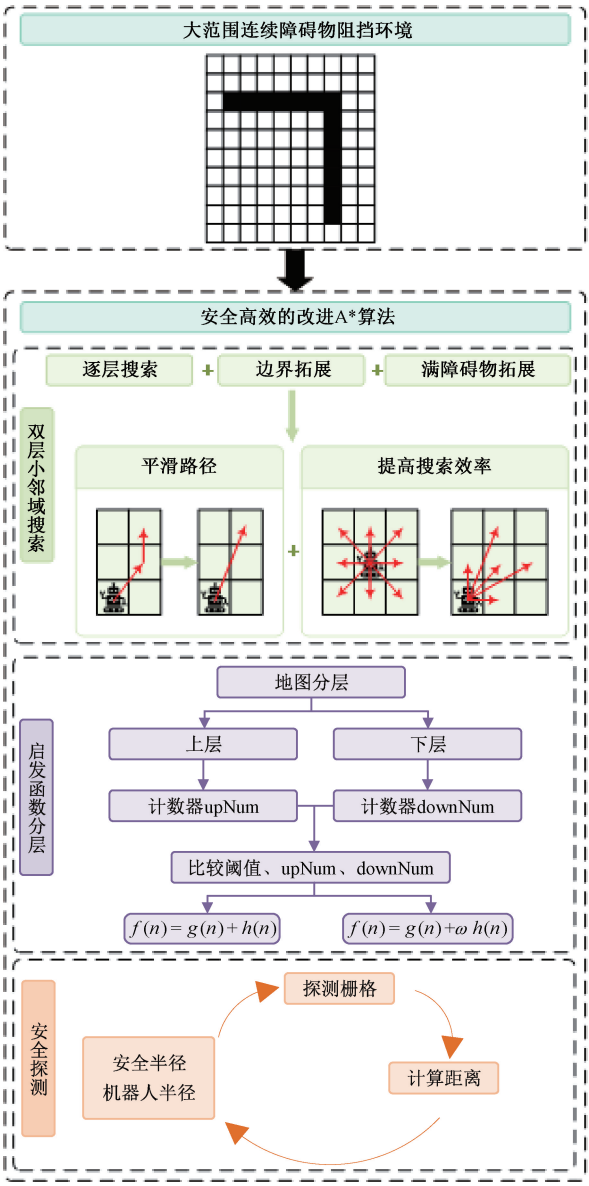


图 4　改进 A* 算法框架

Fig. 4　Flowchart of A* algorithm

Neighborhood 1；α 为 90，则搜索邻域为 Neighborhood 6；$\alpha \in (90,180)$，则搜索邻域为 Neighborhood 2；α 为 180，则搜索邻域为 Neighborhood 7；$\alpha \in (180,270)$，则搜索邻域为 Neighborhood 3；α 为 270，则搜索邻域为 Neighborhood 8；$\alpha \in (270,360)$，则搜索邻域为 Neighborhood 4。α 的值由式(5)~(7)得到。

$$\alpha = \begin{cases} \arccos\left(\dfrac{\boldsymbol{a}\cdot\boldsymbol{b}}{\|\boldsymbol{a}\| \times \|\boldsymbol{b}\|}\right), & y_G \geqslant y_S \\ 360 - \arccos\left(\dfrac{\boldsymbol{a}\cdot\boldsymbol{b}}{\|\boldsymbol{a}\| \times \|\boldsymbol{b}\|}\right), & y_G < y_S \end{cases} \tag{5}$$

$$\boldsymbol{a} = (x_G - x_S, y_G - y_S) \tag{6}$$

$$\boldsymbol{b} = (1,0) \tag{7}$$

式中：(x_S, y_S) 为起点坐标；(x_G, y_G) 为目标点坐标；$\boldsymbol{a}$ 是起点与目标点构成的向量；$\boldsymbol{b}$ 是 x 轴正方向的向量；α 是向量 $\boldsymbol{a}$ 与向量 $\boldsymbol{b}$ 之间的夹角。

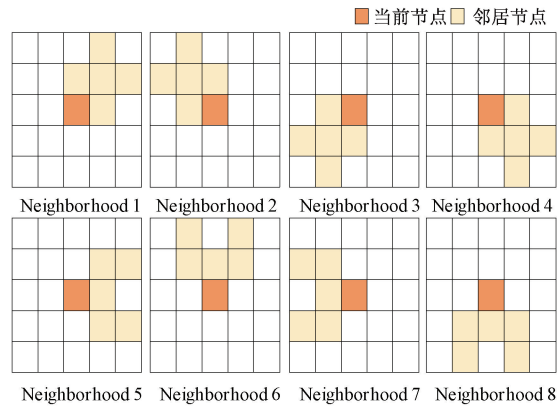


图 5　8 种双层 5 邻域

Fig. 5　Eight types of tow-layer 5-neighborhood

但双层 5 邻域会存在穿越障碍物(图 6(a))和死锁(图 6(b))问题。假设图 6(a)和(b)都是选择 Neighborhood 1 作为搜索邻域。图 6(a)中，邻居节点会选择当前节点作为父节点，从而会使路径穿越障碍物。图 6(b)中，到达目标点需要经过节点 1 或 2。A* 算法只能在搜索区域内搜索，从而导致无法经过节点 1 或 2 到达目标点。因此，该情况就称为死锁。为了解决穿越障碍物和死锁，设计了逐层搜索、边界拓展、满障碍物拓展。

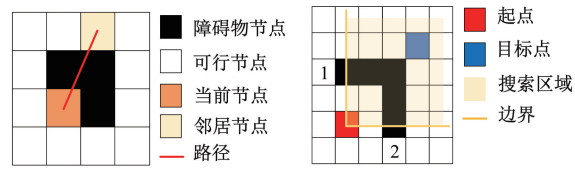


图 6　双层 5 邻域的缺点

Fig. 6　Disadvantage of tow-layer 5-neighborhood

1) 逐层搜索

为了提高搜索的灵活性，A* 算法对起点使用传统的 8 邻域搜索。进入 A* 算法的主体后，将双层 5 邻域划分为两层，分别为内层的 3 邻域和外层的 2 邻域。直线邻域和非直线邻域的双层邻域划分如图 7 所示。

穿越障碍物是由外层邻域导致的，因此逐层搜索策略先搜索内层邻域再搜索外层邻域。如果内层邻域有障碍物节点，则计算障碍物节点对外层邻域的安全值以标记外层邻域的节点是否需要搜索。安全值由式(8)得到。

$$V_s = \sqrt{(x_{out} - x_{obs})^2 + (y_{out} - y_{obs})^2} \tag{8}$$

式中：V_s 是安全值；(x_{out}, y_{out}) 是外层邻域节点坐标；

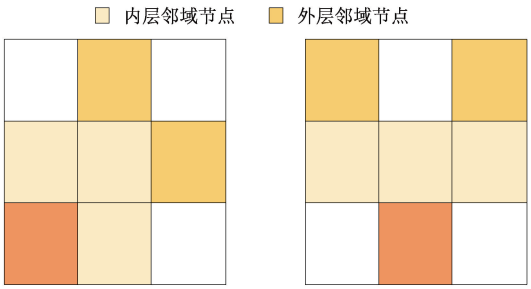


图7　双层5邻域的划分

Fig. 7　Division of tow-layer 5-neighborhood

(x_{obs}, y_{obs}) 是内层障碍物节点坐标。

如图8所示，直线和非直线邻域的最小安全值都是 $\sqrt{2}$。如果搜索完内层邻域后，外层邻域有安全值>$\sqrt{2}$的节点，则搜索这些节点。大于最小安全值的外层邻域节点能确保与当前节点的路径不会发生碰撞。

2）边界拓展

边界拓展能有效解决双层5邻域在搜索区域陷入死锁的问题。搜索区域内的5邻域是由式(5)的 α 确定的。但在搜索区域内无法到达目标点时，不再由 α 确定5邻域。边界拓展可以改变5邻域以使 A* 算法搜索到目标点。使用边界拓展需要知道搜索区域的边界。由于8种双层5邻域是确定的，因此每个双层5邻域的边界也

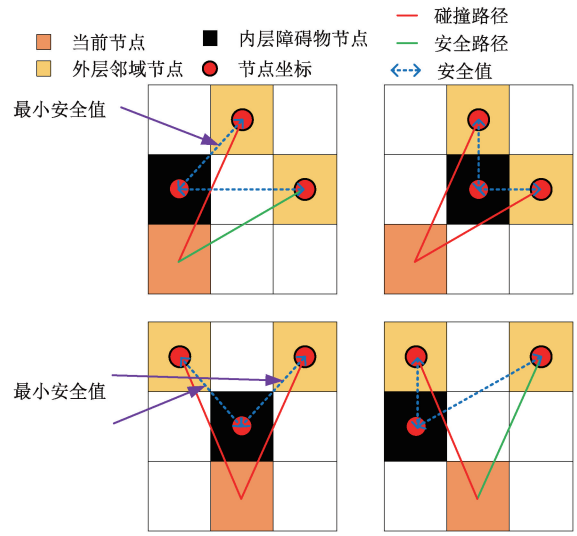


图8　最小安全值示意图

Fig. 8　Diagram of the minimum safety value

是确定的。每个双层5邻域边界和搜索区域的角度如表1所示。得到边界后，判断当前节点在边界上还是在搜索区域之外。如图9所示，如果当前节点在搜索区域外，则计算当前节点到起点的距离 d_1 和到目标点的距离 d_2。如果 $d_1 < d_2$，则计算向量 $\boldsymbol{l}_1$ 与式(7)向量的夹角角度 α_1；否则，计算向量 $\boldsymbol{l}_2$ 与式(7)向量的夹角角度 α_2。

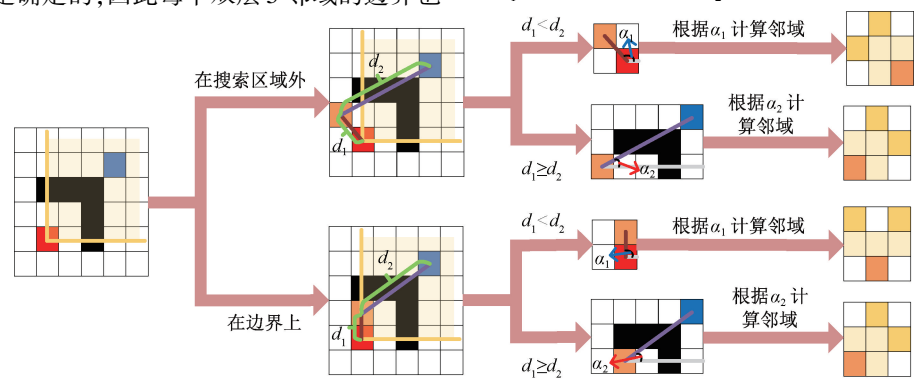


图9　边界拓展的划分

Fig. 9　Diagram of boundary expansion

表1　双层5邻域的边界和搜索区域

Table 1　Boundary and search region of the tow-layer 5-neighborhood

双层5邻域	边界1	边界2	搜索区域范围
Neighbor 1	0	90	[0,90]
Neighbor 2	90	180	[90,180]
Neighbor 3	180	270	[180,270]
Neighbor 4	270	360	[270,360]
Neighbor 5	315	45	[315,360)∪[0,45]
Neighbor 6	45	135	[45,135]
Neighbor 7	135	225	[135,145]
Neighbor 8	225	315	[225,315]

两个向量的夹角角度的计算与式(7)类似。根据夹角角度，边界拓展即可选出对应的双层5邻域。d_i 的公式如式(9)所示，$\boldsymbol{l}_1$ 和 $\boldsymbol{l}_2$ 的公式如式(10)和(11)所示。

$$d_i = \sqrt{(x_c - x_i)^2 + (y_c - y_i)^2} \tag{9}$$

$$\boldsymbol{l}_1 = (x_c - x_1, y_c - y_1) \tag{10}$$

$$\boldsymbol{l}_2 = (x_2 - x_c, y_2 - y_c) \tag{11}$$

其中，(x_c, y_c) 是当前节点的坐标，(x_1, y_1) 是起点目标点坐标，(x_2, y_2) 是起点目标点坐标。

3）满障碍物拓展

如图10所示，当5个邻居节点都不可搜索时，双层5邻域搜索会陷入局部死锁。因此，当5个邻居节点与当

前节点的路径都发生碰撞时,则原来的 5 邻域需要拓展额外的节点。根据搜索邻域的不同,拓展方式也不一样。直线邻域在当前节点的左右或上下新增邻居节点。非直线邻域根据图 2(a)新增额外的两个邻居节点。

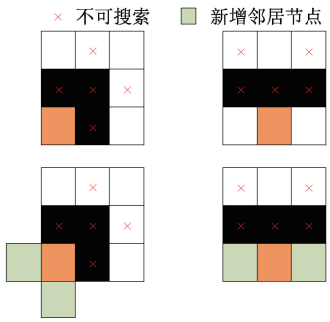


图 10　满障碍物示意图

Fig. 10　Diagram of full obstacle expansion

2.2　启发函数分层策略

如图 11 所示,在大范围连续障碍物阻挡的环境中 A* 算法会遍历过多的节点。A* 算法产生的路径在下侧,但 A* 算法会在上侧搜索路径。为了使 A* 算法在概率最大的一侧搜索路径,改进的 A* 算法使用启发函数分层策略。

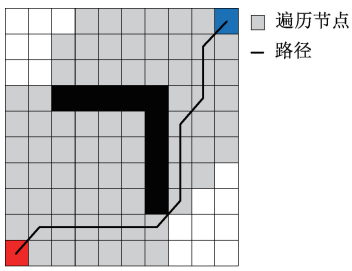


图 11　大范围连续障碍物环境中的结果

Fig. 11　Results in large-scale continuous obstacle environments

启发函数分层策略首先要确定分层的边界。分层策略将地图划分为两层,因此起点与目标点所连直线作为分层的边界。边界为式(15)所求直线。

$$A = y_G - y_S \tag{12}$$

$$B = x_S - x_G \tag{13}$$

$$C = x_G y_S - x_S y_G \tag{14}$$

$$Ax + By + C = 0 \tag{15}$$

其中,(x_S, y_S) 和 (x_G, y_G) 分别为起点和目标点。

如图 12 所示,在分层后的地图中,遍历节点可只会出现在上层、下层和边界。定义两个计数器 upNum 和 downNum 用于计算上层和下层遍历节点个数。并设置一个分层阈值 λ 用于判断是否将启发函数分层。假设地图大小为 $n \times m$,则 λ 的取值范围为 $[0, n \times m]$。λ 值的大小代表启发函数分层前初步探索的程度。λ 值相对地图越小,初步探索程度越低,也易陷入局部最优;λ 值相对地图越大,初步探索程度越高。当 λ 为 $n \times m$ 时等于 A* 算法,全局路径必定最优,但搜索效率如同 A* 算法一样低。因此,为了避免路径陷入局部最优以及提高搜索效率,λ 的值根据地图大小设置为 $\max\{n, m\}$。节点在下层时,downNum 的值+1。如果 downNum 的值 $\leqslant \lambda$,则 A* 算法使用原来的启发函数。此时,改进 A* 算法以探索为目的。如果 downNum 的值 $> \lambda$,改进 A* 算法结束探索模式,并进入分层搜索模式。在分层搜索模式中,如果 downNum 的值比 upNum 的值大,则说明改进的 A* 算法偏向于搜索下层;反之,改进 A* 算法偏向于搜索上层。当前节点在偏向搜索层时,该节点计算邻域启发值的函数为原 A* 算法的启发函数。如果当前节点不在偏向搜索层时,启发函数采用带权重的启发函数以增加非偏向搜索层节点的预估代价。这种启发函数分层策略能有效避免在非偏向搜索层遍历过多的节点,提高了时间效率和减少了内存消耗。一个遍历节点采用启发函数分层策略计算其邻居节点总代价的流程如图 13 所示。

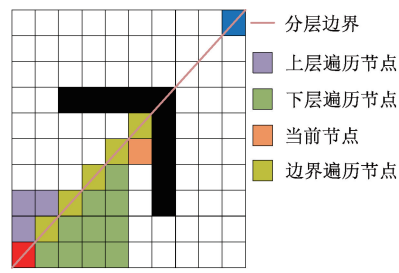


图 12　分层策略示意图

Fig. 12　Diagram of the hierarchical strategy

2.3　安全探测

A* 算法移动方式只有平移和斜移两种。路径与障碍物发生碰撞通常发生在斜移的情况。如图 14(a)所示,若不考虑移动机器人大小,障碍物与斜移生成的路径之间的距离为 0,从而导致碰撞。本文栅格大小为 1,机器人半径为 r_r,安全半径为 r_s。若 A* 算法要生成路径,r_r 和 r_s 必须都<0.5 倍栅格大小。此外,考虑避免碰撞后,r_s 取值范围为 $[r_r, 0.5)$,r_r 的取值范围为 $[0, 0.5)$。r_r 取 0 代表不考虑机器人大小。参数的图示意图如图 14(b)所示。

考虑参数的设置和正常生成路径的条件,平移必然不会发生碰撞。因此,安全探测方法仅针对斜移情况。安全探测的主要思想是探测具有威胁的栅格,然后判断这些栅格中是否有障碍物。如果有障碍物则计算这些障碍物与斜移的路径是否保持安全距离。如图 15 所示,紫

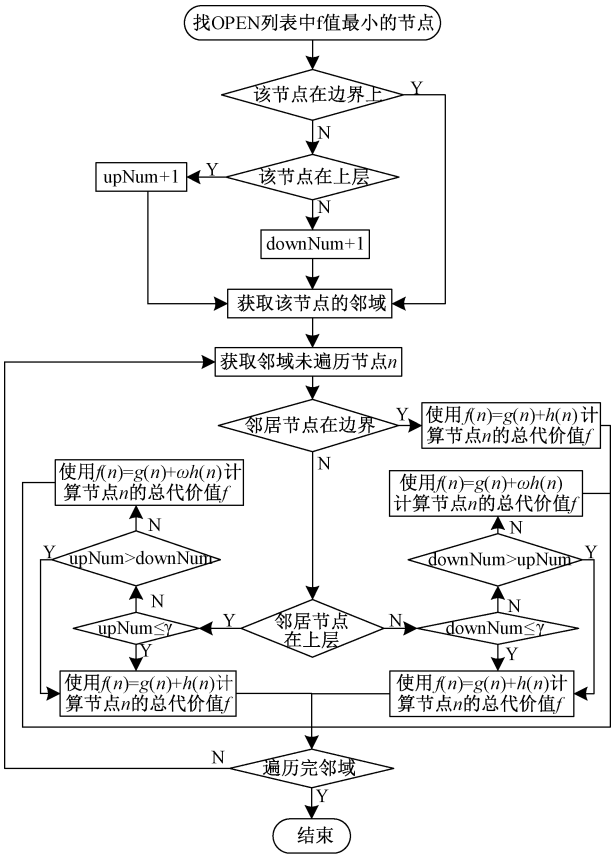
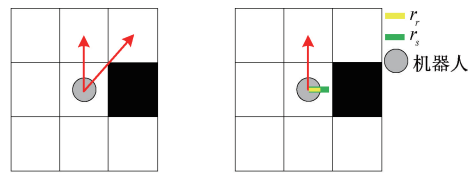


图 13　分层策略流程

Fig. 13　Flowchart of the hierarchical strategy



(a) 两种移动方式　(b) 参数示意图

(a) Tow motion modes　(b) Diagram of parameters

图 14　机器人碰撞的条件

Fig. 14　Collision conditions for robots

色栅格为在内层邻域和在外层邻域斜移时改进算法需要探测的栅格。内层或外层邻域探测栅格的坐标由式(16)和(17)求得。

$$D_1 = (x_c, y_N) \tag{16}$$

$$D_2 = (x_N, y_c) \tag{17}$$

其中，(x_c, y_c) 是当前节点坐标，(x_N, y_N) 是内层或外层的邻居节点坐标。

无论是内层邻域还是外层邻域的斜移都有两个探测栅格。为了确保路径安全，任一为障碍物的探测栅格都需与移动机器人保持 r_s 的安全距离，否则当前节点不能作为对应邻居节点的父节点。因此，安全探测方法首先

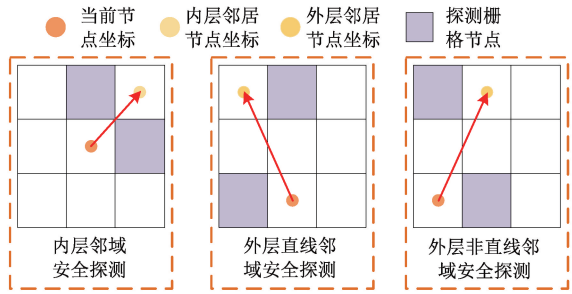


图 15　探测栅格示意图

Fig. 15　Diagram of detection grid

需要求出当前节点与邻域节点的直线。直线的表达式为式(18)。其次，安全探测方法还需获取探测栅格离直线最近的直角点坐标(O_1 和 O_2)。O_1 和 O_2 由式(20)和(21)得到。

$$(y_N - y_c)x + (x_c - x_N)y + (x_N y_c - x_c y_N) = 0 \tag{18}$$

$$I = ((x_c + x_N)/2, (y_c + y_N)/2) \tag{19}$$

$$O_1 = (I(1) + 0.5bin(I(1)), I(2) + 0.5bin(I(2))) \tag{20}$$

$$O_2 = (I(1) - 0.5bin(I(1)), I(2) - 0.5bin(I(2))) \tag{21}$$

$$bin(t) = \begin{cases} 1, & t \in \mathbf{Z} \\ 0, & t \notin \mathbf{Z} \end{cases} \tag{22}$$

式中：$bin(t)$ 是将输入 t 为整数时输出置 1 的二值化函数；I 是当前节点与邻居节点的中点；O_1 是 D_1 离直线最近的直角点坐标；O_2 是 D_2 离直线最近的直角点坐标。

任取 O_1 和 O_2 中的一个点，其坐标记为 $P(x_0, y_0)$。P 到直线的距离 d_l 由式(23)得到。如果 $d_l > r_s$，则说明路径与障碍物没有保持安全距离。因此，不选择当前节点作为 P 对应的探测栅格的父节点。当 P 为 O_1 时计算 d_l 的示意图如图 16 所示。

$$d_l = \frac{|(y_N - y_c)x_0 + (x_c - x_N)y_0 + (x_N y_c - x_c y_N)|}{\sqrt{(y_N - y_c)^2 + (x_c - x_N)^2}} \tag{23}$$

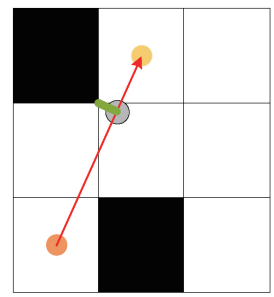


图 16　计算 d_l 示意图

Fig. 16　Diagram of calculating d_l

3　算法仿真与实验结果分析

所有实验均用 MATLAB R2023a 进行算法的仿真。电脑的处理器型号为 AMD Ryzen 7 7735H with Radeon Graphics 3.20 GHz,内存为 16.0 GB。所有实验使用的地图为栅格地图,栅格大小为 1。A* 算法和改进的 A* 算法使用的启发函数为欧氏距离。机器人半径 $r_r=0.3$,安全半径 $r_s=0.4$,分层阈值 λ 为地图长宽的最大值,分层启发函数权重 $\omega=1.3$。所有地图的环境都是静态的。地图中的黑色栅格为障碍物,白色栅格为可行区域,红色栅格为起点,蓝色栅格为目标点,灰色栅格为遍历节点,红色实线为路径。

衡量算法的指标有运行时间、路径长度、安全路段占比、转折点个数、遍历节点数和方向角变化总和。运行时间是 A* 算法从起点到目标寻路的所耗费的时间。由于运行时间受计算机设备影响从而产生波动,因此采用 20 次运行时间的平均值作为算法的运行时间。遍历节点数是拓展的节点数。路径长度是起点到目标的路程。转折点个数是指方向角发生变化的个数,其中方向角为以 x 轴正方向为起始边,逆时针旋转到机器人朝向所形成的角度。遍历节点数是寻路过程拓展的节点个数。方向角变化总和是方向角发生变化时,其变化幅度绝对值的累加。

3.1　各个改进模块的仿真与分析

每个改进模块针对的问题都不相同。双层小领域搜索策略(模块 1)是为了使路径更平滑。启发函数分层策略(模块 2)是为了减少遍历节点数。而安全探测(模块 3)是为了使机器人避免碰撞。为了验证每个模块的效果,使用大小为 30×30 的栅格地图(Map 1)对每个模块进行仿真实验。此外,对所有模块融合得到改进的 A* 算法,并在 Map 1 中对改进的 A* 算法进行仿真。Map 1 具有狭窄可行区域、障碍物密度较高以及大范围连续障碍物阻挡的特点。Map 1 中的起点为(11,2),目标点为(11,29)。图 17 所示是 A* 算法(图 17(a))、3 个模块(图 17(b)~(d))和改进 A* 算法(图 17(e))的仿真结果。模块 1 与 A* 算法相比,模块 1 生成的路径更平滑。模块 2 与 A* 算法相比,模块 2 的遍历节点数量显著减少了。模块 3 与 A* 算法相比,在斜移的区域模块 3 生成的路径比 A* 算法更安全。

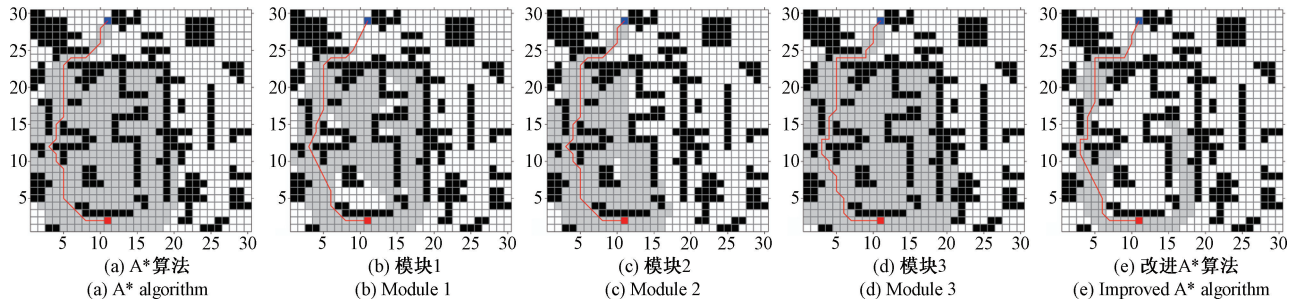


图 17　不同模块的仿真结果

Fig. 17　Simulation results of different modules

此外,在时间效率方面,模块 1 和模块 2 对提升时间效率都有贡献。如图 18 所示,与 A* 算法相比,模块 1 和模块 2 的平均运行时间分别减少了 21.9%、38.1%,改进的 A* 算法的平均运行时间减少了 61.8%。

运行时间的减少受遍历节点数的影响。如图 19 所示,模块 1 和模块 2 的遍历节点数分别减少了 20.8%、27.8%。改进 A* 算法的遍历节点数减少了 60.4%。而在路径长度方面,只有模块 1 的路径平滑效果会减少路径的长度。与 A* 算法相比,模块 1 的路径长度减少了 3.4%。模块 2 与 A* 算法的路径长度一样。由于避免碰撞导致的机器人斜移路径减少了,模块 3 和改进的 A* 算法不可避免地增加了路径的长度。但模块 3 和改进的 A* 算法的路径更安全。如表 2 所示,A* 算法和模块 2 的安全路段占比只有 69.1%,模块 2 对路径平滑后安全路段占比仅有 58.7%。而模块 3 和改进的 A* 算法的安全路段占比能达到百分比。

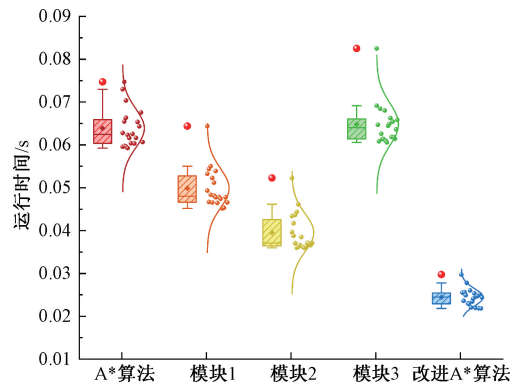


图 18　不同模块的运行时间

Fig. 18　Running time of different modules

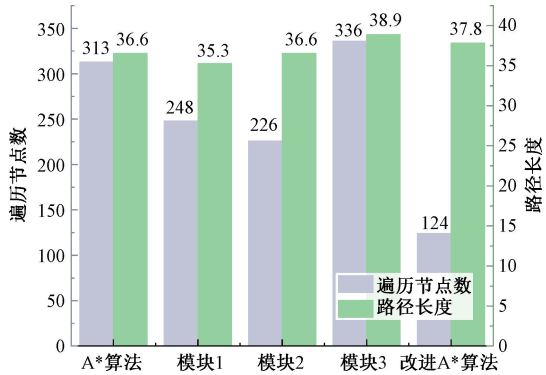


图 19　不同模块的遍历节点数和路径长度

Fig. 19　Traversed nodes and path lengths of different modules

表 2　3 个指标的实验数据

Table 2　Experimental data of three indicators

算法	安全路段占比/%	转折点个数	方向角变化总和/(°)
A* 算法	69.1	14	675
模块 1	58.7	11	402.8
模块 2	69.1	14	675
模块 3	100	17	945
改进 A* 算法	100	12	582.8

在路径平滑度方面,模块 1 和改进 A* 算法的指标优于 A* 算法。由表 2 可知,模块 1 和改进 A* 算法的转折点个数分别为 11、12。转折点数的减少意味着转向次数更小,从而使路径平滑。此外,转向次数的减少也降低了机器人的能量消耗。为了进一步体现模块 1 发挥的路径平滑效果,图 20 所示为路径方向角的变化幅度。A* 算法和模块 2 的方向角变化较为频繁,且最大变化幅度为 90°,最小变化幅度为 45°。为了避免碰撞,模块 3 的方向角变化更为频繁。而模块 1 的方向角变化次数减少了,并且最大变化幅度只有 53°,最小变化幅度仅有 18°。此外,由表 2 可知,模块 1 的方向角变化总和最小,与 A* 算法相比减少了 40.3%。融合 3 个模块的改进 A* 算法不仅保留了模块 1 的平滑路径性质,还继承了模块 3 的安全性。因此,虽然改进 A* 算法的路径仍有两次方向角变化幅度为 90°,但方向角变化总和与 A* 算法相比减少了 13.7%。

同时,为了验证分层阈值 λ 的合理性及对不同地图的普适性,使用 5 种不同的地图进行验证。5 种地图的大小分别为 30×30、60×60、90×90、120×120、150×150。每张地图使用从 0~λ 以 0.1 倍 λ 采样的 11 个不同的阈

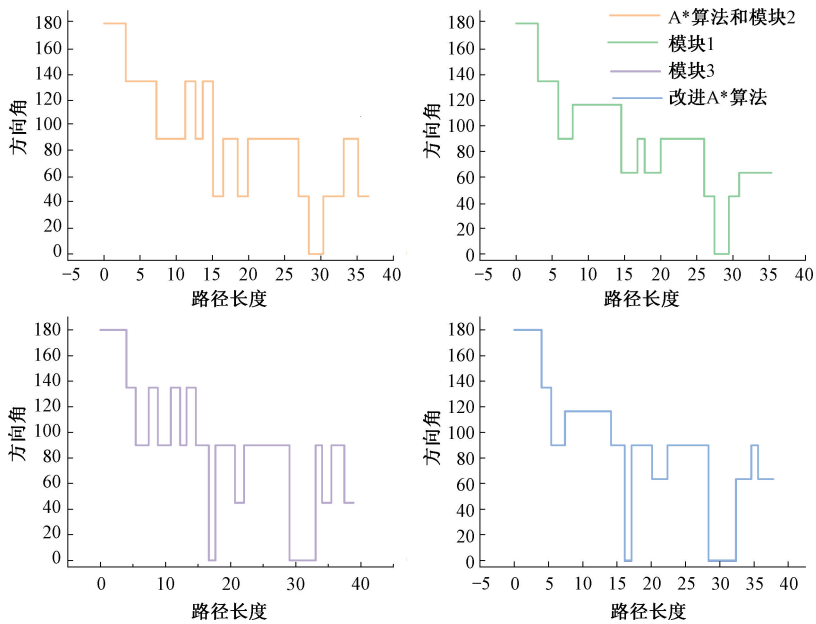


图 20　不同模块的方向角

Fig. 20　Direction angles of different modules

值进行实验。实验结果如图 21 所示,红色线为遍历节点数,蓝色线为路径长度,不同的符号表示不同的地图。图 21 中,当阈值在 0.9λ 附近,每个地图路径长度就已等于 A* 算法的路径长度,表明取阈值为 λ 能够确保最优性。同时,阈值在 0.9λ 附近遍历节点数已与最低遍历节点数接近,表明了阈值取 λ 既能保证最优性又能提高搜索效率。

综合而言,每个模块都有各自的优点,但同时也存在不足。通过融合 3 个模块,每个模块的优缺点进行互补。得到的改进 A* 算法的综合性能已超越 A* 算法,尤其在

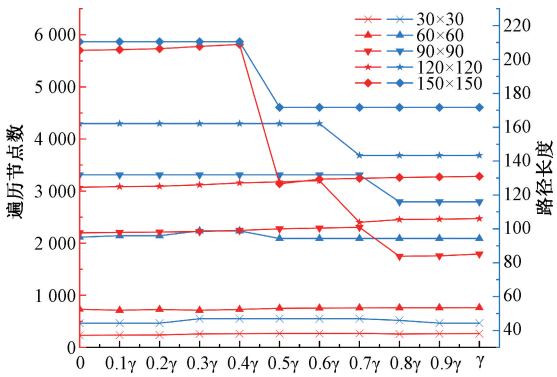


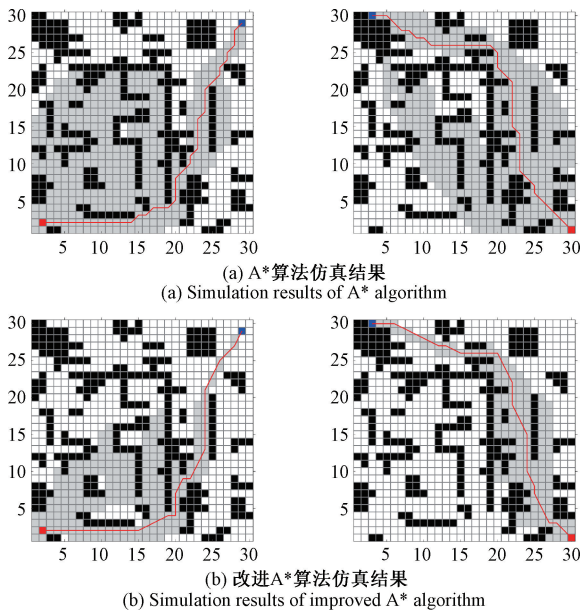
图 21　5 种地图下不同阈值的验证

Fig. 21　Verification of different thresholds under five maps

运行时间、遍历节点数和安全路段占比方面表现最佳。

3.2　不同起点和目标点的仿真与分析

为了验证改进 A* 算法的鲁棒性，使用两组与仿真 1 不同的起点和目标点在 Map 1 进行仿真。组 1 的起点为(2,2)，目标点为(29,29)。组 2 的起点(30,1)，目标点为(3,30)。组 1 的仿真结果如图 22(a)所示，组 2 的仿真结果如图 22(b)所示。与 A* 算法相比，改进的 A* 算法的路径更平滑、更安全，且遍历节点数更少。



(a) A*算法仿真结果
(a) Simulation results of A* algorithm

(b) 改进A*算法仿真结果
(b) Simulation results of improved A* algorithm

图 22　不同起点和目标点的仿真

Fig. 22　Simulations of different starting and goal points

从指标上分析，改进 A* 算法依然表现优越。如图 23 所示，改进 A* 算法的时间效率提升依然显著。与 A* 算法相比，两组改进 A* 算法的平均运行时间分别减少了 55.2%、66.4%。转折点个数、路径长度、安全路段占比和遍历节点数 4 个指标与 A* 算法的对比如图 24 所示。

从安全性上分析，两组实验的改进 A* 算法的安全路段占比都为 100%，而 A* 算法分别仅有 78.9%、82.1%。并且改进 A* 算法在确保安全性的同时，路径也是最短的。两组实验的改进 A* 算法路径长度分别减少了 0.5%、1.7%。从内存消耗上分析，改进 A* 算法的遍历节点数量显著减少，分别减少了 43.8%、57.6%。从路径平滑度上分析，两组改进 A* 算法的转折点个数分别减少了 47.4%、13.3%。此外，路径的方向角的变化次数和幅度都减少了。如图 25 所示，组 1 中改进 A* 算法的方向角变化次数显著减少。且改进 A* 算法的整体方向角变化幅度较小，方向角变化总和与 A* 算法相比提升了 54.8%。组 2 中改进 A* 算法的方向角变化次数也有所减少，与 A* 算法相比，改进 A* 算法的整体方向角变化幅度也较小，方向角变化总和与 A* 算法相比提升了 35.2%。

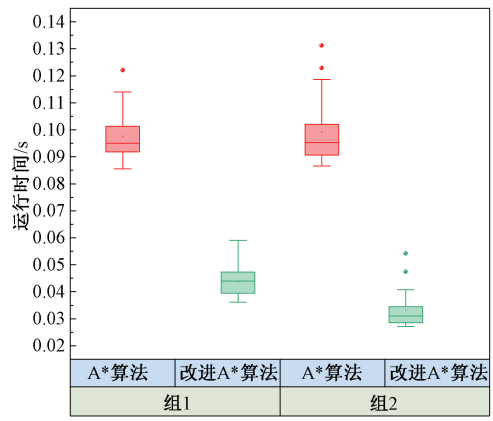


图 23　两个组的运行时间

Fig. 23　Running time of two groups

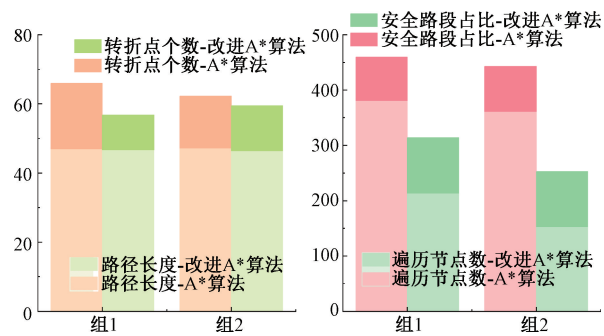


图 24　两个组的 4 个指标

Fig. 24　Four indicators of two groups

3.3　不同算法的仿真与分析

为了验证改进 A* 算法的优势，将改进 A* 算法与 A* 算法、Weighted A* 算法、JPS 算法、文献[20]算法及文献[26]算法进行比较。Weighted A* 算法、JPS 算法和文献[26]算法是高效的 A* 算法变体。而文献[20]算法是 A* 算

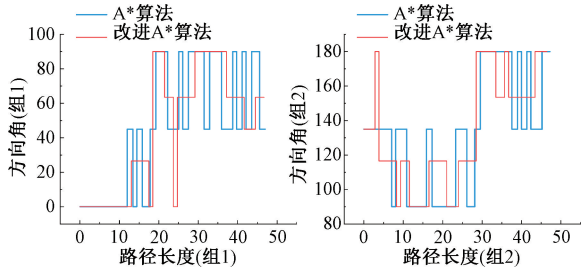


图 25　两个组的方向角

Fig. 25　Direction angles of two groups

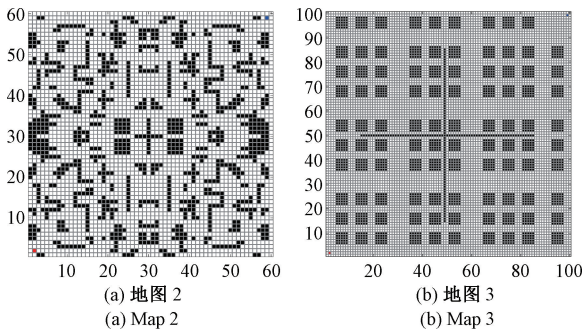


图 26　两种不同的场景

Fig. 26　Two different scenarios

法对安全性和平滑性的改进。此外,为了验证改进 A* 算法的可拓展性,仿真地图的尺寸增加到 100。图 26(a)所示是常规地图(Map 2),Map 2 具有障碍物密度高、可行区域较窄的特点。图 26(b)所示是大范围连续障碍物阻挡的地图(Map 3)。Map 3 具有可行区域较宽、障碍物密度适中、大范围连续障碍物阻挡的特点。Map 2 是为了验证对普通地图的适用性,而 Map 3 是为了验证可拓展性。Map 2 和 Map 3 的起点都为(2,2),目标点分别为(59,59)、(99,99)。使用图 26 的两个地图对每个算法进行仿真实验。5 种算法及改进 A* 算法的仿真结果如图 27 和 28 所示。仿真实验数据如表 3、4 所示。

与 A* 算法相比,在两种地图中改进 A* 算法的运行时间、安全路段占比、遍历节点数和方向角变化总和均优于 A* 算法。改进 A* 算法的运行时间平均减少了 79.1%,安全路段占比平均增加了 16.1%,遍历节点数平均减少了 56.7%,方向角变化总和平均减少了 15.8%。由于 Map 3 可行区域较窄和障碍物密度高,A* 算法路径发生碰撞的路段多。为了避免碰撞,改进 A* 算法将发生碰撞的斜移改成平移,从而增加了路径长度和转折点个数。但在 Map 2 中改进 A* 算法的路径长度和转折点个数依然有所减少。与 A* 算法相比,改进 A* 算法在

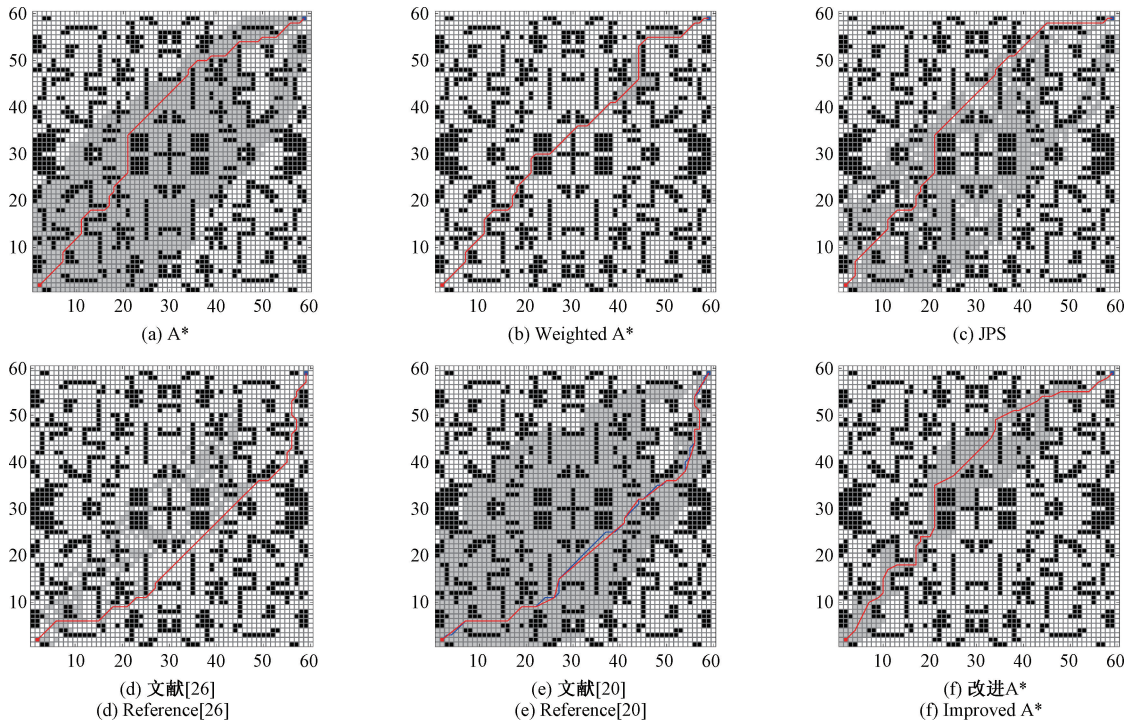


图 27　在 Map 2 中 6 种算法的仿真结果

Fig. 27　Simulation results of six algorithms in Map 2

Map 2 和 Map 3 上的表现证明了其可拓展性、普适性、高效性以及高安全性。

与高效的 Weighted A* 算法、JPS 算法和文献[26]算法相比,改进 A* 算法具有更强的鲁棒性。在 Map 2 中,

表 3　在 Map 2 中不同算法的实验数据

Table 3　Experimental data of different algorithms in Map 2

指标	A* 算法	Weighted A* 算法	JPS 算法	文献[26]算法	文献[20]算法	改进 A* 算法
运行时间/s	0.427 3	**0.035 5**	0.223 8	0.131 8	0.574 6	0.071 8
路径长度(/栅格大小)	**90.6**	91.2	**90.6**	90.8	91.5	92.4
安全路段占比/%	81.5	76.9	78.4	79.9	**100**	**100**
转折点个数	24	25	**19**	21	**19**	27
遍历节点数	1 347	**106**	870	298	1 516	420
方向角变化总和/(°)	1 080	1 125	855	945	**847.7**	1 010.6

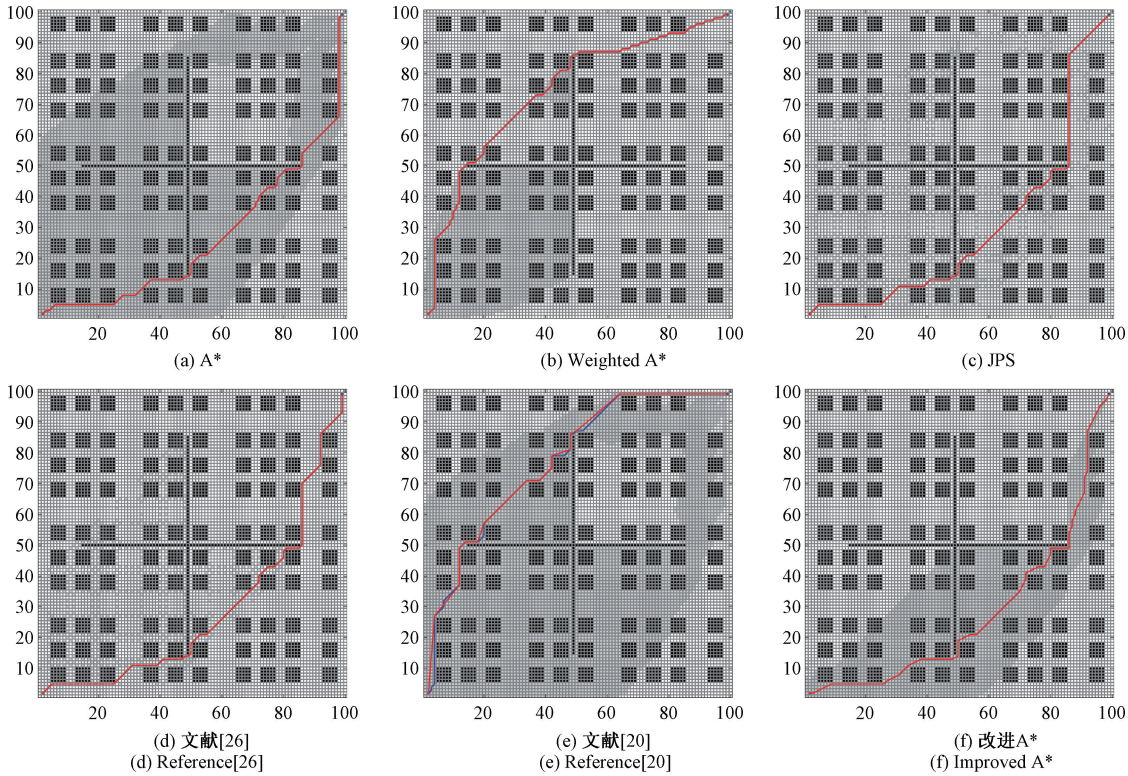


图 28　在 Map 3 中 6 种算法的仿真结果

Fig. 28　Simulation results of six algorithms in Map 3

表 4　在 Map 3 中不同算法的实验数据

Table 4　Experimental data of different algorithms in Map 3

指标	A* 算法	Weighted A* 算法	JPS 算法	文献[26]算法	文献[20]算法	改进 A* 算法
运行时间/s	2.958 6	0.699 5	0.460 4	**0.274 6**	3.366 2	0.474 3
路径长度(/栅格大小)	162.4	162.4	162.4	162.4	163.5	**161.6**
安全路段占比/%	89.6	90.5	91.3	91.3	**100**	**100**
转折点个数	28	47	22	25	**13**	31
遍历节点数	4 531	1 578	1 036	**637**	4 836	2 180
方向角变化总和/(°)	1 260	2 115	990	1 125	**657.1**	1 085.9

Weighted A* 算法的时间效率最高，运行时间为 0.035 5 s。改进 A* 算法的运行时间是 Weighted A* 算法的 2.02 倍，而 JPS 算法和文献[26]算法的运行时间却分别是 Weighted A* 算法的 6.30 倍、3.71 倍。在 Map 3 中，文献[26]算法的时间效率最高，运行时间为 0.274 6 s。但改进 A* 算法的运行时间仅是文献[26]算法的 1.72 倍。在不同的场景下，改进 A* 算法始终能保持较低的运行时间。此外，在不同场景下 Weighted A* 算法和 JPS 算

法的遍历节点数和方向角变化总和并不总是优于改进 A* 算法。因此,改进 A* 算法的鲁棒性比 Weighted A* 算法、JPS 算法和文献[26]算法更强。为了保证路径的安全,不同场景下改进 A* 算法的转折点数不可避免地比 Weighted A* 算法、JPS 算法和文献[26]算法多。但改进 A* 算法的路径更安全,不同场景下安全路段占比分别比 Weighted A* 算法、JPS 算法、文献[26]算法平均提升了 20.3%、18.6%、17.4%。在路径长度指标上,Map 3 中改进 A* 算法的路径长度比 Weighted A* 算法、JPS 算法、文献[26]算法短,而在 Map 2 中比所有算法都长。随着安全性的提高,路径长度通常会增加。

由表3可知,文献[20]算法转折点个数、方向角变化总和均优于改进 A* 算法。文献[20]算法对 A* 算法的邻域排除强迫邻居节点,得到一次规划的安全路径。如图 27(e)和 28(e)的蓝线所示,这样的方法并没有缩小搜索邻域,从而导致为了确保安全性降低了时间效率。而文献[20]算法为了进一步提高路径的平滑度,将一次规划的路径提取转折点个数并去除冗余路径节点。此外,文献[20]还进一步根据障碍物保留必经路径节点。最终,二次规划的路径如图 27(e)和 28(e)的红线所示。可以看到,路径比本文改进算法更平滑,这也是文献[20]算法转折点个数、方向角变化总和均优于改进 A* 算法的原因。但该方法导致的问题就是时间效率不高且遍历节点数过多。在不同场景下,改进 A* 算法的运行时间和遍历节点数分别比文献[20]算法平均减少了 86.7%、63.6%。此外,如图 27 和 28 所示,文献[20]算法生成的路径不一定是最优解。而改进的 A* 算法不仅在不同场景下生成的路径是最优解,同时其安全路段占比也与文献[20]算法一样达到 100%。如图 29 所示,文献[20]算法的方向角变化次数少于改进 A* 算法,但改进 A* 算法的方向角变化最小幅度更小,且方向角小幅度变化的次数更多。因此,与文献[20]的高安全性和平滑性算法相比,改进 A* 算法既能保持相同的安全性和较高的平滑性,又具备高的时间效率、低的内存消耗和能找到最优解。

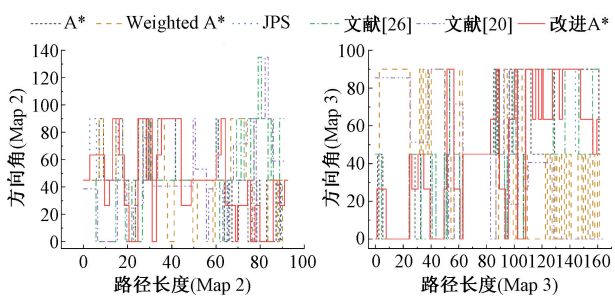


图 29　不同算法的方向角

Fig. 29　Direction angles of different algorithms

4　结　论

针对 A* 算法在大范围连续障碍物环境中搜索效率低、安全性不足的问题,本文设计了一种安全高效的改进 A* 算法。首先,改进 A* 算法通过设计的双层小邻域搜索策略解决了使用小邻域搜索提高搜索效率会陷入死锁的问题。双层小邻域搜索采用的边界拓展和满障碍物拓展策略兼顾搜索效率和避免陷入死锁。其次,改进 A* 算法通过将搜索区域分层并根据分层阈值使用不同的启发函数的启发函数分层策略,解决了遍历节点数过多的问题,从而进一步提高搜索效率。同时,改进的 A* 算法提出的安全探测确保了路径的安全性。现有研究中,移动机器人路径规划的优化往往聚焦于效率和安全性其中一个点。而本文提出的改进的 A* 算法同时对这两方面性能进行提升且不依赖二次规划,为移动机器路径规划提供了更安全高效的方案。改进 A* 算法不仅在大范围连续障碍物环境中有着很好的综合性能,而且在普通环境中也能保持其高效性和安全性。对于多种环境,改进 A* 算法具有更强的鲁棒性、优越的综合性能和可拓展性。改进 A* 算法的平滑性不够好,未来若要进一步从算法本质上提高路径平滑度且兼具时间效率,可增加搜索邻域的层数同时减少搜索方向,但这会存在极难处理的越障问题和陷入死锁问题。

参考文献

[1] LIU L, WANG X, YANG X, et al. Path planning techniques for mobile robots: Review and prospect[J]. Expert Systems with Applications, 2023, 227: 120254.

[2] WANG J, CHI W, LI C, et al. Neural RRT*: Learning-based optimal path planning [J]. IEEE Transactions on Automation Science and Engineering, 2020, 17(4): 1748-1758.

[3] YANG T Y, ZOU L, WU Z X, et al. An improved A-star algorithm coupled with graph division and AIS data for ship path planning[J]. Ocean Engineering, 2025, 330: 121234.

[4] MIYOMBO M E, LIU Y, MULENGA C M, et al. Optimal path planning in a real-world radioactive environment: A comparative study of A-star and Dijkstra algorithms[J]. Nuclear Engineering and Design, 2024, 420: 113039.

[5] HAN Z, SUN H, HUANG J, et al. Path planning algorithms for smart parking: Review and prospects[J]. World Electric Vehicle Journal, 2024, 15(7): 322.

[6] GHAMBARI S, GOLABI M, JOURDAN L, et al. UAV path planning techniques: A survey [J]. RAIRO-Operations Research, 2024, 58(4): 2951-2989.

[7] 许瑶,刘晓峰. 基于改进 RRT 算法的无人车路径规划研究与测试[J]. 国外电子测量技术,2023,42(8):132-138.
XU Y, LIU X F. Research and test of unmanned vehicle path planning based on improved RRT algorithm[J]. Foreign Electronic Measurement Technology, 2023, 42(8): 132-138.

[8] LI Y, JUAN R, ZHOU Y, et al. AD-RRT*: An RRT*-based global path planning approach for underwater gliders with alpha shapes and DBSCAN[J]. Expert Systems with Applications, 2025, 291: 128219.

[9] ZHOU X, WANG X, XIE Z, et al. A Collision-free path planning approach based on rule guided lazy-PRM with repulsion field for gantry welding robots[J]. Robotics and Autonomous Systems, 2024, 174: 104633.

[10] WU L, HUANG X, CUI J, et al. Modified adaptive ant colony optimization algorithm and its application for solving path planning of mobile robot[J]. Expert Systems with Applications, 2023, 215: 119410.

[11] 肖金壮,余雪乐,周刚,等. 一种面向室内 AGV 路径规划的改进蚁群算法[J]. 仪器仪表学报,2022,43(3):277-285.
XIAO J ZH, YU X L, ZHOU G, et al. An improved ant colony algorithm for indoor AGV path planning [J]. Chinese Journal of Scientific Instrument, 2022, 43(3): 277-285.

[12] CHEN K, YOU B, ZHANG Y, et al. Automatic lift path planning of prefabricated building components using semantic BIM, improved A* and GA[J]. Engineering, Construction and Architectural Management, 2025, 32(6): 4100-4124.

[13] 朱洪波,殷宏亮. 分层平滑优化 A* 引导 DWA 用于机器人路径规划[J]. 电子测量与仪器学报,2024,38(9):155-168.
ZHU H B, YIN H L. Hierarchical smoothing optimization A*-guided DWA for robot path planning[J]. Journal of Electronic Measurement and Instrumentation, 2024, 38(9): 155-168.

[14] 李芳娣,邓晓燕,吴伟铭,等. 移动机器人复杂角点定位和停车策略研究与实现[J]. 电子测量技术,2023,46(22):26-31.
LI F D, DENG X Y, WU W M, et al. Research and implementation of complex corner location and parking strategy for mobile robot[J]. Electronic Measurement Technology,2023,46(22):26-31.

[15] ZHANG W, WANG N, WU W. A hybrid path planning algorithm considering AUV dynamic constraints based on improved A* algorithm and APF algorithm[J]. Ocean Engineering, 2023, 285: 115333.

[16] LIN Z, WU K, SHEN R, et al. An efficient and accurate A-star algorithm for autonomous vehicle path planning [J]. IEEE Transactions on Vehicular Technology, 2023, 73(6): 9003-9008.

[17] AINE S, SWAMINATHAN S, NARAYANAN V, et al. Multi-heuristic A [J]. The International Journal of Robotics Research, 2016, 35(1-3): 224-243.

[18] DOLGOV D, THRUN S, MONTEMERLO M, et al. Practical search techniques in path planning for autonomous driving[J]. Ann Arbor, 2008, 1001(48105): 18-80.

[19] ZHANG H, TAO Y, ZHU W. Global path planning of unmanned surface vehicle based on improved A-Star algorithm[J]. Sensors, 2023, 23(14): 6647.

[20] 杨国,吴晓,肖如奇,等. 改进 A* 算法的安全高效室内全局路径规划[J]. 电子测量与仪器学报,2024,38(7):131-142.
YANG G, WU X, XIAO R Q, et al. Improved A* algorithm for secure and efficient indoor global path planning[J]. Journal of Electronic Measurement and Instrumentation, 2024, 38(7): 131-142.

[21] 姜媛媛,张阳阳. 改进 8 邻域节点搜索策略 A* 算法的路径规划[J]. 电子测量与仪器学报,2022,36(5):234-241.
JIANG Y Y, ZHANG Y Y. Improved path planning of A* algorithm of domain node search strategy 8[J]. Journal of Electronic Measurement and Instrumentation, 2022, 36(5): 234-241.

[22] 赖荣燊,窦磊,巫志勇,等. 融合改进 A* 算法和动态窗口法的移动机器人路径规划[J]. 系统仿真学报,2024,36(8):1884-1894.
LAI R S, DOU L, WU ZH Y, et al. Fusion of improved A* and dynamic window approach for mobile robot path planning [J]. Journal of System Simulation, 2024, 36(8): 1884-1894.

[23] 董雅文,杨静雯,张宝锋,等. 改进邻域扩展 A* 算法的移动机器人路径规划[J]. 机械设计与制造,2025(1):291-295.
DONG Y W, YANG J W, ZHANG B F, et al. An improved A* algorithm based on neighbor expansion for mobile robot path planning[J]. Machinery Design & Manufacture, 2025(1): 291-295.

[24] 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.

[25] WANG F, SUN W, YAN P, et al. Research on path planning for robots with improved A* algorithm under

bidirectional JPS strategy[J]. Applied Sciences, 2024, 14(13): 5622.

[26] SUN Z, LUO Q, ZHANG Z, et al. An integrated path planning and tracking framework based on adaptive heuristic JPS and B-Spline optimization[J]. Machines, 2025, 13(8): 710.

[27] YANG L, LI P, WANG T, et al. Multi-area collision-free path planning and efficient task scheduling optimization for autonomous agricultural robots [J]. Scientific Reports, 2024, 14(1): 18347.

[28] ALI Z A, YAKOVLEV K. Safe interval path planning with kinodynamic constraints [C]. Proceedings of the AAAI Conference on Artificial Intelligence, 2023, 37(10): 12330-12337.

[29] MENG S, WANG Y, YANG C F, et al. LLM-A*: Large language model enhanced incremental heuristic search on path planning [J]. ArXiv preprint arXiv: 2407.02511, 2024.

作者简介

黄家博,2023 年于西南科技大学获得学士学位,现为西南科技大学硕士研究生,主要研究方向为移动机器人路径规划。
E-mail: jbhuang1219@163.com

Huang Jiabo received his B. Sc. degree from Southwest University of Science and Technology in 2023. Now he is a M. Sc. candidate at Southwest University of Science and Technology. His main research interest includes mobile robot path planning.

陈春梅(通信作者),2000 年于西南科技大学获得学士学位,2010 年于西南科技大学获得硕士学位,2020 年于中国工程物理研究院获得博士学位,现为西南科技大学副教授,主要研究方向为人工智能和机器视觉。
E-mail: ccm@mails.swust.edu.cn

Chen Chunmei (Corresponding author) received her B. Sc. degree from Southwest University of Science and Technology in 2000, M. Sc. degree from Southwest University of Science and Technology in 2010, and Ph. D. degree from China Academy of Engineering Physics in 2020, respectively. Now she is an associate professor in Southwest University of Science and Technology. Her main research interests include artificial intelligence and computer vision.

龚渔民,2023 年于成都工业学院获得学士学位,现为西南科技大学硕士研究生,主要研究方向为可见光与红外图像融合。
E-mail: 3251898216@qq.com

Gong Yumin received his B. Sc. degree from Chengdu Technological University in 2023. Now he is a M. Sc. candidate at Southwest University of Science and Technology. His main research interest includes visible and infrared image fusion.

刘桂华,1993 年于西南科技大学获得学士学位,2001 年于武汉理工大学获得硕士学位,2012 年于西南交通大学获得博士学位,现为西南科技大学教授,主要研究方向为机器人场景智能感知和 FPGA 集成电路设计。
E-mail: liuguihua@swust.edu.cn

Liu Guihua received her B. Sc. degree from Southwest University of Science and Technology in 1993, M. Sc. degree from Wuhan University of Technology in 2001, and Ph. D. degree from Southwest Jiaotong University in 2012, respectively. Now she is a professor in Southwest University of Science and Technology. Her main research interests include robotic scene perception and FPGA integrated circuit design.

徐敏,2016 年于西华师范大学获得学士学位,2023 年于西南交通大学获得博士学位,现为西南科技大学讲师,主要研究方向为光学三维传感和精密三维测量。
E-mail: xumin_9302@163.com

Xu Min received her B. Sc. degree from China West Normal University in 2016, M. Sc. degree and Ph. D. degree from Southwest Jiaotong University in 2023, respectively. Now she is a lecturer in Southwest University of Science and Technology. Her main research interests include optical 3D sensing and precision 3D measurement.