文章编号:1671-6833(2026)03-0047-10

基于改进 JPS 和 DWA 算法的移动机器人路径规划

程 博[1], 蔡龙帅[1], 郭桂芳[2], 张 烜[1]

(1. 长安大学 工程机械学院,陕西 西安 710064;2. 西藏民族大学 信息工程学院,陕西 咸阳 712082)

摘 要: 针对传统跳点搜索(JPS)算法在路径规划过程中因访问大量无关扩展节点而致使搜索盲目性增强、内存占用增大,以及输出路径存在冗余节点等问题,提出了一种基于目标点搜索方向优先级和动态权重评价函数的改进 JPS 算法。首先,依据目标点与移动机器人的位置关系,提升算法寻路时目标点所在方向的优先级,并引入基于距离的动态权重评价函数,以此减少搜索无关节点造成的资源浪费和效率损耗;其次,对改进后的 JPS 算法输出的全局路径实施二次规划,消除原路径中存在的冗余节点,使全局路径更为平滑;再次,引入并改进动态窗口法(DWA)算法作为局部路径规划算法,改进后的 DWA 算法将采用基于碰撞距离的动态优先级策略,自动避让交叉路径上的移动机器人;最后,分别对改进后的 JPS 算法和 DWA 算法进行仿真验证。结果表明:相较于传统跳点算法,所提算法搜索到的扩展节点数平均减少了 60.0%,轨迹节点数平均减少了 43.6%,路径拐点数平均减少了 23.9%。此外,改进后的 DWA 算法能够有效解决传统 DWA 算法在处理路径冲突等动态环境问题时存在的缺陷与不足,显著提高了 DWA 算法在多机器人路径规划中的协同性和适应性。

关键词:跳点搜索算法; 路径规划; 优先级避障; 动态权重评价函数; 冲突路径

中图分类号: TP242 **文献标志码**:A **doi**:10.13705/j.issn.1671-6833.2025.03.018

路径规划是移动机器人技术的核心环节,对于实现机器人的高效、安全与智能运行起着至关重要的作用[1]。移动机器人路径规划主要分为全局路径规划与局部路径规划[2]。全局路径规划属于静态规划,它依据全局环境地图中的已知信息进行路径规划,以避开障碍物,常用的算法主要有 A* 算法[3]、RRT 算法[4]、粒子群算法[5]、遗传算法[6]和 JPS(jump point search)算法[7]等;局部路径规划属于动态规划,旨在对行驶环境中可能出现的未知障碍物进行避障,常用的算法主要有人工势力场法[8]、DWA(dynamic window approach)[9]、TEB(timed elastic band)[10]和 MPC(model predictive control)[11]等。

JPS 算法是对 A* 算法的改进与优化。相较于其他算法,该算法在进行路径规划时具有显著优势:一方面,JPS 能够在开阔区域快速找到最短路径,因为它可以跳过众多不必要的节点扩展,这是其他算法难以实现的;另一方面,JPS 算法因其高效性,特别适用于对路径搜索速度有严格要求的应用场景,例如实时或近实时的路径规划问题。尽管 JPS 算法及其变种在上述场景中表现优异,但它主要适用于静态环境中的路径规划,并且依赖于网格结构[12-14]。因此,本文将 JPS 算法与 DWA 算法相融合,即先利用 JPS 算法快速规划出移动机器人的全局路线,再通过融合 DWA 算法,使其能够根据当前行驶的环境状况制定相应的行驶策略。

秦齐等[15]运用了双向 JPS 搜索算法,提高了搜索速度和移动机器人路径规划的导航适应性,但是双向搜索和动态跳点处理都需要额外的计算资源,可能无法满足对实时性的严格要求。Su 等[16]通过引入人工势场计算扩展方向优先级来指导后续跳点搜索,从而提高了寻路效率,但是构建人工势场(artificial potential field,APF)和方向图可能会增加算法的复杂度,特别是在高维度或大规模的环境中。Chen 等[17]将 JPS 与蚁群优化算法(ant colony optimization,ACO)相结合,能够处理复杂环境中的动态障碍物,并能通过蚁群算法自适应调整路径规划,但

收稿日期:2025-10-15;**修订日期**:2025-12-21

基金项目:国家自然科学基金资助项目(12102065);中央引导地方科技发展资金项目(XZ202301YD0003C);西藏民族大学重大项目培育(17MDZP07)

作者简介:程博(1976—),男,陕西大荔人,长安大学副教授,博士,主要从事智能变频控制系统研究,E-mail:2280273795@qq.com。

引用本文:程博,蔡龙帅,郭桂芳,等. 基于改进 JPS 和 DWA 算法的移动机器人路径规划[J]. 郑州大学学报(工学版),2026,47(3):47-56. (CHENG B, CAI L S, GUO G F, et al. Path planning for mobile robots based on improved JPS and DWA algorithms[J]. Journal of Zhengzhou University (Engineering Science),2026,47(3):47-56.)

蚁群算法的性能对参数设置较为敏感,需要精细调整以获得最佳效果。张庆等[18]通过使用切比雪夫距离替代欧氏距离来优化启发函数,使得启发式函数精确等于实际最佳路径,从而减少了节点的扩展数量,但是文献中没有考虑到路径平滑性的问题,特别是在机器人运动特性要求下,路径的平滑性对于实际应用至关重要。DWA 算法在静态环境具有良好避障能力,但在动态环境中表现欠佳。Kobayashi 等[19]对 DWA 算法进行了更深层次的扩展,他们利用激光测距仪和 RGB-D 相机实时获取环境信息,并通过估算盲区和改进的代价函数来优化局部路径规划安全性和最优性,但是在动态环境中,物体的运动可能导致盲区的变化,增加了预测和规避的难度。刘宙浩等[20]基于障碍物距离移动机器人的位置,在评价函数中加入了动态权重系数,提升了算法路径规划时的适应性,但在遇到移动障碍物时可能存在较大的偏移角度和较长的避障路径等问题。Berti 等[21]在算法中加入 Lyapunov 稳定性判据,能使移动机器人以稳定的速度输出控制指令,保证了移动机器人运动控制的安全性和稳定性,但也带来了一定的计算开销,特别是在实时性要求较高的系统中,可能会导致性能下降。王超等[22]对移动障碍物的预测轨迹采用两直线相交法,保证了移动机器人的安全性,但该算法在密集障碍物区域可能无法选择最佳路径,导致避障效果不佳。

因此,本文针对传统 JPS 算法在搜索时无法有效解决计算成本和内存消耗急剧上升等问题,改进了算法的搜索方向和核心评价函数,在保证算法高效性的同时,减少了不必要扩展节点的搜索;其次对全局路径进行二次规划,消除了原有路径中存在的冗余节点。此外,针对 DWA 算法在相遇其他移动机器人时适应性和灵活性较差等问题,提出了一种基于碰撞距离的优先级动态避让策略。最终实现改进 JPS 算法快速高效地规划出移动机器人全局路线,改进 DWA 算法根据全局路径的具体环境状况做出相应的避障策略。

1 算法基础及预处理

1.1 地图建立及预处理

传统 JPS 算法是将全局地图简化成栅格地图,如图 1 所示。为了简单实现环境建模,对于形状不规则的障碍物,规定只要涉及障碍物的栅格区域,使其占满一个完整的栅格并以黑色标识。

鉴于移动机器人的自身尺寸,为避免移动机器人在拐角处与障碍物发生碰撞,根据栅格地图中障碍物的密集程度对障碍物的最外层进行膨胀处理,可避免将跳点搜索到危险区域以确保移动机器人的安全行驶[23]。

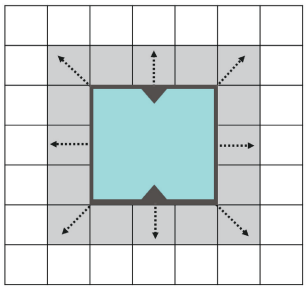


图 1 栅格地图

Figure 1 Grid map

对障碍物进行膨胀处理所依据的评价函数为

$$a=\begin{cases}\lceil kL\rceil, 0\leqslant m<0.7;\\ 0, 0.7\leqslant m<1。\end{cases} \tag{1}$$

式中:a 表示障碍物的膨胀栅格数;m 表示地图中障碍物的密集程度;k 表示移动机器人安全间隔系数;L 表示移动机器人自身的最大空间尺寸,m;⌈⌉为向上取整符号。当 $0.7\leqslant m<1$ 时,高密度障碍物地图已不适合障碍物膨胀和大尺寸机器人行驶,所以不对障碍物进行膨胀处理操作。

1.2 JPS 算法介绍及预处理

JPS 算法是一种启发式搜索算法,通过评价函数(式(2))来确定地图网格中节点的选取[3]。

$$f(n)=g(n)+h(n)。 \tag{2}$$

式中:n 为当前节点;$g(n)$ 为起始节点到当前节点 n 的实际代价;$h(n)$ 为当前节点 n 到目标节点的估计代价;$f(n)$ 为起始节点到当前节点 n 的总代价。其中 $h(n)$ 具有多种计算方式,如图 2 所示。

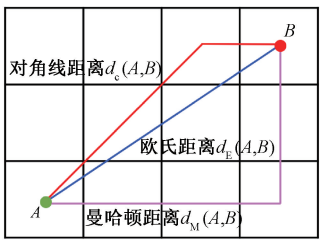


图 2 $h(n)$ 计算方式

Figure 2 $h(n)$ calculation

本文选用更加接近真实距离的对角线距离作为 $h(n)$ 的计算方式[24],计算过程为

$$L_1=|x_{now}-x_{goal}|+|y_{now}-y_{goal}|; \tag{3}$$

$$L_2=\min(|x_{now}-x_{goal}|,|y_{now}-y_{goal}|); \tag{4}$$

$$h(n)=d_c(A,B)=L_1-(2-\sqrt{2})L_2。 \tag{5}$$

式中:(x_{now},y_{now}) 为当前节点的坐标;(x_{goal},y_{goal}) 为目标节点的坐标。

JPS 的高效性得益于其独特的自然领域节点裁

剪和跳跃规则。自然邻域节点裁剪可分为以下两种情况。当父节点 n_p 的自然邻域不存在障碍物时,修剪规则定义为从父节点 n_p 出发,经过某一个自然邻域节点 n_{nature} 到目标节点 n_{goal} 的代价距离小于等于父节点 n_p 经过除去节点 n_{nature} 外的其余节点到目标节点 n_{goal} 的代价距离,裁剪后保留的自然邻域节点 n_{nature} 约束公式[6]表示为

$$\text{length}(\{n_p,n_1,n_2,\cdots,n_{goal}\}\backslash n_{nature}) \geqslant \text{length}(\{n_p,n_{nature},n_{goal}\})。\tag{6}$$

式中:$\{n_p,n_1,n_2,\cdots,n_{goal}\}$ 表示父节点 n_p 到目标节点 n_{goal} 的一系列路径点的集合;\为集合差符号。

当父节点 n_p 的自然邻域周围存在障碍物时,修剪规则将会选取具有强迫邻居的自然邻域节点 n_{nature} 作为待定强制节点,其中强迫邻居节点 n_{forced} 的特征为从父节点 n_p 出发,经过此自然邻域节点 n_{nature} 到强迫邻居节点 n_{forced} 的距离代价比任何其他不经过 n_{nature} 到达强迫邻居节点 n_{forced} 的距离代价都小。强迫邻居节点 n_{forced} 的约束公式[6]表示为

$$\text{length}(\{n_p,n_1,n_2,\cdots,n_{forced}\}\backslash n_{nature}) \geqslant \text{length}(\{n_p,n_{nature},n_{forced}\})。\tag{7}$$

式中:$\{n_p,n_1,n_2,\cdots,n_{forced}\}$ 表示父节点 n_p 到强迫邻居节点 n_{forced} 的一系列路径点的集合。

跳跃规则是 JPS 算法通过迭代当前节点的继承跳点,并通过评价函数选取继承跳点中 f 估计值最小的跳点再进行拓展,直至到达目标点。

2　改进 JPS 算法

2.1　搜索方向的改进

传统 JPS 算法在初始搜索阶段时未确定拓展方向。如图 3(a)所示,由于起点 n_{start} 不存在父节点,起点 n_{start} 开始向周围所有方向扩展,对于目标节点 n_{goal} 来说,起点 n_{start} 的右上方向是最优搜索方向,右方和上方是高效搜索方向,其他方向搜索均为低效搜索方向,这些方向上搜索到的新跳点在增加系统计算资源和内存储存资源的消耗,同时也会增加冗余跳点的数量,降低算法寻优效率。同样,在中间跳点的强迫邻居方向搜索时也会产生相同的问题,如图 3(b)所示。并且随着拓展方向的增加,算法的运行速度也会受到影响而变慢。

为解决上述问题,本文对 JPS 跳点搜索方向进行改进,提出了一种具有优先搜索方向的改进 JPS 算法。具体算法步骤如下。

步骤 1　算法在起点 n_{start} 处搜索跳点时先根据起点与目标点的位置计算目标方向 *Goal_dir*,并将

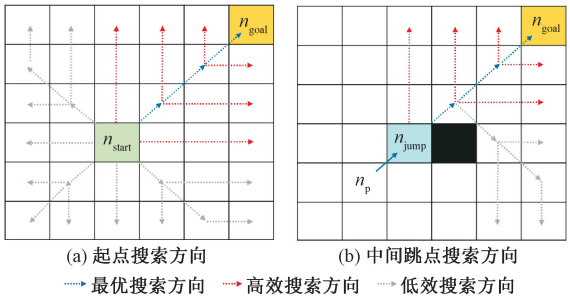


图 3　传统 JPS 搜索方向示意图

Figure 3　Traditional JPS search direction diagram

此方向设为优先搜索方向。为了保证算法的全局最优性,避免算法在搜索过程中陷入局部最优,将与目标方向相邻的两个方向一并设为优先搜索方向,如图 4(a)所示。

步骤 2　当算法在中间跳点 n_{jump} 处进行搜索时,将目标方向及两个相邻方向设为优先搜索方向,并降低跳点 n_{jump} 强迫邻居搜索方向 *Neighbor_dir* 的优先级,规定只有当目标方向未搜索到新的跳点时,算法开始从跳点的强迫邻居方向搜索新的跳点,如图 4(b)所示。

步骤 3　为了保证算法的简洁性和鲁棒性,当目标点方向位于起点的垂直向上或水平向右方向时,默认起点的优先搜索方向为右上角。以此类似,当目标点方向位于起点的垂直向下或水平向左方向时,默认起点的优先搜索方向为左下角。

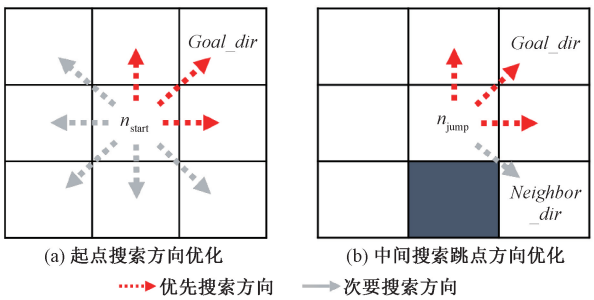


图 4　改进 JPS 搜索方向示意图

Figure 4　Improved JPS search direction diagram

2.2　自适应权重评价函数

JPS 算法路径规划的核心在于评价函数(式(2)),传统 JPS 算法由于评价函数的单一性存在诸多问题,例如出现重复访问不必要跳点、扩展节点搜索范围变大、冗余跳点数目增加的现象。因此,本文引入了移动机器人位置因子对启发函数进行动态加权,通过动态调整启发函数的权重系数来减少无关拓展节点和冗余跳跃节点的搜索。改进后的启发函数为

$$f(n)=k_h h(n)+k_g g(n)。\tag{8}$$

$$\begin{cases} P = l/l'; \\ k_h = 1 + 2P/(P+1); \\ k_g = 2 - P/(P+1)。 \end{cases} \tag{9}$$

式中:k_h 为估计代价 $h(n)$ 的动态权重系数;k_g 为实际代价 $g(n)$ 的动态权重系数;l 为当前跳点到目标点的欧氏距离,m;l' 为起点到目标点的欧氏距离,m。将 k_h 和 k_g 分别取 P 的比值形式的原因是为了缓和当前跳点与父节点相距较远时 P 值激变所带来的影响。

通过引入自适应权重系数后,评价函数会根据当前跳点的位置而动态地改变估计代价 $h(n)$ 和实际代价 $g(n)$ 的占比。当跳点位置距离目标点较远时,算法将提高估计代价 $h(n)$ 的占比,更加关注于快速接近目标点,减少冗余节点的搜索,降低了资源占用对算法搜索效率的影响。当跳点位置接近目标点时,算法将提高实际代价 $g(n)$ 的占比,更关注全局信息以寻找最优路径,提高路径规划的安全性和可靠性。

2.3 冗余节点的删除

针对传统 JPS 算法规划出的全局路径中存在冗余节点导致路径转折点增多、平滑性降低的现象,本文对改进的 JPS 算法规划后的全局路径进行二次处理,提出了冗余节点裁剪算法,即通过判断节点之间的连线是否存在障碍物以去除中间冗余节点。如图 5 所示,当节点 n_1 与节点 n_3 之间不存在障碍物时,算法将以节点 n_1 为基点,节点 n_3 为起点,判断后续节点与节点 n_1 之间是否存在障碍物,直至算法递进到节点 n_5 发现障碍物后,算法将停止递进并裁剪冗余节点 n_3,保留后续关键节点 n_4。随后算法将以节点 n_4 为基点继续递进裁剪。冗余节点裁剪算法具体步骤如下。

步骤 1 输入 3 个相邻的初始路径点 n_i,n_{i+1},n_{i+2}。

步骤 2 判断节点 n_i 和节点 n_{i+2} 之间是否存在障碍物,是则执行步骤 3,否则执行步骤 4。

步骤 3 将节点 n_i 和节点 n_{i+1} 添加到新路径中并更新递进节点位置到 i+1 处,执行步骤 5。

步骤 4 以节点 n_i 为基点,节点 n_{i+2} 为起点开始向后递进搜索,直到 n_i 和 n_{i+m} 之间存在障碍物,将 n_i 和 n_{i+m-1} 添加到新路径中并更新递进节点位置到 i+m−1,执行步骤 5。

步骤 5 判断当前节点位置是否为目标点或目标点前一路径点,若存在,则输出裁剪后的路径;否则转到步骤 2。

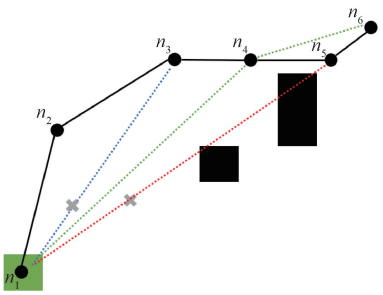


图 5 冗余节点裁剪示例图

Figure 5 Example of redundant node trimming

为验证本文提出的冗余节点裁剪算法的有效性,利用 MATLAB 2022a 搭建模拟仿真环境,构建 100×100 像素的街道栅格地图以及 100×100 像素的仓库栅格地图,分别对改进后的 JPS 算法进行仿真验证。在确保各项仿真条件相同的情况下,仿真结果如图 6 所示,路径数据如表 1 所示。

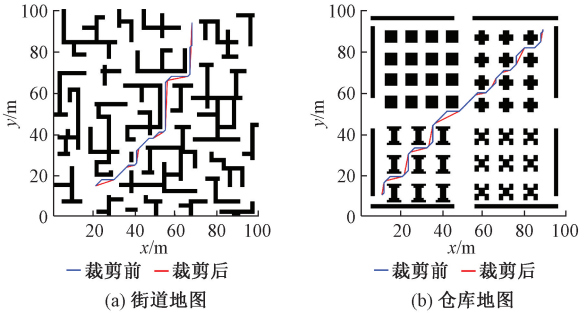


图 6 地图路径对比

Figure 6 Comparison of map paths

表 1 裁剪前后结果对照

Table 1 Comparison of results before and after cutting

地图	路径长度		节点数	
	优化前	优化后	裁剪前	裁剪后
街道	110.77	107.55	24	16
仓库	131.30	125.46	41	25

由图 6 和表 1 可知,应用冗余节点裁剪算法后,在街道栅格地图中,裁剪后的路径长度比裁剪前的路径长度缩短了 2.9%,节点数减少了 33.3%;在仓库栅格地图中,裁剪后的路径长度比裁剪前的路径长度缩短了 4.4%,节点数减少了 39.0%。分析可知,本文设计的冗余节点裁剪算法能够有效去除路径中存在的冗余节点,提升路径规划的平滑性和机器人路径跟踪的安全性。

3 融合 JPS 算法和 DWA 算法

3.1 DWA 算法

DWA 算法是比较常用的局部路径规划算法,其主要步骤是移动机器人首先根据当前运动状态计算出当前的速度和角速度范围,此范围就是动态窗口。

随后在此窗口中计算并模拟出以不同速度和角速度在未来一定时间内的行程轨迹,最后通过评价函数选出评分最高的最优轨迹,重复上述过程直至到达目标点。其运动模型[7]可表示为

$$\begin{cases} x_t = x_{t-1} + v\Delta t\cos\theta_{t-1}; \\ y_t = y_{t-1} + v\Delta t\sin\theta_{t-1}; \\ \theta_t = \theta_{t-1} + \omega\Delta t。 \end{cases} \tag{10}$$

为了保证移动机器人安全行驶,对其速度和角速度的采样范围进行约束,约束公式[7]为

$$V_m = \{(v,\omega) \mid v \in [v_{min}, v_{max}], \omega \in [\omega_{min}, \omega_{max}]\}。 \tag{11}$$

式中:v_{min}、v_{max} 分别为最小、最大线速度;ω_{min}、ω_{max} 分别为最小、最大角速度。

移动机器人的线加速度和角加速度受自身电机性能的限制,在一个速度采样周期内,移动机器人实际能到达的速度为[7]

$$V_d = \{(v,\omega) \mid v \in [v_c - \dot{v}_a\Delta t, v_c + \dot{v}_b\Delta t], \omega \in [\omega_c - \dot{\omega}_a\Delta t, \omega_c + \dot{\omega}_b\Delta t]\}。 \tag{12}$$

式中:v_c、ω_c 表示当前时刻的线速度与角速度;$\dot{v}_a$、$\dot{v}_b$ 表示在电机影响下线速度所能达到的最大减速度与最大加速度;$\dot{\omega}_a$、$\dot{\omega}_b$ 表示在电机影响下角速度所能达到的最大减速度与最大加速度。

为了避免移动机器人因速度过快与障碍物发生碰撞,对移动机器人安全制动范围进行约束[7],即

$$V_a = \{(v,\omega) \mid v < \sqrt{2\mathrm{dist}(v,\omega)\cdot\dot{v}_a}, \omega < \sqrt{2\mathrm{dist}(v,\omega)\cdot\dot{\omega}_a}\}。 \tag{13}$$

式中:$\mathrm{dist}(v,\omega)$ 表示速度空间对应的轨迹离障碍物最近的距离。

综上所述,移动机器人最终动态窗口的移动速度 V_r 取值范围为

$$V_r = V_m \cap V_d \cap V_a。 \tag{14}$$

轨迹空间中有多条基于不同速度和角速度形成的预测轨迹,为了选出最优轨迹,需要引入评价函数对其进行打分,评价函数[7]为

$$G(v,\omega) = \sigma(\alpha\cdot\mathrm{Heading}(v,\omega) + \beta\cdot\mathrm{dist}(v,\omega) + \gamma\cdot\mathrm{velocity}(v,\omega))。 \tag{15}$$

式中:$\mathrm{Heading}(v,\omega)$ 为轨迹切线方向与目标点的夹角;$\mathrm{dist}(v,\omega)$ 为机器人与最近障碍物间的距离;$\mathrm{velocity}(v,\omega)$ 为当前轨迹速度大小;α、β、γ 为各项的加权系数。

3.2　DWA 算法的改进

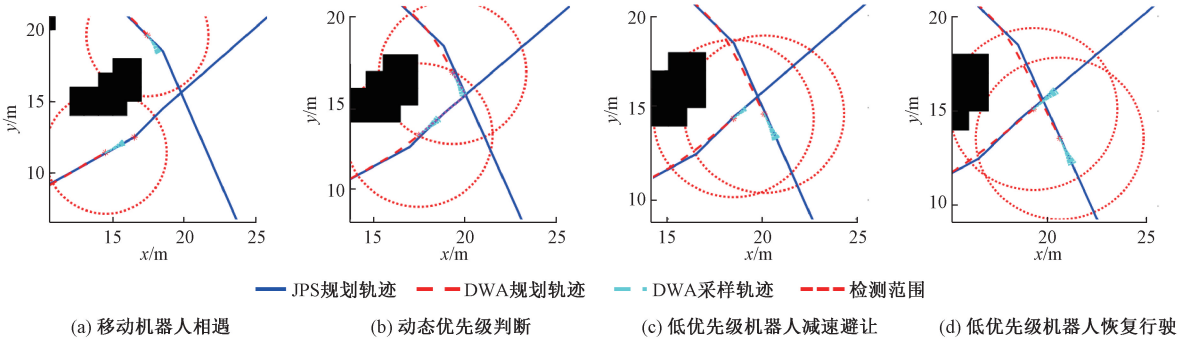
在现实场景中的多移动机器人路径规划中往往存在交叉路径,在运用传统的 DWA 算法对多移动机器人进行路径规划时可能会因算法本身缺乏多机器人间的互动协调机制以及应对动态环境的有效策略而产生一定的规划问题,如 DWA 在动态避让其他机器人时可能选择绕行较长的路径,增加了路径规划时间和能量上的消耗,在存在交叉路径时也有可能会因避让不及时而产生碰撞的风险。为了解决此类问题,本文引入优先级避让策略以提高 DWA 算法的安全性和灵活性。

鉴于动态优先级的优越性和广泛应用性,本文引入基于碰撞距离的动态优先级避让策略,其避让策略具体步骤如下。

步骤 1　移动机器人在进入到彼此障碍物检测范围后(可根据刹车距离自定义检测范围)先进行相互识别,如图 7(a)所示。在识别完成后开始计算各自当前位置到达轨迹交点处的距离,并选取此距离作为后续优先级判断依据,如图 7(b)所示。

步骤 2　距离轨迹交点越近的移动机器人优先级别越高,当它与其他移动机器人相遇时,将会忽视比其优先级低的机器人的位置冲突并继续行驶;相反距离轨迹交点越远的机器人的优先级越低,低优先级的移动机器人会将高优先级的移动机器人实时位置和其采样范围内每一个行程预测轨迹的终点位置视为障碍物进行处理,以实现主动避让优先级高的移动机器人的目的,如图 7(c)所示。

步骤 3　待高优先级的移动机器人通过后,低优先级的机器人开始加速通行直到恢复正常行驶,如图 7(d)所示。



(a) 移动机器人相遇　(b) 动态优先级判断　(c) 低优先级机器人减速避让　(d) 低优先级机器人恢复行驶

图 7　动态优先级避让

Figure 7　Dynamic priority avoidance

3.3 融合JPS算法和DWA算法

鉴于改进的JPS算法在非网格的连续空间以及动态环境中对障碍物或机器人缺乏有效的应对措施,而改进的DWA算法又容易陷入局部最优,因此,本文将改进的JPS算法与改进的DWA算法进行融合。此举旨在确保移动机器人在沿着全局路径行驶的同时,还能在动态环境中顺利且安全地抵达设定的目标点。融合算法的流程图如图8所示。

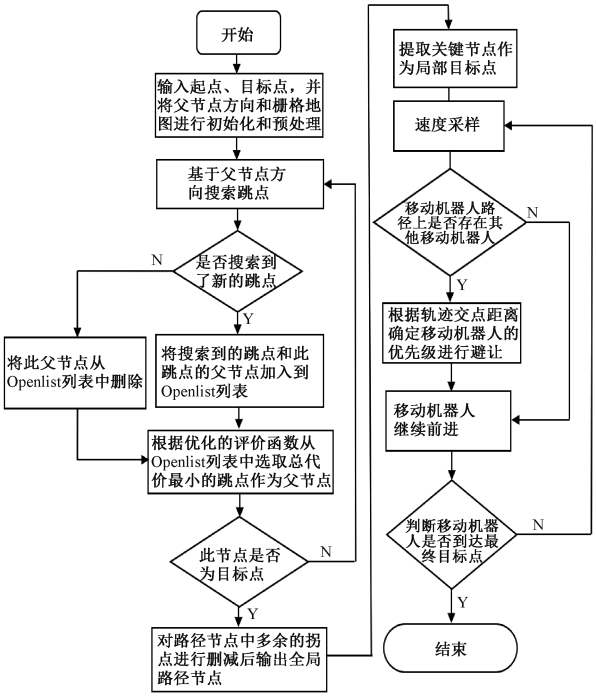


图8 算法融合流程图

Figure 8 Algorithm fusion flowchart

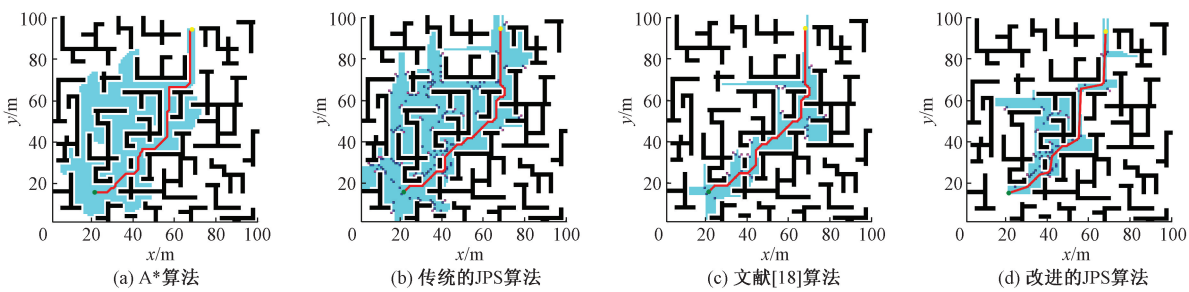
4 仿真分析

为验证本文算法的有效性,利用MATLAB 2022a软件分别对改进的JPS算法和改进的DWA算法进行仿真验证。首先对于改进的JPS算法,在3种不同复杂程度的障碍物环境下对A*算法、传统JPS算法、文献[18]算法和改进的JPS算法进行仿真对比实验。实验地图分别为100×100像素的街道地图、100×100像素的仓库地图、100×100像素的商场地图,各地图障碍物膨胀栅格数为1,分别如图9~11所示,并选取各算法的运行搜索时间、路径长度、扩展节点数、节点数、轨迹节点数和路径拐点个数作为实验结果性能指标,各地图的算法性能指标如表2所示。使用的操作系统为Windows11,处理器为Intel Core i7-14650HX 2.20 GHz。

图9~11中,黑色栅格代表障碍物,绿色圆圈代表起点,黄色圆圈代表目标点,浅蓝色栅格代表各算法扩展的节点数,深蓝色方形代表改进的JPS算法的跳点,玫红色方形代表改进的JPS算法跳点的强迫邻居,其余栅格均代表可行区域;最后实线表示算法规划出的全局路径。表2中,节点数表示算法将访问过的栅格数筛选后收录到open列表中的栅格数。对于A*算法而言,因为没有定义特殊的筛选规则,所以A*算法的扩展节点数就等于节点数。

由表2可知,在地图的总体搜索规模上,改进的JPS算法和文献[18]算法在3个场景中扩展的节点数均显著减少,其中改进的JPS算法在街道和商场场景地图中扩展的节点数比文献[18]算法更少。在复杂街道场景中,改进的JPS算法相较于A*算法、传统的JPS算法和文献[18]算法分别减少了56.46%,64.70%和6.29%。在商场场景中,改进的JPS算法相较于A*、传统的JPS算法和文献[18]算法分别减少了55.39%,58.28%和1.70%。在仓库场景地图中,文献[18]算法扩展的节点数最少,但其规划出的路径长度相较于改进的JPS算法却增加了21.99%,具体的原因可能是算法在追求效率提升的过程中,忽略了一些潜在的更优路径,而这些路径恰恰需要通过搜索那些未被扩展的节点来实现。

在算法的轨迹节点搜索方面,改进的JPS算法的优化效果在复杂街道地图中最为明显,相对于A*减少了84.76%,相较于传统的JPS算法和文献[18]算法都减少了44.83%。在算法搜索用时方面,4种算法用时均较短,其中A*算法在3种地图



(a) A*算法 (b) 传统的JPS算法 (c) 文献[18]算法 (d) 改进的JPS算法

图9 复杂街道路径规划图

Figure 9 Complex street path planning map

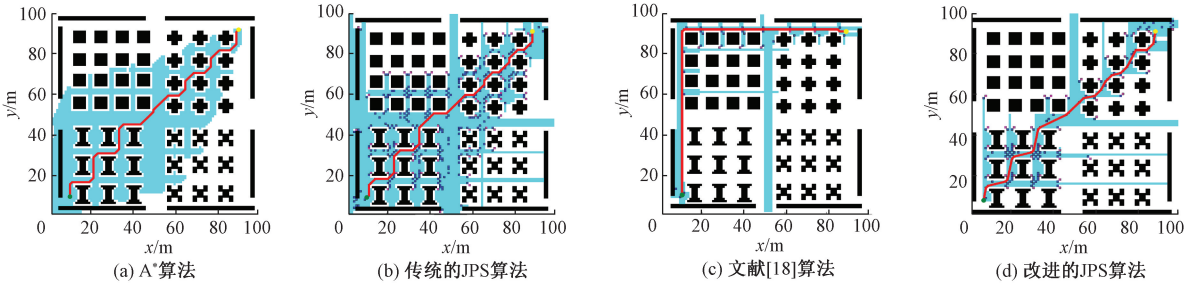


图 10　仓库路径规划图

Figure 10　Warehouse path planning diagram

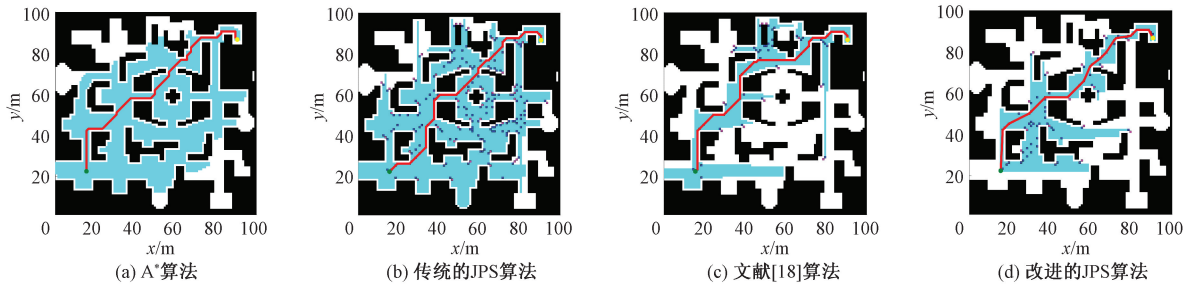


图 11　商场路径规划图

Figure 11　Shopping mall path planning map

表 2　不同算法性能对比

Table 2　Performance comparison of different algorithms

地图	算法	扩展节点数	节点数	轨迹节点数	搜索时间/s	路径拐点数	路径长度/m
街道	A* 算法	1 881	1 881	105	0.45	21	113.11
	传统的 JPS 算法	2 320	153	29	0.22	19	109.50
	文献[18]算法	874	51	29	0.17	19	110.08
	改进的 JPS 算法	819	64	16	0.15	12	107.55
仓库	A* 算法	2 874	2 874	121	0.63	25	136.57
	传统的 JPS 算法	3 544	334	40	0.25	23	130.12
	文献[18]算法	1 155	44	15	0.15	4	160.83
	改进的 JPS 算法	1 522	130	25	0.14	21	125.46
商场	A* 算法	2 999	2 999	109	0.74	22	124.57
	传统的 JPS 算法	3 207	191	33	0.23	19	119.88
	文献[18]算法	1 361	61	25	0.21	17	125.15
	改进的 JPS 算法	1 338	71	17	0.15	14	117.31

场景中的搜索时间普遍比 JPS 类算法耗时长,而改进的 JPS 算法耗时与其他算法相比均用时最短。最后在输出路径的长度和平滑度方面,改进的 JPS 算法经过冗余节点的裁剪,输出的全局路径变得更加平滑和快捷。在复杂街道场景中,相较于 A* 算法、传统的 JPS 算法和文献[18]算法,改进的 JPS 算法的拐点数分别减少了 42.86%,36.84% 和 36.84%,路径长度分别减少了 4.92%,1.78% 和 2.30%;在仓库场景中,相对 A* 算法、传统的 JPS 算法,改进的 JPS 算法的拐点数分别减少了 16.0% 和 8.70%,路径长度相较于 A* 算法、传统的 JPS 算法和文献[18]算法分别减少了 8.13%,3.58% 和 21.99%;在商场场景中,相较于 A* 算法、传统的 JPS 算法和文献[18]算法,改进的 JPS 算法的拐点数分别减少了 36.36%,26.31% 和 17.65%,路径长度分别减少了 5.83%,2.14% 和 6.26%。

将改进的 JPS 算法与改进的 DWA 算法融合,对于融合算法,在同一个仿真环境中设置了 3 个同类型移动机器人 R_1、R_2、R_3 进行路径规划,并设置了 2 个冲突路径以验证改进的 DWA 算法的有效性,如图 12 所示。图 12 中,黑色栅格表示静态障碍物,三角形和圆形分别表示各移动机器人的起点和目标点位置,红色的星形表示改进的 DWA 算法的局部目标点,黑色实线表示改进的 JPS 算法规划出的全局路径,而蓝色、玫红色和绿色等实线分别表示各移动机器人的改进的 DWA 算法规划出的路径轨迹。

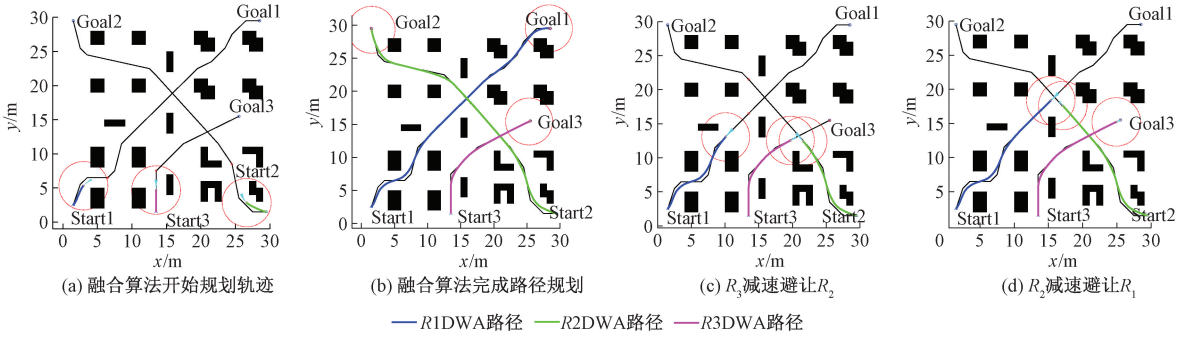


图 12 多机器人路径规划

Figure 12 Multi-robot path planning

由图 12(a)可知,当改进的 JPS 算法和改进的 DWA 算法的融合算法在进行路径规划时,DWA 算法将首先选取 JPS 规划出的全局路径中的某一个轨迹点作为局部目标点进行规划任务,当移动机器人即将抵达这一目标点时,局部目标点将再次递进至 JPS 全局路径中的下一个轨迹点,依次类推,直至最终目标点。从图 12(b)可以看出,改进的 JPS 算法融合改进的 DWA 算法规划出的路径在确保安全性的同时,也提升了路径的平滑性,增强了路径规划的整体性能。在图 12(c)和图 12(d)中,经过改进后的 DWA 算法在遇到交叉路径时会根据基于碰撞距离的动态优先级避让策略自动做出行驶判断。在图 12(d)中,R_1 与 R_2 相遇时,改进后的 DWA 算法判断出了 R_1 具有更高的优先级,并做出了让 R_2 减速避让而 R_1 继续行驶通行的决策。

图 13 为各机器人速度变化图。从图 13 可以看到,在 400~500 的步数段内,R_2 的速度经历了大幅度的减速和加速过程,而 R_1 由于具有高优先级并未进行任何加速、减速操作,仍保持原有状态继续行驶。

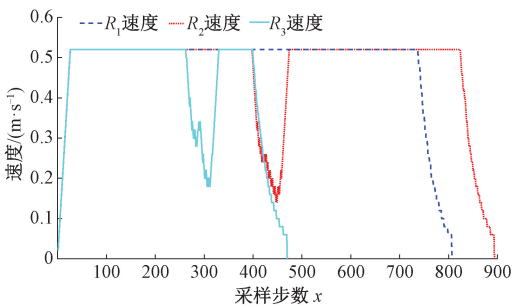


图 13 各机器人速度变化图

Figure 13 Speed variation chart of each robot

根据仿真结果可知,改进后的 DWA 算法可以很好地改善传统 DWA 算法在处理路径冲突等动态环境问题上的缺陷和不足,避免了传统算法因动态避障带来的较大的偏移角度和较长的避障路径等问题,同时也提高了 DWA 算法在进行多机器人路径规划时的协同灵活性和鲁棒适应性。

5 结论

在路径规划理论和技术领域,本文借助 MATLAB,模拟了同类型移动机器人在路径规划过程中出现的路径冲突现象。在此基础上,提出了基于碰撞距离的动态优先级避让策略,为机器人行驶提供了全新的避障思路和解决方案。同时,通过对 JPS 算法进行改进,有效削减了算法在搜索路径时扩展节点、节点以及轨迹节点的数量,消除了原有路径中存在的冗余节点。这不仅提升了算法的资源利用率和寻路效率,而且对于内存资源有限的嵌入式芯片而言,能够显著降低计算资源的消耗,提高算法运行的效率与实时性,进而优化系统的整体性能和响应速度。

参考文献:

[1] 卢凌霄, 董乾鹏, 张天乐, 等. 机器人运动学与运动规划算法综述[J]. 印刷与数字媒体技术研究, 2023(5): 1-16.
LU L X, DONG Q P, ZHANG T L, et al. A review of robot kinematics and motion planning algorithms [J]. Printing and Digital Media Technology Study, 2023(5): 1-16.

[2] 白晓兰, 袁铮, 周文全, 等. 移动机器人路径规划算法研究综述[J]. 机械工程师, 2024(8): 24-28, 33.
BAI X L, YUAN Z, ZHOU W Q, et al. Survey of path planning algorithms for mobile robots [J]. Mechanical Engineer, 2024(8): 24-28, 33.

[3] HART P E, NILSSON N J, RAPHAEL B. A formal basis for the heuristic determination of minimum cost paths [J]. IEEE Transactions on Systems Science and Cybernetics, 1968, 4(2): 100-107.

[4] WANG H, LI G Q, HOU J, et al. A path planning method for underground intelligent vehicles based on an improved RRT* algorithm [J]. Electronics, 2022, 11

(3): 294.

[5] 高岳林,武少华. 基于自适应粒子群算法的机器人路径规划[J]. 郑州大学学报(工学版),2020,41(4):46-51.
GAO Y L, WU S H. Robot path planning based on adaptive particle swarm optimization[J]. Journal of Zhengzhou University (Engineering Science), 2020, 41(4): 46-51.

[6] HAO K, ZHAO J L, LI Z S, et al. Dynamic path planning of a three-dimensional underwater AUV based on an adaptive genetic algorithm [J]. Ocean Engineering, 2022, 263: 112421.

[7] HARABOR D, GRASTIEN A, HARABOR D, et al. Online graph pruning for pathfinding on grid maps[C]//Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence. New York:ACM, 2011: 1114-1119.

[8] 丁承君, 阎欣怡, 冯玉伯, 等. 基于 APF 的 AGV 局部路径规划改进算法研究[J]. 计算机工程与应用, 2022, 58(22): 305-312.
DING C J, YAN X Y, FENG Y B, et al. Improved algorithm of AGV local path planning based on APF[J]. Computer Engineering and Applications, 2022, 58(22): 305-312.

[9] FOX D, BURGARD W, THRUN S. The dynamic window approach to collision avoidance[J]. IEEE Robotics & Automation Magazine, 1997, 4(1): 23-33.

[10] MAGYAR B, TSIOGKAS N, DERAY J, et al. Timed-elastic bands for manipulation motion planning[J]. IEEE Robotics and Automation Letters, 2019, 4(4): 3513-3520.

[11] SHEN D, CHEN Y B, LI L X, et al. Trajectory tracking for autonomous vehicles using robust model predictive control [J]. IFAC-PapersOnLine, 2024, 58(10): 94-101.

[12] 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 (ICMA). Piscataway: IEEE, 2019: 2463-2468.

[13] MA L, GAO X, FU Y X, et al. An improved jump point search algorithm for home service robot path planning [C]//2019 Chinese Control and Decision Conference (CCDC). Piscataway:IEEE, 2019: 2477-2482.

[14] 黄健萌, 吴宇雄, 林谢昭. 移动机器人平滑 JPS 路径规划与轨迹优化方法[J]. 农业机械学报, 2021, 52(2): 21-29, 121.
HUANG J M, WU Y X, LIN X Z. Smooth JPS path planning and trajectory optimization method of mobile robot[J]. Transactions of the Chinese Society for Agricultural Machinery, 2021, 52(2): 21-29, 121.

[15] 秦齐, 万熠, 侯嘉瑞, 等. 基于双向动态跳点搜索算法的 AGV 路径规划研究[J]. 单片机与嵌入式系统应用, 2021, 21(8): 55-58, 63.
QIN Q, WAN Y, HOU J R, et al. AGV path planning based on bidirectional dynamic jumping search algorithm [J]. Microcontrollers & Embedded Systems, 2021, 21(8): 55-58, 63.

[16] SU Q H, MA S 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.

[17] 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.

[18] 张庆, 刘旭, 彭力, 等. 融合 JPS 和改进 A* 算法的移动机器人路径规划[J]. 计算机科学与探索, 2021, 15(11): 2233-2240.
ZHANG Q, LIU X, PENG L, et al. Path planning for mobile robots based on JPS and improved A* algorithm [J]. Journal of Frontiers of Computer Science and Technology, 2021, 15(11): 2233-2240.

[19] KOBAYASHI M, MOTOI N. Path planning method considering blind spots based on ROS navigation stack and dynamic window approach for wheeled mobile robot[C]//2022 International Power Electronics Conference. Piscataway:IEEE, 2022: 274-279.

[20] 刘宙浩, 万超一, 尹明锋, 等. 改进 A* 算法与 DWA 融合的移动机器人的路径规划算法研究[J]. 制造业自动化, 2023, 45(12): 55-60.
LIU Z H, WAN C Y, YIN M F, et al. Research on path planning algorithm of mobile robot based on improved A* algorithm and dynamic window approach[J]. Manufacturing Automation, 2023, 45(12): 55-60.

[21] BERTI H, SAPPA A D, AGAMENNONI O E. Improved dynamic window approach by using Lyapunov stability criteria[J]. Latin American Applied Research, 2008, 38(4): 289-298.

[22] 王超, 梅瑛, 张溢, 等. 基于改进 DWA 算法的移动机器人避障研究[J]. 机械设计与研究, 2024, 40(1): 92-96.
WANG C, MEI Y, ZHANG Y, et al. Research on obstacle avoidance of mobile robot based on improved DWA algorithm [J]. Machine Design & Research, 2024, 40(1): 92-96.

[23] 蔡佳成, 白克强, 李旭春, 等. 基于 JPS 改进的移动

机器人路径规划算法[J]. 计算机应用研究, 2022, 39(7): 1985-1991.
CAI J C, BAI K Q, LI X C, et al. Improved path planning algorithm for mobile robot based on JPS[J]. Application Research of Computers, 2022, 39(7): 1985-1991.
[24] 石英托, 陈华, 张连新, 等. 基于改进A*算法的AGV转运机器人路径规划研究[J]. 制造技术与机床, 2022(5): 19-22.
SHI Y T, CHEN H, ZHANG L X, et al. Research on path planning of AGV transport robot based on improved A* algorithm[J]. Manufacturing Technology & Machine Tool, 2022(5): 19-22.

Path Planning for Mobile Robots Based on Improved JPS and DWA Algorithms

CHENG Bo[1], CAI Longshuai[1], GUO Guifang[2], ZHANG Xuan[1]

(1. School of Construction Machinery, Chang'an University, Xi'an 710064, China; 2. School of Information Engineering, Xizang Minzu University, Xianyang 712082, China)

Abstract: To tackle the problems of heightened search blindness, elevated memory usage, and redundant nodes in the output path resulting from the access of a large number of irrelevant extension nodes in traditional JPS algorithms, an improved JPS algorithm was proposed. This algorithm based on the priority of the target-point search direction and a dynamic-weight evaluation function. Firstly, according to the positions of the target point and the mobile robot, the priority of the direction of target point in the algorithm's path-finding process was enhanced. A distance-based dynamic-weight evaluation function was introduced to mitigate the resource waste and efficiency loss caused by searching for irrelevant nodes. Meanwhile, the global path output by the improved JPS algorithm underwent secondary planning. Redundant nodes in the original path were eliminated, making the global path smoother. Secondly, the DWA algorithm was introduced and improved as a local path-planning algorithm. The improved DWA algorithm adopted a dynamic priority strategy based on collision distance to automatically avoid mobile robots on cross-paths. Finally, the improved JPS algorithm and DWA algorithm were separately simulated and verified. The results indicated that, compared with traditional jump-point algorithms, the improved algorithm in this study reduced the average number of searched extended nodes by 60.0%, the average number of trajectory nodes by 43.6%, and the average number of path turning points by 23.9%. The improved DWA algorithm could effectively address the drawbacks and limitations of traditional DWA algorithms in dynamic environmental problems such as path conflicts, and improve the collaboration and adaptability of DWA algorithms in multi-robot path planning.

Keywords: jump point search algorithm; path planning; priority obstacle avoidance; dynamic weight evaluation function; conflict path