改进 A* 算法融合改进动态窗口法的移动机器人路径规划

王志特[1]，罗丽平[1,2*]，廖义奎[1,2]

（1.广西民族大学电子信息学院，广西 南宁 530006；2.广西高校智慧无人系统与智能装备重点实验室，广西 南宁 530006）

摘　要：针对机器人路径规划对于路径最短、搜索效率以及平滑度的性能要求，提出一种改进 A* 算法与改进动态窗口法（DWA）相融合的算法。针对传统 A* 算法在复杂场景下输出非最优路径、寻路效率低等问题，结合曼哈顿距离和对角线距离设计新的启发函数，并对其动态分配权重，实现全局路径最短，减少寻路时间。针对传统 8 邻域 8 方向搜索方式搜索效率低、耗时长等问题，提出一种基于 8 邻域改进的搜索策略，对当前节点实时动态分配最优的搜索方向。针对路径存在多余无用节点的问题，使用 Floyd 算法去除冗余节点，减少转向次数，缩短路径长度。针对传统动态窗口法规划的路径非全局最优、目标点附近存在障碍物时规划的路径长度增加或者规划失败的问题，加入全局关键节点信息和引入目标点距离评估子函数。针对关键节点距离较长导致融合算法规划的路径偏离全局最优路径的问题，提出关键点密集化策略。最后，将提出的改进 A* 算法、融合算法和已有的其他改进算法进行比较，仿真结果表明：改进的 A* 算法能够在复杂环境中生成最短全局路径，平均转向次数减少 16.3%，平均寻路时间缩短 55.66%；融合算法在临时障碍物环境下，平均路径长度和平均运行时间分别缩短 6.1%和 14.7%，在移动障碍物环境下，平均路径长度和平均运行时间分别缩短 1.6%和 39.8%。

关键词：路径规划；A* 算法；动态窗口法；复杂环境；时间效率

中图分类号：TP242　　**文献标志码：**A　　**DOI：**10.19678/j.issn.1000-3428.0068483

Mobile Robot Path Planning by Improved A* Algorithm Fused with Improved Dynamic Window Approach

WANG Zhite[1], LUO Liping[1,2*], LIAO Yikui[1,2]

(1. School of Electronic Information, Guangxi Minzu University, Nanning 530006, Guangxi, China;
2. Key Laboratory of Intelligent Unmanned System and Intelligent Equipment in Guangxi Universities, Nanning 530006, Guangxi, China)

【Abstract】To satisfy the performance requirements for robot path planning, an algorithm integrating improved A* algorithm and improved Dynamic Window Approach(DWA) is proposed, which shortens the path length and improves the searching efficiency and path smoothness. To combat the challenges of the traditional A* algorithm in complex scenarios, a new heuristic function is designed based on Manhattan distance and the diagonal distance. The weights are assigned dynamically, and the global shortest path and the least searching time are obtained. Next, an improved search strategy based on the 8-neighborhood is proposed, which involves dynamically assigning the optimal search direction to the current node, thus improving the searching efficiency and reducing the time consumption compared to the traditional 8-neighborhood 8-direction search method. Subsequently, the Floyd algorithm is employed to remove redundant nodes, reduce the steering times, and shorten the path distance. Additionally, the traditional DWA faces certain challenges; for instance, the path is not globally optimal, the path planning may fail, or the path length may increase. To solve these problems, a keypoint densification strategy is proposed to modify the deflective path. Finally, the proposed improved A* algorithm and fusion algorithm are compared with existing methods. The simulation results show that the improved A* algorithm can generate the shortest global path in complex environments, reducing the average steering time by 16.3% and shortening the average path searching time by 55.66%. For the fused algorithm, the average path length and average runtime shorten by 6.1% and 14.7% in the temporary obstacle environment, respectively, and shorten by 1.6% and 39.8%, respectively, in the moving obstacle environment.

【Key words】path planning; A* algorithm; Dynamic Window Approach(DWA); complex environments; time efficiency

0　引言

路径规划是指移动机器人在空间环境中自主寻找一条最优的无障碍路径。当移动机器人可以获取到周围环境中所有障碍物的完整信息时，这样的路径规划被分类为全局路径规划。反之，如果机器人

收稿日期：2023-09-27　　**修回日期：**2023-11-28

基金项目：国家自然科学基金（61762011）；广西自然科学基金（2018GXNSFAA294059）；广西科技重大专项（2023AA17010）。

通信作者 E-mail：* luoliping@gxmzu.edu.cn

计算机工程 www.ecice06.com

只能获取到局部的、不完整的障碍物信息，那么这样的路径规划被归为局部路径规划。在目前的研究中，全局路径规划常用的算法包括 A* 算法[1-3]、Dijkstra 算法[4-6]、D* 算法[7-8]等，而局部路径规划则主要依赖于动态窗口法（DWA）[9-11]、人工势场法[12-14]等手段来实现。

近些年，国内外学者们对路径规划算法展开了深入的研究。为了提高 A* 算法的寻路效率，现有研究采用优化子节点选择方法[15]、改进启发函数的方法[16]或者双向交替搜索的方式[17]，然而，在复杂场景中，启发函数的度量方式可能并不总是最优的，导致全局路径轨迹质量难以保证。相比之下，文献[18]使用对角线距离作为新的度量方式，并加入了权重因子来提高路径搜索效率，但该方法在前期搜索方向受限，无法保证输出最短全局路径。针对 DWA 的改进，文献[19]引入了全局路径弯曲因子。具体来说，当前节点与相邻前后节点的夹角为锐角或直角时，机器人须减速通过，为钝角时则加速通过，但如果当前节点与相邻前后节点的距离较远且夹角不为平角时，机器人的行驶速度会非常缓慢，导致导航效率低下。该文提到，当 2 个子目标点距离较远时，可在这 2 个目标点之间插入适当的点。然而，如何确定插入点的数量和位置，文中并未给出实际的解决方案。另一方面，现有研究对改进的 A* 算法做了进一步优化。例如，文献[20]通过提取关键点作为融合算法的局部导航子目标来提高导航效率。然而，这种方法并未对关键点进行密集化处理，当相邻 2 个子目标点距离较远时，规划的路径可能无法贴合全局最优路径。此外，文献[21]提出的融合算法在目标点靠近障碍物的情况下可能无法规划出最短路径，甚至可能规划失败。

随着应用场景越来越多，现有算法要能够实现在任意地图下实时地输出一条转折点少、足够平滑的最短行驶路径仍是一项具有挑战性的工作。为了满足机器人导航过程中的最优性和实时性，必须解决以下问题：

1）对于全局路径规划来说，实时输出一条最短的全局路径至关重要，需要最大化地降低 A* 算法在复杂环境下陷入局部极小值的可能，保证算法输出最短的全局路径。其次，在输出最短全局路径的基础上，尽可能地缩短寻路时间和减少路径转向次数，以此来满足全局路径规划算法最优性和实时性的要求。

2）对于局部路径规划来说，在保证能安全抵达目标点的前提下，需要使机器人的行驶路径更加贴合全局最优路径，尽可能地缩短路径长度和运行时间。

针对当前移动机器人路径规划面临的上述问题，本文提出一种改进 A* 算法与改进 DWA 相融合的算法，主要工作包括：

1）优化传统 A* 算法中距离的度量方式，并加入权重项实现了启发函数的动态调节，通过这种方法可以更加准确地估计每个节点到目标节点的距离，减少无用节点的扩展，提高寻路效率。

2）优化传统 8 邻域 8 方向搜索方式，对当前节点实时动态分配最优的搜索方向，减少无用节点的拓展，缩短寻路时间。

3）改进 A* 算法规划的路径仍存在多余无用节点，因此使用 Floyd 算法对路径进一步优化，提取出关键节点，并对关键节点密集化处理，得到优化后的全局最优路径点。

4）在原动态窗口法中引入目标点距离评估子函数，并将密集化后的关键点融入方位角评估子函数。

1 复杂环境下改进的 A* 算法

针对传统 A* 算法无法满足路径规划算法的实时性和最优性等性能要求的问题，对传统 A* 算法进行改进，使其更加适合在复杂环境中运行，获得更高的时间效率和更平滑的最短行驶路径。

1.1 传统 A* 算法

A* 算法是一种启发式全局路径规划算法，是在 Dijkstra 算法基础上演变而来的，其通过迭代函数，从出发点向目标点方向的邻节点进行拓展求解。传统 A* 算法的评估函数表示为：

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

式中：$g(n)$ 表示出发点到当前节点的实际代价值；$h(n)$ 表示当前节点到目标点的预估代价值。启发函数通常用欧几里得距离、曼哈顿距离和对角线距离来表示，这 3 种距离的计算公式为：

$$d_{\text{Enclid}}=\sqrt{(x_c-x_g)^2+(y_c-y_g)^2} \tag{2}$$

$$d_{\text{Manhattan}}=|x_c-x_g|+|y_c-y_g| \tag{3}$$

$$d_{\text{Diagnol}}=|x_c-x_g|+|y_c-y_g|+(\sqrt{2}-2)\min(|x_c-x_g|,|y_c-y_g|) \tag{4}$$

1.2 基于曼哈顿距离和对角线距离的启发函数设计

面对复杂场景，常用的启发函数对节点的选择并非最优，规划的路径出现非最短、寻路效率低、转向次数多、冗余点多等问题。使用曼哈顿距离度量时，规划的路径虽然比较直，但仅考虑了节点在网格线上的投影，而忽略了实际距离，因此存在一定的误差。而对角线距离则考虑了节点间的实际距离，因

计算机工程 www.ecice06.com

此更加准确。本文将这 2 种距离结合起来，可以取长补短，提高距离计算的准确性。因此对启发函数 $h(n)$ 做以下 2 点改进：

1）优化传统 A* 算法中距离的度量方式。使用曼哈顿距离与对角线距离相结合的方式来表示节点 x 到目标点 g 之间的距离 d_{x_g}，即：

$$\begin{cases} d_{x_g}=\alpha\cdot d_{\text{Manhattan}}+\beta\cdot d_{\text{Diagnol}} \\ \alpha+\beta=1 \end{cases} \tag{5}$$

式中：$d_{\text{Manhattan}}$ 表示当前点到目标点的曼哈顿距离；d_{Diagnol} 表示当前点到目标点的对角线距离；$\alpha=0.4$；$\beta=0.6$。

2）在启发函数前添加一个动态启发式因子 w_{h}，这个因子可以在当前节点远离目标节点时增强启发式函数的作用，从而加快搜索速度。同时，在当前节点靠近目标节点时，这个因子可以削弱启发式函数的作用，从而顺利到达目标点。

$$w_{\text{h}}=1+\frac{d_{c_g}}{l_c\times w_c} \tag{6}$$

$$\begin{cases} l_c=l-|x_c| \\ w_c=w-|y_c| \end{cases} \tag{7}$$

式中：w_{h} 为当前所围的栅格地图与当前节点到目标点之间的距离结合确定的权重因子；d_{c_g} 为当前节点 c 到目标点 g 之间的距离，使用式(5)表示；l_c 为栅格地图的长度 l 与当前节点的横坐标之差；w_c 为栅格地图的宽度 w 与当前节点的纵坐标之差。

结合以上 2 点改进，可将 $h(n)$ 函数设计为：

$$h(n)_{\text{design}}=w_{\text{h}}\times d_{c_g} \tag{8}$$

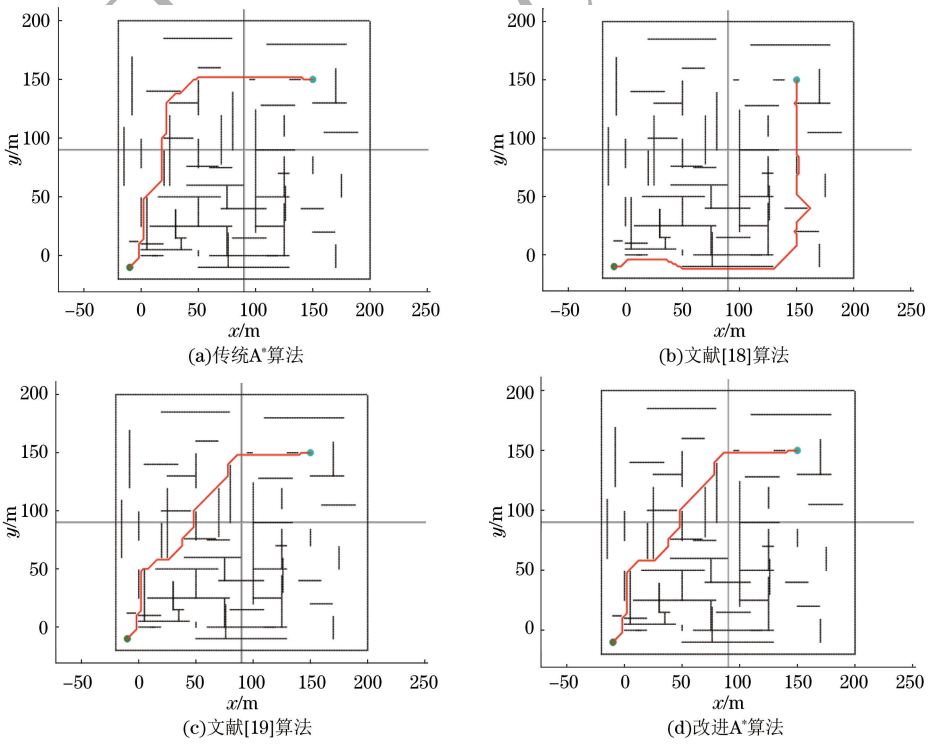
式中：w_{h} 是动态启发式因子；d_{c_g} 为当前节点 c 到目标点 g 之间的距离，使用式(5)表示。

可将改进后的 A* 算法写成：

$$f(n)=g(n)+w_{\text{h}}\cdot h(n)_{\text{design}} \tag{9}$$

通过添加权重因子 w_{h}，能够更好地平衡评估函数在路径搜索过程中的权重，解决原算法在搜索前期因启发函数权重过大出现局部最优解的问题；通过改进的 $h(n)_{\text{design}}$ 函数，能够更加准确地估计每个节点到目标点的距离，减少无用节点的扩展。

为验证改进评估函数的有效性，将传统 A* 算法、文献[18]和文献[19]所提出的算法与本文改进评估函数进行仿真实验。路径规划结果如图 1 所示（彩色效果见《计算机工程》官网 HTML 版，下同），性能对比如表 1 所示。



(a)传统A*算法　(b)文献[18]算法

(c)文献[19]算法　(d)改进A*算法

图 1　4 种算法路径规划结果对比(地图 1)

Fig. 1　Path planning result comparison of four algorithms(map 1)

从图 1 和表 1 结果可以看出，传统 A* 算法和文献[18]算法用时较短、转向次数最少、节点数最少，但路径长度非最短，而文献[19]算法和本文改进 A* 算法都能找到最短路径，但改进 A* 算法在寻路时间、转向次数、节点数方面更优，验证了本文改进启发函数的有效性和可行性。另外，还需要确定改进启发函数中参数 α 和 β 的最优值，在地图 1 上进行仿真实验，具体数据如表 2 所示。

计算机工程 www.ecjournal.com

表 1 4 种算法性能对比(地图 1)

Table 1 Performance comparison of four algorithms(map 1)

算法	路径长度/m	寻路时间/s	转向次数/次	节点数/个
传统 A* 算法	291.28	0.58	13	562
文献[18]算法	329.32	0.92	23	731
文献[19]算法	270.78	3.67	17	1 577
改进 A* 算法	270.78	1.69	15	1 018

表 2 参数 α 和 β 分析表(地图 1)

Table 2 Parameters α and β analysis table(map 1)

(α,β)	路径长度/m	寻路时间/s	转向次数/次	节点数/个
(0.1,0.9)	270.78	2.95	15	1 914
(0.2,0.8)	270.78	2.55	15	1 587
(0.3,0.7)	270.78	2.04	15	1 265
(0.4,0.6)	270.78	1.69	15	1 018
(0.5,0.5)	270.78	1.10	13	770
(0.6,0.4)	275.96	0.71	13	580
(0.7,0.3)	275.96	0.56	13	492
(0.8,0.2)	275.96	0.48	13	413
(0.9,0.1)	275.96	2.90	13	1 914
(0.4,0.8)	275.96	0.61	13	511
(0.4,1.0)	278.58	0.26	22	239
(0.5,0.6)	275.96	0.63	13	524
(0.8,0.6)	278.58	0.47	22	432

从表 2 中得到以下结论：

1)当 $\alpha+\beta>1$ 时，遍历的节点数会变少，寻路效率最高，但规划的路径长度并非最短，不符合全局路径规划的要求。

2)当 α 值逐渐减少、β 值逐渐增加时，能寻得最短路径，但遍历的节点数会变多，寻路时间相对更长；当 α 值逐渐增加、β 值逐渐减少时，遍历的节点数会越来越少，从最短路径变成了非最短。因此，选择 $\alpha<\beta$ 且 $\alpha+\beta=1$ 能获得最短的全局路径。

综上，在保证路径长度最短的前提下，选择寻路时间最少的参数组合，由表 2 可得参数(0.4,0.6)为最优解。

1.3 搜索邻域的改进

目前 A* 算法常用的路径搜索方式分为 8 个邻域、16个邻域和32个邻域。在以16个邻域和32 个邻域的方式搜索时，算法不仅要对当前节点能否通行进行判断，也要对与上一个节点之间的连线进行判断，这一定程度上增加了算法的复杂性，因此本文选择在 8 邻域搜索方向上进行优化和改进。

如图 2 所示，传统 A* 算法在每个父节点的 8 个邻近方向上都进行搜索，势必会造成一些无用节点也参与运算。因此，为避免在一些不相关的节点上花费过多的时间，本文优化思路如下：根据当前节点与目标点之间的连线与正东方向的夹角 φ 分为 6 类，根据不同分类实时动态分配搜索方向，具体分类规则如表 3 所示。

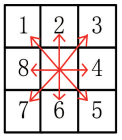


图 2 8 邻域搜索

Fig.2 8-neighbourhood search

表 3 搜索方向分类规则

Table 3 Search direction classification rules

类别	φ	保留的节点方向	删除的节点方向
Ⅰ	(22.5°,90°]	1,2,3,4,5,6	7,8
Ⅱ	[0°,22.5°]∪(337.5°,360°]	2,3,4,5,6	1,7,8
Ⅲ	(90°,157.5°]	1,2,3,6,7,8	4,5
Ⅳ	(157.5°,202.5°]	1,2,6,7,8	3,4,5
Ⅴ	(202.5°,270°]	1,2,5,6,7,8	3,4
Ⅶ	(270°,337.5°]	2,3,4,5,6,7	1,8

计算机工程 www.ecice06.com

在表 3 中，6 种分类都保留了重要的节点方向，同时也舍弃一些不必要的节点方向，算法根据 φ 的值动态调整搜索方向，避免一些无用节点的扩展，有效提高了寻路效率。为验证上述改进的有效性，将改进 A* 算法和传统 A* 算法在地图 1 上进行仿真实验，具体结果如表 4 所示。

表 4 改进 A* 算法和传统 A* 算法的方向搜索性能对比(地图 1)

Table 4 Direction search performance comparison of improved A* algorithm and traditional A* algorithm(map 1)

算法	邻域搜索方式	路径长度/m	寻路时间/s	转向次数/次	节点数/个
传统 A* 算法	8 邻域搜索	291.28	0.58	15	562
	本文改进邻域搜索	291.28	0.36	15	424
改进 A* 算法	8 邻域搜索	270.78	1.69	15	1 018
	本文改进邻域搜索	270.78	0.95	15	929

从表 4 数据可以看出，2 种算法在地图 1 上分别使用了 8 邻域 8 方向搜索和本文改进邻域搜索，虽然在终点路径长度方面一致，但在寻路时间和扩展节点数上，本文改进邻域搜索更占优势，从而验证了本文改进邻域的有效性和可行性。

1.4 基于 Floyd 算法的路径优化

改进 A* 算法是基于栅格地图输出最优路径，但该路径下依然存在着不必要的转折点。基于此，使用 Floyd 算法对改进 A* 算法输出的路径进行优化处理，思路如下：

步骤 1 从起点开始，如果有相邻 3 个以上(含 3 个)的节点共线，且该直线斜率为 0 或者不存在，那么将中间节点去除，只保留每条共线的首尾节点，并将保留的节点依次添加到列表中。

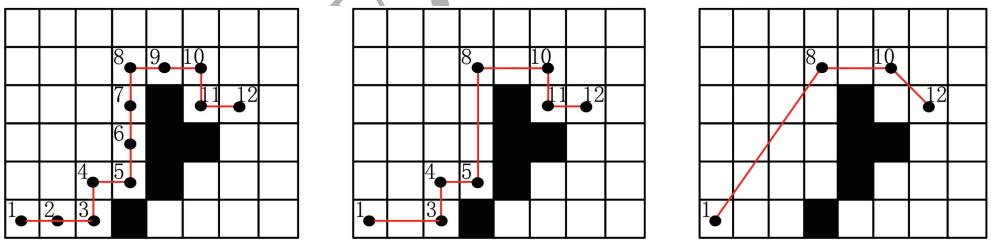
步骤 2 从列表的起点开始，将当前节点 t 与相邻节点 $t+1$ 的连线作为备选路径，如果该备选路径离最近障碍物的距离大于机器人的膨胀半径 R，那么删除节点 $t+1$，由节点 $t+2$ 来代替节点 $t+1$；否则保留该节点，把该节点的上一个节点作为当前节点 t，并将节点 t 添加到新列表中。

重复步骤 2，直至当前节点 t 为最后一个节点。步骤 2 的过程可用式(10)描述说明：

$$l_{\text{list}}=l_{\text{list}}+\{n[k-1]\},\quad d_{i_k}<R,\ k>1\ \text{且}\ k>i \tag{10}$$

式中：l_{list} 为新列表；$n[k-1]$ 表示第 $k-1$ 个节点；d_{i_k} 表示当前节点 i 到第 k 个节点之间的直线距离；R 为机器人的膨胀半径。

如图 3 所示：经过步骤 1，删除节点 2、6、7、9，保留节点 1、3、4、5、8、10、11、12；经过步骤 2，删除节点 3、4、5、11，提取节点 1、8、10、12 作为新的路径节点。



(a)路径优化前 (b)步骤1 (c)步骤2

图 3 路径优化示意图

Fig.3 Schematic diagram of path optimization

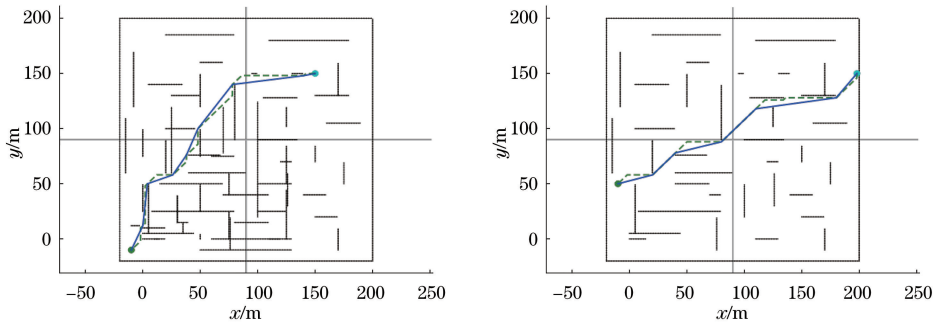
为了验证使用 Floyd 算法去除不必要节点的有效性，在栅格地图上使用改进 A* 算法进行仿真实验，实验结果如图 4 所示，性能对比如表 5 所示。在图 4 中，绿色虚线为原路径，蓝色实线为优化后的路径。从表 5 可知，相比平滑前，平滑后的路径长度平均缩短 4.6%，转向次数减少 54.1%。

表 5 路径平滑前后性能对比(地图 1、地图 2)

Table 5 Performance comparison before and after path smoothing(map 1, map 2)

地图	路径平滑前/后	路径长度/m	转向次数/次
地图 1	路径平滑前	270.78	15
	路径平滑后	256.13	6
地图 2	路径平滑前	251.76	9
	路径平滑后	242.12	5

计算机工程 www.ecice06.com



(a)地图1 (b)地图2

图 4 Floyd 算法去除多余节点实验结果(地图 1、地图 2)

Fig.4 Experimental results of removing redundant nodes by Floyd algorithm(map 1, map 2)

2 动态窗口法

动态窗口法是一种常用的局部路径规划方法，它主要考虑了机器人的运动约束和环境动态信息，能实时地避开障碍物并选择合适的运动方向。如图 5 所示，机器人相邻时刻 Δt 内(ms 级)运动距离极短，可将相邻两点之间的运动轨迹 Δs 看成一条直线，从而得到机器人的运动学方程：

$$\begin{cases} x_t = x_{t-1} + v_{t-1}\Delta t\cos\theta_t \\ y_t = y_{t-1} + v_{t-1}\Delta t\sin\theta_t \\ \theta_t = \theta_{t-1} + w_{t-1}\Delta t \end{cases} \tag{11}$$

式中：x_t、y_t、θ_t 表示机器人在 t 时刻的位姿信息；x_{t-1}、y_{t-1}、θ_{t-1} 表示机器人在 $t-1$ 时刻的位姿信息；v_{t-1}、w_{t-1} 分别表示机器人在 $t-1$ 时刻的 x 轴线速度和 z 轴角速度。动态窗口法对模拟出的机器人的若干运动轨迹进行评估择优，其评估函数为：

$$G(v,w) = \alpha \cdot h(v,w) + \beta \cdot d_{co}(v,w) + \gamma \cdot v_{vel}(v,w) \tag{12}$$

式中：$h(v,w)$、$d_{co}(v,w)$、$v_{vel}(v,w)$ 分别为方位角、障碍物距离、速度评价子函数；α、β、γ 分别为各评价子函数的加权系数。

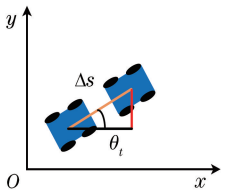


图 5 移动机器人运动模型

Fig.5 Mobile robot kinematics model

3 改进 A* 算法与改进动态窗口法的融合

针对机器人路径规划对于路径最短、搜索效率以及平滑度的性能要求，将改进 A* 算法融合改进动态窗口法，实现动静态导航，获取更优的行驶路径和更高的时间效率。

3.1 融合算法

传统动态窗口法有着不错的动态避障性能，但仍存在以下问题：由于缺乏全局先验信息，导致输出的路径并非最优甚至无法抵达目标点；目标点附近存在障碍物时，规划的路径长度增加或者规划失败。本文改进的 A* 算法能够实时获取全局最优路径信息，但仅适用于静态环境下的避障处理。针对上述 2 种算法存在的问题，综合 2 种算法优缺点，对传统动态窗口法做如下改进：先将改进 A* 算法输出的全局最优路径使用 Floyd 算法提取出关键点，再将关键点融入方位角函数中；然后再加入目标点距离评估子函数 $d_{cg}(v,w)$。

$$G(v,w) = \alpha \cdot o_{oh}(v,w) + \beta \cdot d_{cg}(v,w) + \gamma \cdot d_{co}(v,w) + \delta \cdot v_{vel}(v,w) \tag{13}$$

式中：新方位角 $o_{oh}(v,w)$ 函数为当前预测位姿到最近关键点的方位角偏差；$d_{cg}(v,w)$ 表示目标位姿到起点位姿欧氏距离减去当前预测位姿到目标位姿欧氏距离的绝对值；α、β、γ、δ 为各评价子函数的加权系数。α、β、γ、$\delta \in (0,1)$，且 β 和 γ 满足如下关系：

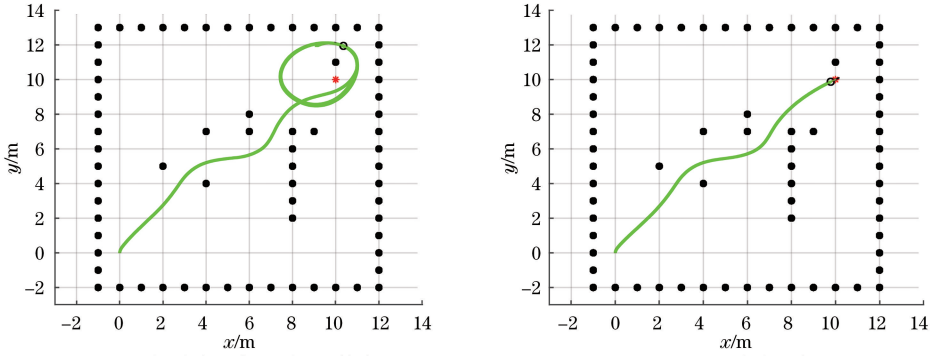
$$\begin{cases} \beta = e^{-kd_{cg}(v,w)} \\ \gamma = 1 - e^{-kd_{cg}(v,w)} \end{cases},\ d_{cg}(v,w) \neq 0; \tag{14}$$

β 随着 $d_{cg}(v,w)$ 增加从趋向 0 变化为趋向 1，而 γ 随着 $d_{cg}(v,w)$ 增加从趋向 1 变化为趋向 0，达到了动态调节 $d_{cg}(v,w)$ 函数与 $d_{co}(v,w)$ 函数权重的目的，解决了原算法在目标点附近存在障碍物时因 $d_{co}(v,w)$ 函数权重过大导致路径规划变长问题。

为了验证引入新距离评估子函数 $d_{cg}(v,w)$ 的可行性和有效性，在栅格地图上使用改进的动态窗口法与原动态窗口法、文献[21]算法进行仿真实验，仿真结果如图 6 所示。

当目标点附近有障碍物时，原动态窗口法和文献[21]算法中 $d_{co}(v,w)$ 函数会更倾向选择远离障碍物的预测轨迹，也即 $d_{co}(v,w)$ 权重过大导致，从而无法直接抵达目标点。相反，改进动态窗口法的

计算机工程 www.ccjc06.com



(a)原动态窗口法或文献[21]算法 (b)改进动态窗口法

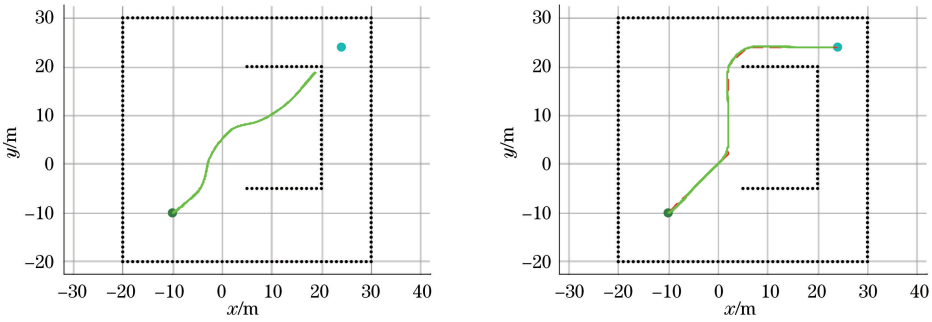
图 6 目标点附近有障碍物情况下的路径规划结果对比

Fig.6 Path planning result comparison with obstacles near the target point

预测轨迹逐渐靠近目标点时，$d_{cg}(v,w)$ 函数的权重会逐渐增加，而 $d_{co}(v,w)$ 函数的权重会逐渐降低，此时削弱了障碍物对预测轨迹的影响，机器人可直接抵达目标点。

为验证新方位角 $o_{oh}(v,w)$ 函数的有效性，将本文提出的融合算法和动态窗口法进行仿真实验，仿真结果如图 7 所示。

从图 7 可以看到，融合算法克服了原动态窗口法遇 U 型障碍物易陷入局部最优的弊端，实现了路径长度、平滑度和安全性的三重优化。



(a)动态窗口法不可达目标点 (b)融合算法可达目标点

图 7 U 型障碍物地图路径规划结果对比

Fig.7 Path planning result comparison in U-shaped obstacle map

3.2 关键点密集化策略

Floyd 算法提取的关键点之间存在距离较长，导致融合算法规划的路径比较偏离全局最优路径的问题。基于此，需要将稀疏的关键路径点转换成更稠密的等距离路径点，本文基于 B 样条曲线[22-24]进行关键点密集化。

假设有 2 个初始关键点 P_1 和 P_2，现在希望在 P_1 和 P_2 之间构造一条 B 样条曲线，使得曲线上的每新增 2 个相邻点之间的距离都等于 d，思路如下：

1）根据关键点 P_1 和 P_2 的位置和形状确定控制点 P_0，P_1，…，P_n。这些控制点之间的连线构成了 B 样条曲线。对于给定的参数 t，B 样条曲线上的点可以表示为：

$$C(t)=\sum_{i=0}^{n}P_i\cdot B_i(t) \tag{15}$$

$$B_i(t)=\frac{t_i-t_{i-1}}{t_i-t_{i-1}}B_{i-1}(t)+\frac{t_{i+1}-t_i}{t_{i+1}-t_i}B_{i+1}(t),\quad t\in[t_{i-1},t_i] \tag{16}$$

式中：$B_{i-1}(t)$ 和 $B_{i+1}(t)$ 是在相邻的 2 个控制点 t_{i-1} 和 t_{i+1} 所确定的参数区间上定义的 B 样条基函数。

2）在 B 样条曲线上计算 m 个等距离的点，取 $m-1$ 个参数值 $t_1<t_2<\cdots<t_{m-1}$，使得相邻 2 个参数值之间对应曲线上的点之间的距离等于 d。根据这些参数值，得到在 B 样条曲线上相应的点，即：

$$C_{new}(t_j)=C(t_{j-1})+d\cdot\frac{C(t_j)-C(t_{j-1})}{d},\quad j=1,2,\cdots,m-1 \tag{17}$$

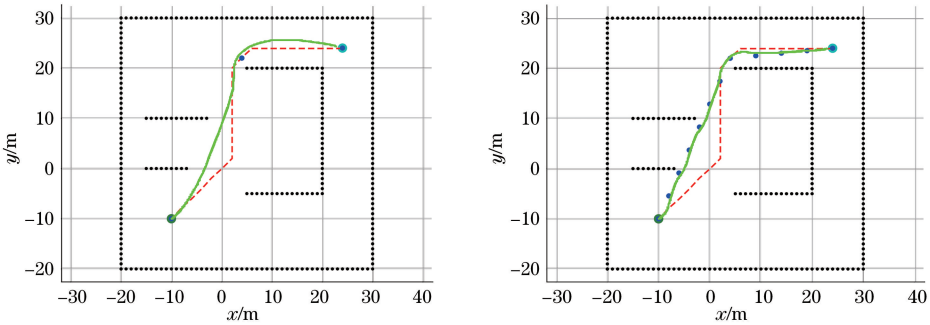
式中：d 是 2 个相邻点之间的距离。

3）在本文中设置 $d=4$ m，关键点 $P_1=(x_1,y_1)$，$P_2=(x_2,y_2)$，m 为 P_1 和 P_2 之间等距离点的个数。对每对相邻的关键点做如下处理：

$$d_{P_1P_2}=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2} \tag{18}$$

$$\begin{cases} m=0, & d_{P_1P_2}<d \\ m=\dfrac{d_{P_1P_2}}{4}+1, & d\leqslant d_{P_1P_2}\leqslant 3d \\ m=\dfrac{d_{P_1P_2}}{4}, & d_{P_1P_2}>3d \end{cases} \tag{19}$$

从式(18)、式(19)可以看出，每 2 个相邻的关键点距离不同，新增点的个数也不同，可以根据 2 个关键点之间的距离有效控制新增等距离点的个数，避免新增点数过多，影响导航效率，同时结合式(17)可得密集化后点的对应坐标。为验证提出的关键点密集化策略的有效性，将密集化前关键点和密集化后关键点分别加入融合算法中，仿真结果如图 8 所示。



(a)密集化前的路径　　(b)密集化后的路径

图 8　关键点密集化前后路径对比

Fig. 8　Path comparison before and after keypoint densification

在图 8 中，虚线为改进 A* 算法规划的全局路径，实线为本文融合算法规划的路径。在图 8(a)中，蓝色点为 Floyd 算法提取的关键点(密集化前的关键点)，融合算法规划的路径长度为 56.45 m；在图 8(b)中，蓝色点为密集化后的关键点，融合算法规划的路径长度为 54.93 m。可以看出，密集化后的关键点作为融合算法的局部导航子目标点，其行驶的路径更贴合全局最优路径，路径长度更短。

3.3　融合算法实现

融合算法实现了动静态路径规划，具体算法实现如图 9 所示。

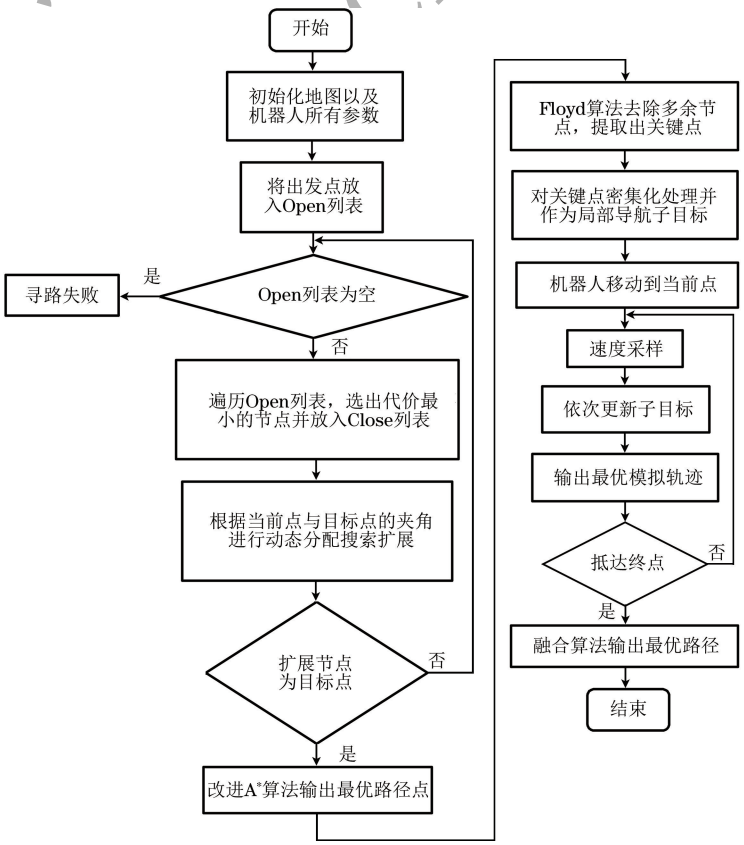


图 9　融合算法控制流程

Fig. 9　Control flow of fusion algorithm

以下是融合算法的伪代码：

算法 改进 A* 算法与改进动态窗口法的融合算法

输入 地图数据，机器人参数，起始点(start)，目标点(goal)

输出 行驶路径点 result_x，行驶速度 u

```
open_set=PriorityQueue()//使用优先队列
open_set.put(start,0)//将起点加入 open_set
close_set={}//节点距起点的代价
came_from={}//节点的父节点
came_from[start]=None
while(true)//改进 A* 算法
{
if(open_set.empty())//寻路失败
break
current=open_set.get()//优先级最高的节点
if(current==goal)//找到终点
{
//返回全局最优路径列表
Pathx, Pathy=final_path(goal,closed_set)
break
}
for next in graph.neighbors(current)//遍历当前节
//点的改进邻域
{
new_cost = close_set[current] + graph.cost
(current,next)//计算 next 代价
if(! close_set.contain(next) or new_cost < close_
set[next])//next 不在 close_set 或有更小代价
{
close_set[next]=new_cost//更新代价
priority=new_cost+ h(n)//计算优先级
open_set.put(next,priority)//加入 open 列表
came_from[next]=current//设置其父节点
}
}
}
Pathx,Pathy=FPath(Pathx, Pathy)//关键点提取
Pathx,Pathy=DPath(Pathx, Pathy)//密集化处理
goal=[Pathx[0],Pathy[0]]//起点
end=[Pathx[len(Pathx)], Pathy[len(Pathy)]]
//终点
while(true)
{
//改进的 DWA 算法，算出速度 u 和轨迹 traj
u,traj=DWA(x,Kinematic,goal,eval,obstacle,R)
x=f(x,u)//计算下一刻的位置、速度、角度
result_x=result_x+x//行驶轨迹的保存
if(distance(x,goal)<=0.5) and (j<len(Pathx))
j = j + 1
goal=[Pathx[j],Pathy[j]]//更新局部目标点
if(distance(goal, end)<=0.5)//到达终点
break
}
```

融合算法的时间复杂度分析如下：

1)改进 A* 算法的时间复杂度可由 $O(b^D)$ 来表示，其中，b 是每个节点的平均子节点数，D 是搜索深度(从起始节点到目标节点所经过的父节点的数量)，算法运行的时间与输入规模 b^D 成正比。

2)改进的动态窗口法时间复杂度可由 $O(n^2)$ 来表示，其中 n 是指优化变量的数量，随着优化变量数量的增加，算法的运行时间将呈二次方增长。本文中 $n=4$，分别为当前机器人位置 (x,y)、航向角、线速度、角速度。

综合上述分析，可得融合算法的时间复杂度为：$O(b^D)+O(n^2)$。

具体而言，改进的 A* 算法输出全局最优路径点，接着使用 Floyd 算法对该路径点提取出关键节点，并对关键节点进行密集化处理，得到局部导航的子目标。融合算法根据这些子目标进行速度采样，输出最优的行驶路径。

4 实验与仿真分析

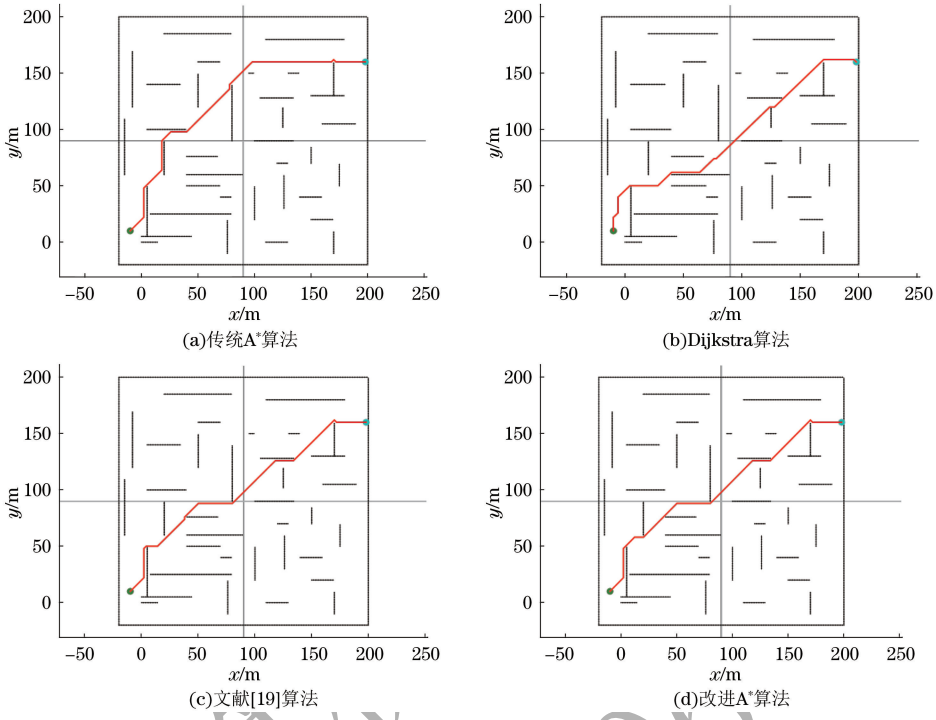
实验验证分为 2 部分：第 1 部分是在静态环境下，使用改进的邻域分别应用于改进 A* 算法、传统 A* 算法、Dijkstra 算法和文献[19]提出的算法，重点评估搜索时间、搜索点数、路径长度、转向次数 4 个方面的性能；第 2 部分是在动态环境下，使用本文改进的融合算法、传统融合算法[13](传统 A* 与传统动态窗口法相融合的算法)和文献[19]、文献[20]提出的融合算法进行对比，重点评估运行时间、路径长度 2 个方面的性能。以下仿真实验均在 Python3.9 环境下实现。

4.1 改进 A* 算法实验结果

为验证改进 A* 算法的有效性，将改进 A* 算法与传统 A* 算法、Dijkstra 算法、文献[19]提出的算法进行仿真实验，对仿真的结果进行可视化对比展示。

从图 10 和图 11 可以看出，改进的 A* 算法规划的路径更直，转向次数最少。由表 6 和表 7 可以看出，改进的 A* 算法在复杂环境下能够寻得最短路径，且寻路时间最短，与另外 3 种算法相比，平均寻路时间缩短 55.66%，平均转向次数减少 16.3%；与文献[19]算法相比，路径长度缩短 0.3%；与传统 A* 算法相比，路径长度缩短 2.5%。综合来看，改进的 A* 算法具有路径长度最短、搜索效率最高、转向次数最少等优势。

计算机工程 www.ecice.com

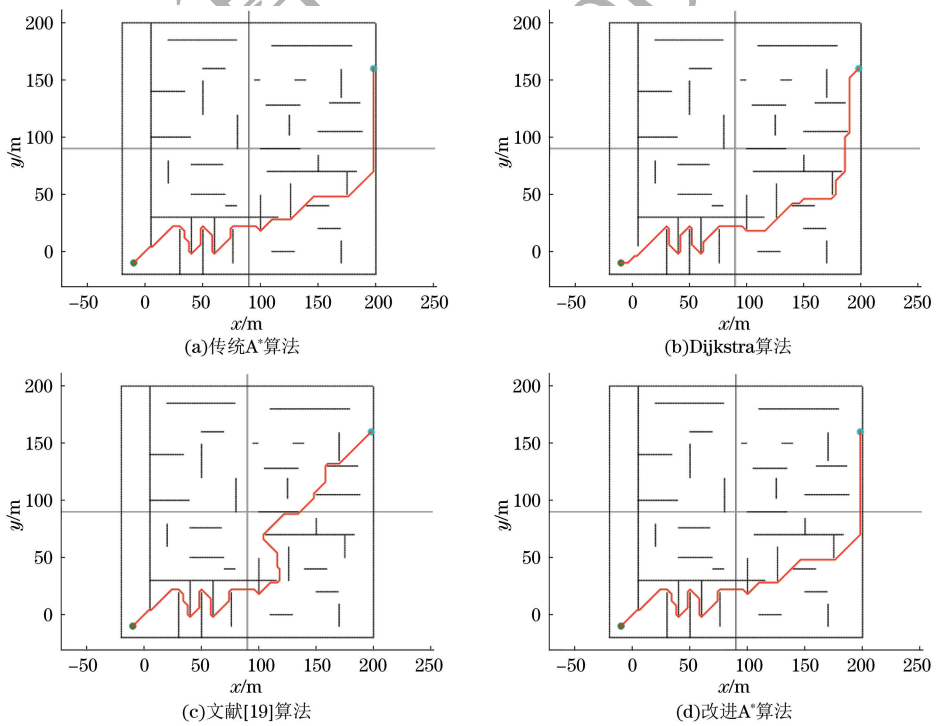


(a)传统A*算法 (b)Dijkstra算法

(c)文献[19]算法 (d)改进A*算法

图 10 4 种算法路径规划结果对比(地图 3)

Fig. 10 Path planning result comparison of four algorithms(map 3)



(a)传统A*算法 (b)Dijkstra算法

(c)文献[19]算法 (d)改进A*算法

图 11 4 种算法路径规划结果对比(地图 4)

Fig. 11 Path planning result comparison of four algorithms(map 4)

表 6 4 种算法性能对比(地图 3)

Table 6 Performance comparison of four algorithms(map 3)

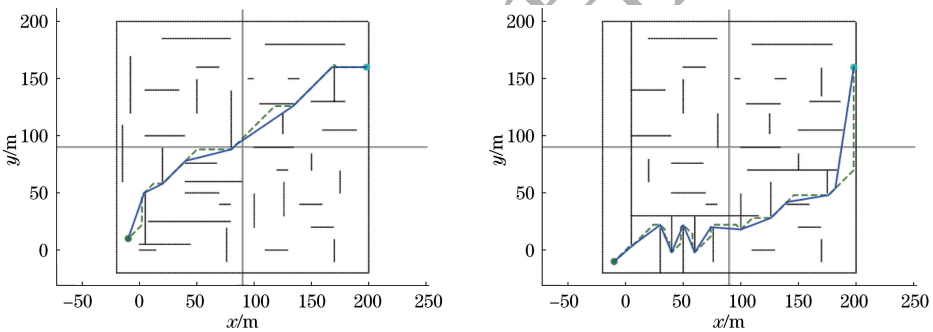
算法	寻路时间/s	搜索点数/个	路径长度/m	转向次数/次
传统 A* 算法	0.36	244	304.58	12
Dijkstra 算法	3.95	10 974	287.01	13
文献[19]算法	2.08	897	288.18	12
改进 A* 算法	0.58	410	287.01	10

表 7 4 种算法性能对比(地图 4)

Table 7 Performance comparison of four algorithms(map 4)

算法	寻路时间/s	搜索点数/个	路径长度/m	转向次数/次
传统 A* 算法	2.63	2 796	403.50	25
Dijkstra 算法	3.17	10 397	403.50	29
文献[19]算法	5.15	3 337	404.54	36
改进 A* 算法	2.13	2 643	403.50	25

为获得更短、更平滑的全局路径,使用本文提出的去除多余节点方法,对改进 A* 算法规划的路径平滑处理,平滑结果如图 12 所示,性能对比如表 8 所示。



(a)地图3 (b)地图4

图 12 Floyd 算法去除多余节点实验结果(地图 3、地图 4)

Fig. 12 Experimental results of removing redundant nodes by Floyd algorithm(map 3, map 4)

表 8 路径平滑前后性能对比(地图 3、地图 4)

Table 8 Performance comparison before and after path smoothing(map 3, map 4)

地图	路径平滑前/后	路径长度/m	转向次数/次
地图 3	路径平滑前	287.01	10
	路径平滑后	273.98	7
地图 4	路径平滑前	403.50	11
	路径平滑后	381.25	5

如图 12 所示,虚线为原路径,实线为平滑后的路径,平滑后的路径明显更优。从表 8 可以看出,相比平滑前,平滑后的路径长度平均缩短 5.2%,转向次数减少 42.8%。因此,平滑后的路径更加适合机器人做底层控制。

4.2 融合算法实验结果

为了验证本文融合算法的有效性,将改进的融合算法与传统融合算法、文献[19]、文献[20]融合算法分别在临时障碍物地图上、临时障碍物+移动障碍物地图上进行仿真实验。实验参数如下:最大线速度为 1.0 m/s,线加速度为 0.2 m/s^2,线速度分辨率为 0.01 m/s,最大角速度为 20.0(°)/s,最大角加速度为 50.0(°)/s^2,角速度分辨率为 1(°)/s,时间分辨率为 0.1 s,预测周期为 5 s。各评价子函数参数为:$\alpha=0.9$,$\beta=0.1$,$\gamma=0.1$,δ 使用式(14)表示。在仿真之前对不同颜色和形状的线型做如下说明,如图 13 所示。

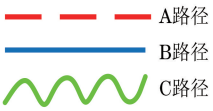


图 13 不同颜色和形状的线型分类

Fig. 13 Classification of threads with different colours and shapes

4.2.1 临时障碍物实验结果

将改进的融合算法与传统融合算法、文献[19]、文献[20]融合算法在有 2 个临时障碍物的地图(地图 5)上进行仿真实验,仿真结果如图 14 所示,性能对比如表 9 所示,机器人控制参数如图 15~图 18 所示。

在地图 5 中,三角形为临时障碍物,A 路径为各算法规划的全局路径,B 路径为各算法去除冗余节点后的路径,C 路径为各融合算法规划的行驶路径。

计算机工程 www.ecice.cn

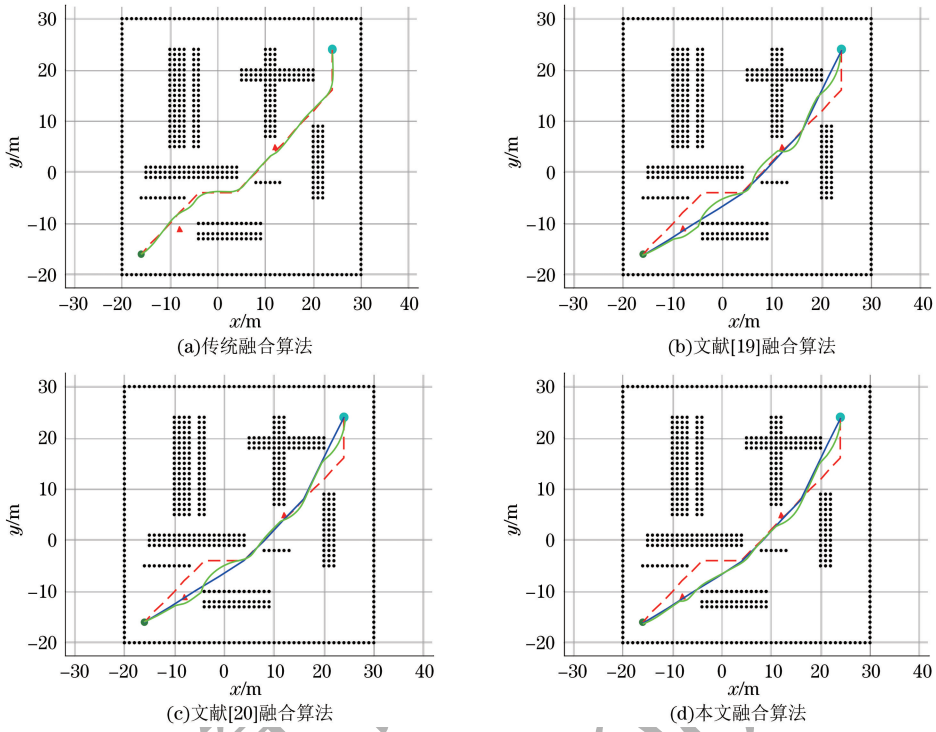


图 14 4 种融合算法路径规划结果对比(地图 5)

Fig. 14 Path planning result comparison of four fusion algorithms(map 5)

表 9 4 种融合算法在临时障碍物环境下性能对比(地图 5)

Table 9 Performance comparison of four fusion algorithms in temporary obstacle environment(map 5)

算法	运行时间/s	路径长度/m
传统融合算法	373.88	61.87
文献[19]融合算法	593.07	62.83
文献[20]融合算法	377.07	60.72
本文融合算法	364.90	58.07

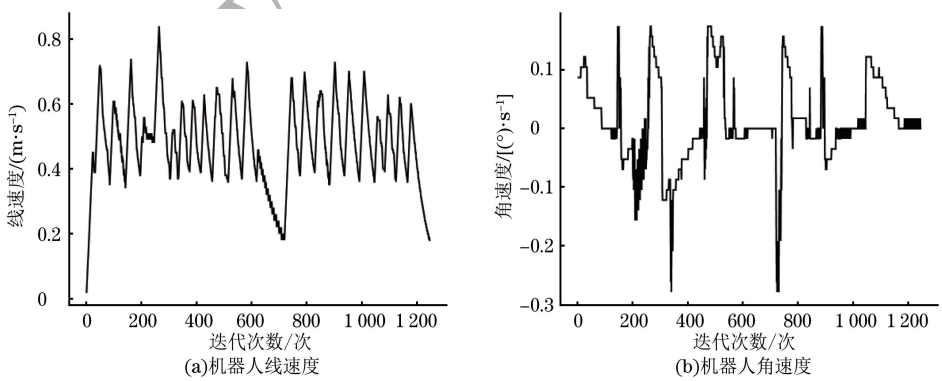


图 15 传统融合算法控制参数(地图 5)

Fig. 15 Traditional fusion algorithm′s control parameters(map 5)

从地图 5 中可以看到,传统融合算法没有去除多余的全局节点,因此融合算法规划的路径最长。文献[19]、文献[20]融合算法和本文融合算法都能成功避开临时障碍物,但文献[19]、文献[20]融合算法没有对关键点密集化处理,导致规划的路径也较长。本文融合算法规划的路径与全局最优路径更加贴合,更适用于机器人做导航控制。从表 9 中可以看出,与另外 3 种融合算法相比,本文融合算法在路径长度和运行时间上更占优势,平均路径长度减少 6.1%,平均运行时间缩短 14.7%,验证了本文融合算法在临时障碍物环境下的有效性和可行性。

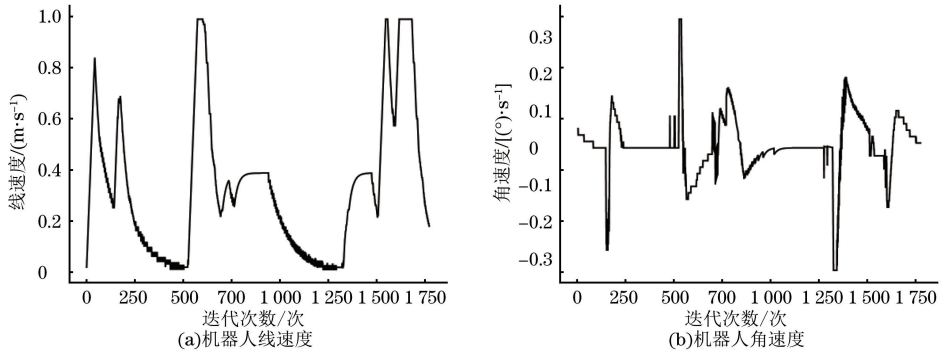


图 16 文献[19]融合算法控制参数(地图 5)

Fig. 16 Literature [19] fusion algorithm's control parameters(map 5)

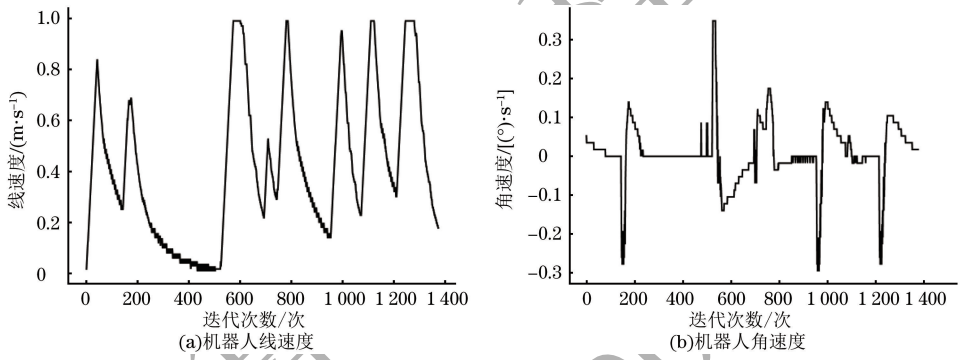


图 17 文献[20]融合算法控制参数(地图 5)

Fig. 17 Literature [20] fusion algorithm control parameters(map 5)

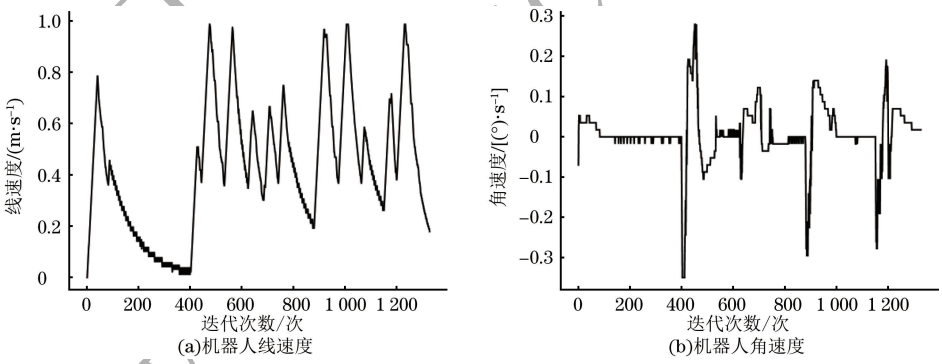


图 18 本文融合算法控制参数(地图 5)

Fig. 18 The proposed fusion algorithm's control parameters(map 5)

4.2.2 临时障碍物和移动障碍物实验结果

将改进的融合算法与文献[19]、文献[20]融合算法在临时障碍物和移动障碍物地图上进行仿真实验,仿真结果如图 19～图 21 所示,性能对比如表 10 所示,机器人控制参数如图 22～图 24 所示。

如地图 6 所示,三角形为临时障碍物,五角星为移动障碍物,规定移动障碍物从点(－10,5)移动到点(2.1,－5.3)后停止移动。黄色路径为移动障碍

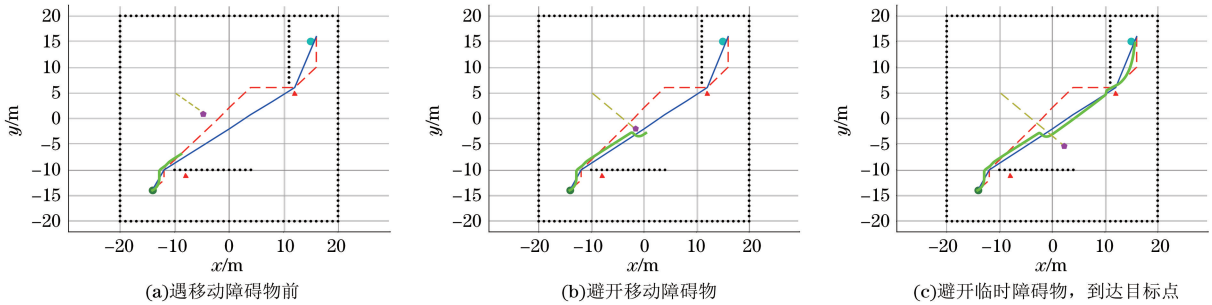
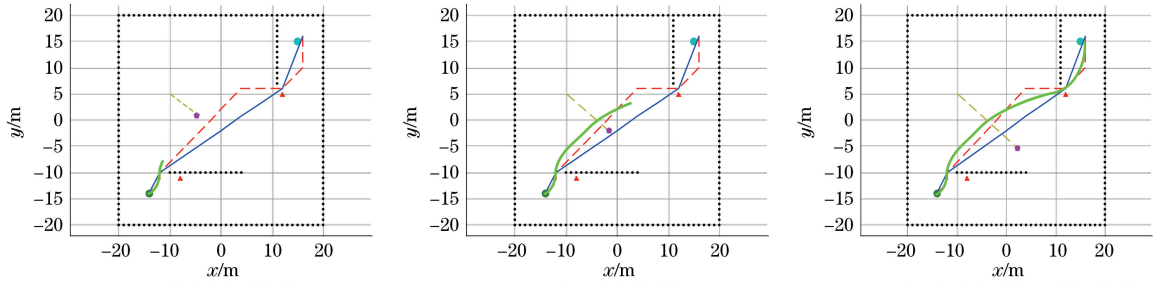


图 19 文献[19]融合算法仿真结果(地图 6)

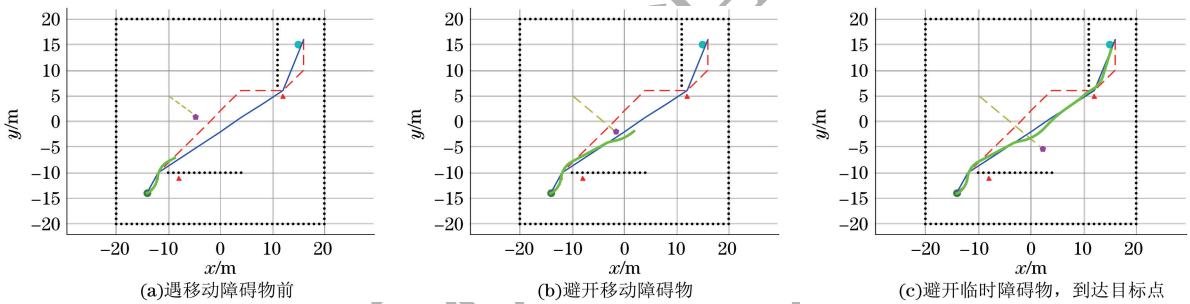
Fig. 19 Simulation results of literature[19] fusion algorithm(map 6)



(a)遇移动障碍物前 (b)避开移动障碍物 (c)避开临时障碍物，到达目标点

图 20 文献[20]融合算法仿真结果(地图 6)

Fig. 20 Simulation results of literature[20] fusion algorithm(map 6)



(a)遇移动障碍物前 (b)避开移动障碍物 (c)避开临时障碍物，到达目标点

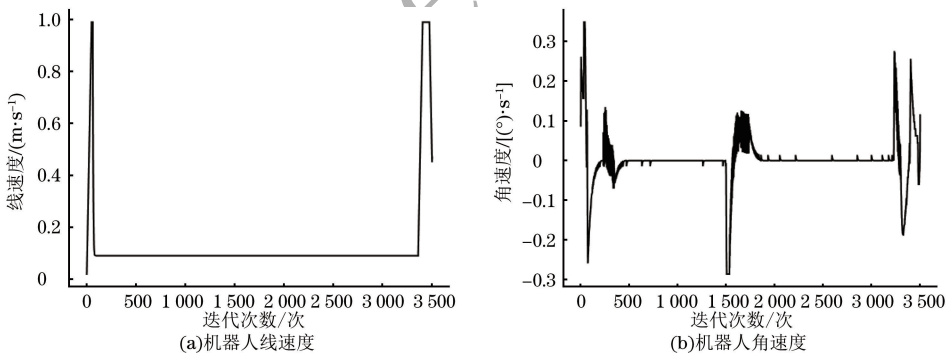
图 21 本文融合算法仿真结果(地图 6)

Fig. 21 Simulation results of the proposed fusion algorithm(map 6)

表 10 3 种融合算法在临时和移动障碍物环境下性能对比(地图 6)

Table 10 Performance comparison of three fusion algorithms in temporary and moving obstacle environments(map 6)

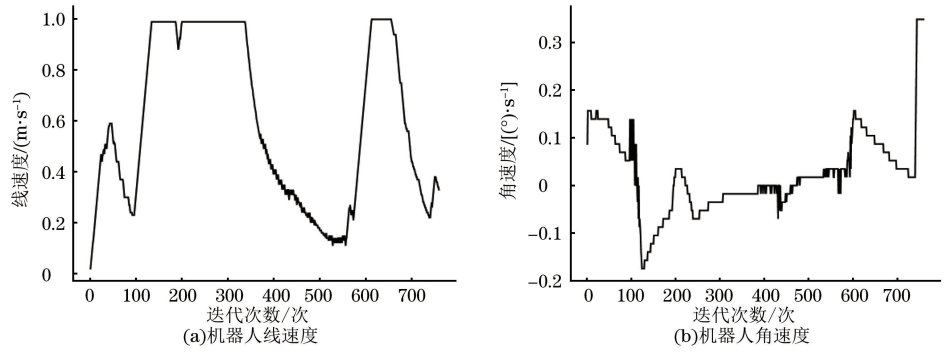
算法	运行时间/s	路径长度/m
文献[19]融合算法	1 802.58	45.07
文献[20]融合算法	149.31	44.60
本文融合算法	168.01	42.78



(a)机器人线速度 (b)机器人角速度

图 22 文献[19]融合算法控制参数(地图 6)

Fig. 22 Literature[19] fusion algorithm's control parameters(map 6)



(a)机器人线速度 (b)机器人角速度

图 23 文献[20]融合算法控制参数(地图 6)

Fig. 23 Literature[20] fusion algorithm's control parameters(map 6)

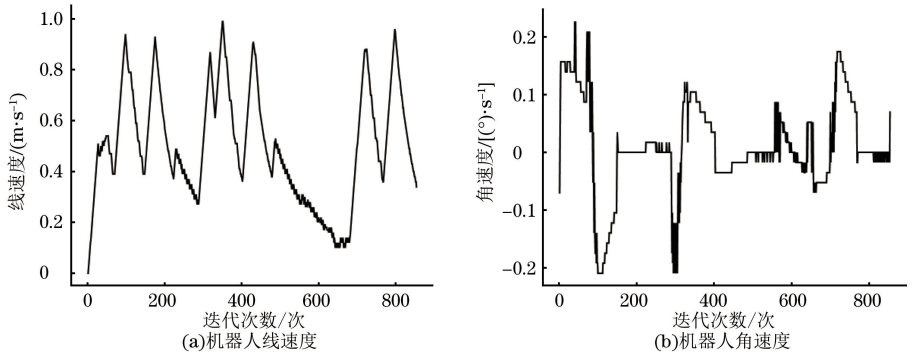


图 24 本文融合算法控制参数(地图 6)

Fig. 24 The proposed fusion algorithm's control parameters(map 6)

物的行驶路径,A 路径为各算法输出的全局路径,B 路径为各算法去除冗余节点后的路径,C 路径是各融合算法规划的行驶路径。

从地图 6 中可以看到,本文融合算法和文献[19]、文献[20]融合算法都能成功避开移动障碍物。文献[20]融合算法因相邻 2 个关键节点之间距离过长,因此规划的路径会偏离全局路径。文献[19]引入了转弯因子,当前子目标点(12,6)与相邻的子目标点(−12,10)、(16,16)距离较远,且转弯因子不为平角,此时机器人速度会非常慢,由图 15(a)也可以看出,机器人的线速度几乎维持在 0.1 m/s,导航效率极低。相比前 2 种融合算法,本文融合算法规划的路径效果最好,更贴合全局最优路径。从表 10 中可以看出:文献[20]融合算法运行时间最短,但路径长度较长;文献[19]融合算法路径规划时间最长,且路径长度也不占优势;相比另外 2 种算法,本文融合算法规划的路径长度最短,平均路径长度和平均运行时间分别缩短 4.6%、39.8%,综合性能比另外 2 种融合算法更优,验证了本文融合算法在移动障碍物环境下的有效性和可行性。

5 结束语

针对机器人在路径规划中的缺陷,提出了一种改进 A* 算法与改进动态窗口法相融合的算法。针对启发函数,结合曼哈顿距离和对角线距离设计了非传统表达的启发函数,并对其动态分配权重,更加准确地估计每个节点到目标节点的距离,避免过多无用节点的扩展,提高算法执行效率,保证全局路径最短。针对传统 8 邻域搜索效率低的问题,提出对当前节点实时动态分配最优搜索方向,以缩短寻路时间。针对改进 A* 算法仍存在无用多余节点的问题,使用 Floyd 算法去除冗余节点,平滑后的路径更适用于底层控制。针对融合算法,将改进 A* 算法输出的全局最优路径使用 Floyd 算法提取出关键点,对关键点密集化处理,将密集后的关键点融入方位角函数中,引入目标距离子函数。为了验证改进的 A* 算法和融合算法的有效性,将本文提出的算法与其他已有的算法进行仿真实验,仿真结果表明:改进的 A* 算法能够在复杂环境下输出全局最短路径,并且具有耗时最少、路径转向次数最少、轨迹平滑等优点;融合算法在动态环境下能够顺利避开临时和移动障碍物,并且行驶路径最短。尽管本文提出的算法在机器人路径规划中取得了一定的成果,但仍存在一些不足之处。例如,针对特定场景的适应性还需要进一步提高,同时算法的实时性也需要进一步优化。未来研究可以针对这些问题进行深入探讨和发展,主要包括以下几点:1)优化算法的效率,采用更高效的搜索算法来提高路径规划的效率,例如使用八叉树、网格简化等技术来缩小搜索空间;2)多机器人协同规划,采用多机器人协同规划的方法,以提高整个机器人系统的性能和效率;3)强化学习等机器学习方法的应用,将强化学习等机器学习方法应用于路径规划中,以实现更加智能和自适应的路径规划。

参考文献

[1] HART P, NILSSON N, 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.

[2] XIANG D, LIN H X, OUYANG J, et al. Combined improved A* and greedy algorithm for path planning of multi-objective mobile robot[J]. Scientific Reports, 2022, 12: 13273.

[3] 詹京昊, 黄宜庆. 融合安全 A* 算法与动态窗口法的机器人路径规划[J]. 计算机工程, 2022, 48(9): 105-112, 120.
ZHANJ W, HUANG Y Q. Path planning of robot combing safety A* algorithm and dynamic window approach[J]. Computer Engineering, 2022, 48(9): 105-112, 120. (in Chinese)

[4] 先梦瑜. 一种基于 Dijkstra 的物流配送路径优化算法设计[J]. 电子设计工程, 2023, 31(2): 20-24.
XIAN M Y. A Dijkstra-based optimization algorithm design for logistics distribution path[J]. Electronic Design

计算机工程 www.ecice06.com

Engineering, 2023, 31(2): 20-24. (in Chinese)
[5] ZHU D D, SUN J Q. A new algorithm based on Dijkstra for vehicle path planning considering intersection attribute[J]. IEEE Access, 2021, 9: 19761-19775.
[6] ZHOU X W, YAN J W, YAN M, et al. Path planning of rail-mounted logistics robots based on the improved dijkstra algorithm[J]. Applied Sciences, 2023, 13(17): 9955.
[7] 王飞鹏,张志安. 基于双向 D* 算法的履带机器人路径规划[J]. 组合机床与自动化加工技术, 2023(5):65-67.
WANG F P, ZHANG Z A. Research on path planning of crawler robot based on bi-directional D* algorithm[J]. Modular Machine Tool & Automatic Manufacturing Technique, 2023(5):65-67. (in Chinese)
[8] FERGUSON D, STENTZ A. Using interpolation to improve path planning: the field D* algorithm[J]. Journal of Field Robotics, 2006, 23(2): 79-101.
[9] 杨桂华,卫嘉乐. 基于改进 A* 与 DWA 算法的物流机器人路径规划[J]. 科学技术与工程, 2022, 22(34): 15213-15220.
YANG G H, WEI J L. Logistics robot path planning based on improved A* and DWA algorithm[J]. Science Technology and Engineering, 2022, 22(34): 15213-15220. (in Chinese)
[10] 迟旭,李花, 费继友. 基于改进 A* 算法与动态窗口法融合的机器人随机避障方法研究[J]. 仪器仪表学报, 2021, 42(3): 132-140.
CHI X, LI H, FEI J Y. Research on robot random obstacle avoidance method based on fusion of improved A* algorithm and dynamic window method[J]. Chinese Journal of Scientific Instrument, 2021, 42(3): 132-140. (in Chinese)
[11] 卞永明,季鹏成, 周怡和, 等. 基于改进型 DWA 的移动机器人避障路径规划[J]. 中国工程机械学报, 2021, 19(1): 44-49.
BIAN Y M, JI P C, ZHOU Y H, et al. Obstacle avoidance path planning of mobile robot based on improved DWA[J]. Chinese Journal of Construction Machinery, 2021, 19(1): 44-49. (in Chinese)
[12] 成昌巍,胡劲文, 王策, 等. 基于改进型人工势场的路径规划方法[J]. 无人系统技术, 2019, 2(6): 10-16.
CHENG C W, HU J W, WANG C, et al. Path planning method based on improved artificial potential field[J]. Unmanned Systems Technology, 2019, 2(6): 10-16. (in Chinese)
[13] FAN X J, GUO Y J, LIU H, et al. Improved artificial potential field method applied for AUV path planning[J]. Mathematical Problems in Engineering, 2020, 2020: 6523158.
[14] ROSTAMI S M H, SANGAIAH A K, WANG J, et al. Obstacle avoidance of mobile robots using modified artificial potential field algorithm[J]. EURASIP Journal on Wireless Communications and Networking, 2019, 2019(1): 70.
[15] TANG G, TANG C Q, CLARAMUNT C, et al. Geometric A-star algorithm: an improved A-star algorithm for AGV path planning in a port environment[J]. IEEE Access, 2021, 9: 59196-59210.
[16] YIN X, CAI P, ZHAO K W, et al. Dynamic path planning of AGV based on kinematical constraint A* algorithm and following DWA fusion algorithms[J]. Sensors, 2023, 23(8): 4102.
[17] LI C G, HUANG X, DING J, et al. Global path planning based on a bidirectional alternating search A* algorithm for mobile robots[J]. Computers & Industrial Engineering, 2022, 168: 108123.
[18] 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.
WEI B W, YAN H. An improved jump point search path-planning algorithm for unstructured environment[J]. Science Technology and Engineering, 2021, 21(6): 2363-2370. (in Chinese)
[19] 齐款款,李二超, 毛玉燕. 改进 A* 算法融合自适应 DWA 的移动机器人动态路径规划[J]. 数据采集与处理, 2023, 38(2): 451-467.
QI K K, LI E C, MAO Y Y. Dynamic path planning of mobile robot based on improved A* algorithm and adaptive DWA[J]. Journal of Data Acquisition and Processing, 2023, 38(2): 451-467. (in Chinese)
[20] 赵久强,赵庭辉, 冯毅萍, 等. 基于改进 A* 算法和 DWA 融合的机器人路径规划研究[J]. 实验技术与管理, 2023, 40(3): 87-92.
ZHAO J Q, ZHAO T H, FENG Y P, et al. Research on robot path planning based on improved A* algorithm and DWA[J]. Experimental Technology and Management, 2023, 40(3): 87-92. (in Chinese)
[21] 程传奇,郝向阳, 李建胜, 等. 融合改进 A* 算法和动态窗口法的全局动态路径规划[J]. 西安交通大学学报, 2017, 51(11): 137-143.
CHENG C Q, HAO X Y, LI J S, et al. Global dynamic path planning based on fusion of improved A* algorithm and dynamic window approach[J]. Journal of Xi'an Jiaotong University, 2017, 51(11): 137-143. (in Chinese)
[22] 孙雷,张丽爽, 周璐, 等. 一种基于 Bezier 曲线的移动机器人轨迹规划新方法[J]. 系统仿真学报, 2018, 30(3): 962-968.
SUN L, ZHANG L S, ZHOU L, et al. A new method on trajectory planning for mobile robots using Bezier curves[J]. Journal of System Simulation, 2018, 30(3): 962-968. (in Chinese)
[23] FENG K, HE X N, WANG M L, et al. Path optimization of agricultural robot based on immune ant colony: B-spline interpolation algorithm[J]. Mathematical Problems in Engineering, 2022, 2022: 2585910.
[24] LI H L, LUO Y T, WU J. Collision-free path planning for intelligent vehicles based on Bézier curve[J]. IEEE Access, 2019, 7: 123334-123340.

编辑 金胡考

计算机工程 www.ecice06.com