文章编号：2095-2163(2024)02-0018-10 中图分类号：TP242.6 文献标志码：A

基于改进 A-star 与 DWA 相融合的移动机器人动态路径规划算法

汤玉春，王睿忠
（上海理工大学 光电信息与计算机工程学院，上海 200093）

摘 要：A-star 算法常用于移动机器人的全局路径规划，但在复杂场景中 A-star 算法存在耗时长、搜索节点过多、路径不平滑、不能避开环境中未知的障碍物等问题。针对于此，本文提出一种融合路径规划算法。首先，在 A-star 算法的基础上引入环境中的障碍物信息和搜索节点到起始位置的距离信息动态调节启发函数的权重，减少搜索节点数，提升 A-star 算法的性能；然后，利用自适应分段步长的高阶贝塞尔曲线对路径进行优化，减少转折点提升路径的平滑性；最后，将改进 A-star 算法规划的全局路径作为引导，将路径节点作为 DWA 算法的中间目标，实现全局路径规划和局部规划的融合，使移动机器人在找到全局最优路径的同时，能够避开环境中的未知障碍物，实现移动机器人的动态路径规划。仿真结果验证了该算法的有效性。

关键词：路径规划；A-star 算法；动态权重；贝塞尔曲线；DWA 算法

Dynamic path planning algorithm for mobile robots based on the fusion of improved A-star and DWA

TANG Yuchun, WANG Ruizhong

(School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China)

Abstract: The A-star algorithm is commonly used for global path planning in mobile robotics. However, in complex environments, the A-star algorithm faces issues such as long computation times, excessive search nodes, non-smooth paths, and an inability to avoid unknown obstacles in the environment. To address these challenges, this paper introduces a fusion path planning algorithm. Firstly, it enhances the A-star algorithm by dynamically adjusting the weights of the heuristic function using information about obstacles in the environment and the distance of search nodes from the starting position, thereby reducing the number of search nodes and improving the A-star algorithm's performance. Next, it optimizes the path using a high-order Bézier curve with adaptive segmented step lengths, reducing turning points to enhance path smoothness. Finally, it combines the globally planned path from the improved A-star algorithm as guidance, treating the path nodes as intermediate goals for the Dynamic Window Approach (DWA) algorithm. This fusion of global and local path planning enables the mobile robot to find the global optimal path while avoiding unknown obstacles in the environment, facilitating dynamic path planning for mobile robots. Simulation experiments validate the algorithm's effectiveness.

Key words: path planning; A-star algorithm; dynamic weight; Bezier curve; DWA algorithm

0 引 言

随着科技的发展和社会生产的需要，移动机器人被广泛应用于各行各业，路径规划算法作为实现移动机器人自动导航过程中不可缺少的一部分，一直以来都是热点研究对象。路径规划的目的是快速准确的搜索出一条从起始位置到目标位置的无碰撞最优路径[1-2]。路径规划算法可分为全局路径规划和局部路径规划，用于全局路径规划的算法有 Dijkstra 算法[3-4]、A-star 算法[5-6]、D* 算法[7]、LPA* 算法等[8]，用于局部路径规划的算法有 DWA 算法[9-10]、人工势场算法[11]、遗传算法[12]等。

A-star 算法有着较好准确性，被广泛应用于移动机器人的自动导航领域。但传统 A-star 算法存在搜索节点多、算法耗时长、转折点过多、路径不平滑、无法避开环境中未知障碍物等问题，众多学者提出了不同的改进方法。Zafar M A 等[13]通过跳点方式，对 A-star 算法做出改进，使其不再将节点扩展

作者简介：汤玉春(1997-)，男，硕士研究生，主要研究方向：SLAM 与机器人导航；王睿忠(1990-)，男，硕士研究生，主要研究方向：SLAM 与机器人导航。

收稿日期：2023-03-04

到相邻节点，而是通过使用跳跃点的概念进行扩展，直到到达障碍物或地图的终点。但在进行某一方向上的扩展时，直到遇到障碍物或地图边缘才会进行下一方向的扩展，使得该算法在空旷区域会搜索完场景中所有的节点，因此算法改进效果并不理想。XiangRong T 等[14]利用双向 A-star 算法提升搜索效率，分别从起点和终点向中间搜索，当两个搜索任务的搜索分支相遇时算法结束，从而搜索到最短路径；但当起点和终点之间存在多个最短路径时，算法可能无法找到最优解。卞永明等[15]在 A-star 算法原有估价函数的基础上，引入惩罚函数和奖励因子，以减少转折点多的问题。虽然转折点减少了，但路径上始终会存在角度较大的转折点。谢春丽等[16]提出利用二阶或三阶贝塞尔曲线，对 A-star 算法的全局路径进行平滑处理，但并未充分考虑贝塞尔曲线在分段处的平滑性。单一使用全局路径规划算法虽然能找到最优路径，但并不能避开环境中未知的障碍物。张恒瑞等[17]将改进的 A-star 算法和人工势场法相融合，在全局路径最优的同时实现动态避障。但由于人工势场算法需要迭代搜索，容易受到环境变化的影响，而且存在局部最优解，所以收敛性比较差，无法保证找到最优解。槐创锋等[18]将改进的 A-star 算法和 DWA 算法相融合，将 8 领域搜索方向改为 5 领域搜索方向以降低计算量，虽然融合算法能实现动态避障，但因为没有考虑剩余方向的障碍物，搜索存在陷入死区的可能。

综上所述，本文提出一种改进的 A-star 和 DWA 相融合算法，将障碍物信息和节点到起始位置的距离信息引入代价函数，随着搜索节点向目标位置的靠近，动态调节启发函数的权重，从而减少搜索节点数，提升算法效率；利用分段高阶贝塞尔曲线进行路径平滑处理，减少转折点；将全局路径中的节点作为 DWA 算法的中间目标点，在保证全局路径最优的同时，实现移动机器人的动态避障。

1　A-star 算法改进

1.1　A-star 算法基本原理

A-star 是在 Dijkstra 算法的基础上，结合 DFS 算法的优点引入了启发函数的概念，在地图中搜索节点时利用启发信息作为指导减少需要搜索的节点数，增加搜索最优路径的效率，通过评价各个节点的代价值，获取下一个需要拓展的节点，直到目标位置。A-star 算法中用 $f(n)$ 表示第 n 个节点到目标位置的代价值，代价值最小的节点作为下一次寻找邻接节点的父节点，$g(n)$ 表示从起点到当前节点的实际代价值，$h(n)$ 是 A-star 算法的启发函数，表示第 n 个节点到目标位置的预计代价值。$f(n)$ 的计算如式(1)所示：

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

A-star 基于栅格地图实现，其寻找邻接节点的方式有四邻域和八邻域。四邻域只在水平和垂直方向寻找邻接的节点(如图 1)，八邻域需要寻找对角的节点(如图 2)。采用四邻域方式算法搜索的节点更少，路径转折点更少，而八邻域方式搜索节点得到的路径更短，且运动方式更符合移动机器人的实际情况。

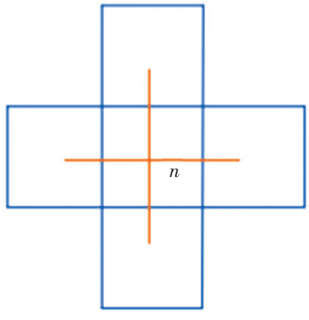


图 1　四邻域

Fig. 1　Four-neighborhood

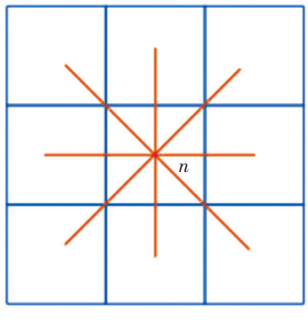


图 2　八邻域

Fig. 2　Eight-neighborhood

节点间的代价值通过栅格间的距离表示，如式(2)~式(4)表示了 A-star 算法中 3 种常见的距离计算方式。

$$g_1(n) = \max\{|x_1 - x_2|, |y_1 - y_2|\} \tag{2}$$

$$g_2(n) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3}$$

$$g_3(n) = |x_1 - x_2| + |y_1 - y_2| \tag{4}$$

式中：(x_1, y_1) 表示父节点栅格中心所在位置的横纵坐标，(x_2, y_2) 表示父节点邻接节点栅格中心所在位置的横纵坐标，$g_1(n)$ 表示切比雪夫距离，$g_2(n)$ 表示欧几里得距离，$g_3(n)$ 表示曼哈顿距离。

切比雪夫距离和曼哈顿距离的计算方式因为省

去了开根号的过程,所以没有浮点型运算,计算效率更高,欧几里得距离计算方式则更加符合移动机器人实际的应用场景。

1.2 启发函数改进

使用 A-star 算法搜索最优路径的过程中,随着搜索节点数的增多 open_list 和 close_list 中的节点会不断增多。每一轮邻接节点加入后,都需要对 open_list 中的所有节点进行遍历以找出代价值最小的节点,随着节点的增多计算量不断增大,最终导致算法搜索效率降低。虽然通过增加启发函数的权重能有效减少算法整体搜索的节点数,但会使得 A-star 算法存在没有完全搜索所有可行解的可能,导致规划出的路径不一定是最优。带有权重的代价函数定义如式(5):

$$f(n) = g(n) + w * h(n) \tag{5}$$

式中 w 表示权重系数。

A-star 算法中启发函数的权重选取对搜索出的路径有着重要的影响。当 $w = 0$ 时,A-star 算法退化为 Dijkstra 算法,此时的 A-star 算法总能规划出一条最短路径,但算法运行速度慢;若 $w * h(n)$ 远远大于 $g(n)$ 时,$f(n)$ 的值主要取决于 $h(n)$,A-star 算法变为 BFS 算法;当 $w < 1$ 时,启发函数的值小于到目标位置的实际距离,A-star 算法可以搜索出最短路径,但搜索范围大,搜索的无用节点较多导致算法效率低;当 $w > 1$ 时,启发函数的值大于到目标位置的实际距离,搜索的节点数少,搜索路径速度快,但是搜索出来的路径不一定是最短的。由此可见,若启发函数使用固定的权重系数,算法本身并不能同时兼顾搜索路径的效率和路径最优。提出动态权重的方法对 A-star 算法的启发函数做出改进,让启发函数的权重随着移动机器人所处环境的障碍物栅格率和距离目标位置的远近动态变化,以此来减少搜索节点数,提升算法的效率。在障碍物较多的场景中,当 w 较大时启发函数占有更大的权重,算法搜索的节点减少,但容易陷入局部最优,导致规划出的路径可能并非最优,所以在障碍物较多的环境中 w 的值应该尽量减小。在障碍物较少的环境中,若 w 取值较小,则启发函数占有较小的权重,此时算法会搜索很多无用的节点,导致算法效率降低,因此在障碍物较少的场景中,可以增大 w 的值让搜索节点尽快靠近目标位置。若用 k 表示环境中的障碍物栅格率,其计算如式(6):

$$k = \frac{num_d}{num_s} \tag{6}$$

式中:num_d 表示环境中障碍物栅格数,num_s 表示总栅格数。

让启发函数的权重随当前节点距离目标位置的远近而动态变化,可以选择在距离目标位置更近时增加启发函数的权重,也可以选择在距离起点位置更近时增加启发函数的权重。通常,距离目标位置更近时增加启发函数的权重更有利于算法搜索到最优路径[19]。

定义权重计算公式如式(7),当移动机器人所在的环境确定后 k 值确定。当起始位置与目标位置确定后,起点与目标位置的距离值 d_t 确定,其计算如式(8)。式中 (x_s, y_s) 表示起点坐标,(x_g, y_g) 表示目标位置坐标。在距离起点近的时候以寻找最短路径为主,此时启发函数的权重系数较小,搜索节点数量较多。式(9)中,d_s 表示当前节点与目标位置的距离,(x_c, y_c) 表示当前节点所在位置。在逐渐向目标位置靠近的过程中,d_s 逐渐增大,w 的值逐渐增大,算法搜索节点的数量逐渐减少,此时以快速靠近目标位置为主。A-star 算法改进后的节点代价计算如式(10)。

$$w = (1 - \ln k) \frac{1}{e^{\left|1-\frac{d_s}{d_t}\right|}} \tag{7}$$

$$d_t = \sqrt{(x_s - x_g)^2 + (y_s - y_g)^2} \tag{8}$$

$$d_s = \sqrt{(x_s - x_c)^2 + (y_s - y_c)^2} \tag{9}$$

$$f(n) = g(n) + (1 - \ln k) \frac{1}{e^{\left|1-\frac{d_s}{d_t}\right|}} \sqrt{(x_c - x_g)^2 + (y_c - y_g)^2} \tag{10}$$

1.3 路径平滑处理

基于栅格地图的路径规划算法搜索出的最优路径普遍存在较多转折点,路径不平滑等特点,致使移动机器人在移动过程中出现频繁减速和原地旋转的情况,影响移动机器人的行驶效率,增加移动机器人的能量消耗。贝塞尔曲线是一种使用数学方法描述的曲线,被广泛用于计算机图形学和动画中生成平滑的曲线。可利用分段高阶贝塞尔曲线对 A-star 算法规划出的全局路径进行平滑处理。

贝塞尔曲线由其控制点决定,n 个控制点对应 $n-1$ 阶的贝塞尔曲线。一阶贝塞尔是一条直线,如图 3 所示。在同一平面上选取两个不共线的控制点 P_0 和 P_1,在线段 P_0P_1 上任选一点 A ,用 t 表示线段 P_0A 和线段 P_0P_1 之间的比值,其取值范围为 0~1,点 A 的位置用 $P(t)$ 表示,其计算如式(11)。变形得到一阶贝塞尔公式如式(12)。

$$P(t) = P_0 + (P_1 - P_0)t \tag{11}$$

$$P(t) = (1 - t)P_0 + tP_1 \tag{12}$$

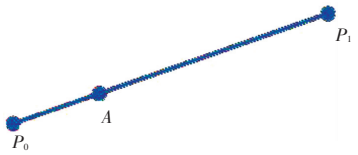


图 3　一阶贝塞尔曲线

Fig. 3　First-order Bezier curve

二阶贝塞尔曲线由同一平面上的 3 个控制点 P_0、P_1、P_2 构成，如图 4 所示。在线段 P_0P_1 上任选一点 A，在线段 P_1P_2 上选一点 B，在 AB 的连线上选一点 C，并满足式(13)所示的关系。由一阶贝塞尔公式可得点 A 的位置 $P_A(t)$，点 B 的位置 $P_B(t)$，如式(14)、式(15)：

$$t = \frac{P_0A}{P_0P_1} = \frac{P_1B}{P_1P_2} = \frac{AD}{AB} \tag{13}$$

$$P_A(t) = (1 - t)P_0 + tP_1 \tag{14}$$

$$P_B(t) = (1 - t)P_1 + tP_2 \tag{15}$$

已知点 A 和 B 的位置由一阶贝塞尔公式可求得点 D 的位置，用 $P(t)$ 表示，其计算如式(16)：

$$P(t) = (1 - t)P_A(t) + tP_B(t) \tag{16}$$

联立式(14)~式(16)可得式(17)：

$$p(t) = (1 - t)((1 - t)P_0 + tP_1) + t((1 - t)P_1 + tP_2) \tag{17}$$

化简式(17)得到二阶贝塞尔公式如式(18)：

$$P(t) = (1 - t)^2P_0 + 2t(1 - t)P_1 + t^2P_2 \tag{18}$$

如图 4 所示，t 的取值从 0 到 1 逐渐变大，点 A 的位置逐渐向 P_1 靠近，点 B 的位置逐渐向 P_2 靠近，点 C 的位置也随之发生变化，且点 C 始终相切于线段 AB，最终点 C 的运动轨迹如图 4 中的曲线 P_0CP_2 所示，即由控制点 P_0、P_1、P_2 所确定的二阶贝塞尔曲线。

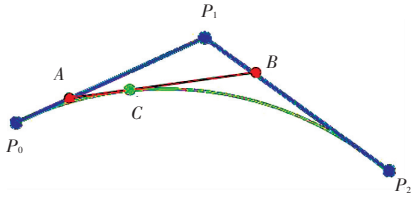


图 4　二阶贝塞尔曲线

Fig. 4　Second-order Bezier curve

三阶贝塞尔曲线由 4 个控制点 P_0、P_1、P_2、P_3 得到，在 P_0P_1、P_1P_2、P_2P_3 之间分别取点 A、B、C。A、B、C 3 个点对应着二阶贝塞尔曲线，继续在 AB、BC 之间分别取点 D、E 对应着一阶贝塞尔曲线，继续在 DE 上取点 F，F 运动轨迹为三阶贝塞尔曲线，如图 5 中的曲线 P_0FP_3 所示。由推导二阶贝塞尔曲线的过程同理可得到三阶贝塞尔曲线的公式如式(19)：

$$P(t) = P_0(1 - t)^3 + 3P_1t(1 - t)^2 + 3P_2t^2(1 - t) + P_3t^3 \tag{19}$$

每一次取点的过程都是一次降阶的过程，可以看出高阶贝塞尔曲线由低阶贝塞尔曲线递归实现，高阶贝塞尔曲线的公式如式(20)、式(21)所示：

$$P(t) = \sum_{i=0}^{n} B_{i,n}(t)P_i \tag{20}$$

$$B_{i,n}(t) = C_n^i t^i (1 - t)^{n-i} = \frac{n!}{i!\ (n - i)!} t^i (1 - t)^{n-i}$$
$$i = 0, 1, \cdots, n \tag{21}$$

式中：P_i 表示第 $i + 1$ 个控制点，n 表示贝塞尔曲线的阶数。

贝塞尔曲线除了具有递归性以外，还具有凸包性。从图 5 可以看出，贝塞尔曲线始终会在包含了所有控制点的最小凸多边形中，并且随着阶数的增加，最终得到的贝塞尔曲线距离最初始的控制点越远，但 A-star 算法规划出的最优路径的节点是贝塞尔曲线最初始的控制点，且在需要绕开障碍物的地方路径通常是紧挨着障碍物绕行的，这就导致平滑处理后的路径会偏入障碍物区域，显然移动机器人沿着该路径行驶下去最终会撞上障碍物，如图 6 所示，因此需要对高阶贝塞尔曲线进行分段处理。

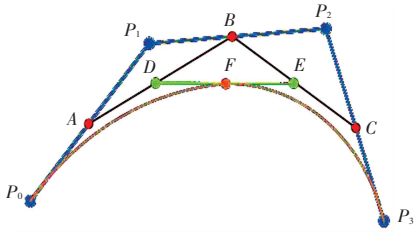


图 5　三阶贝塞尔曲线

Fig. 5　Third-order Bezier curve

已知路径点是由二维数据点 (x, y) 组成，将存放路径点的数组记为 path_list，设置分段步长为 ζ，即在 path_list 中每 ζ 个数据点记为一组，用 Bezier_list 存放，分段后的贝塞尔曲线由 Bezier_list 中的 ζ 个控制点构成。为了保证平滑处理后整体路径的连续性，将上一次分段结束的路径点作为下一次分段的起始点。设置 ζ 的值为 16，从图 7 中可以看出，在点(3,11)和点(17,20)处依然存在明显转折点，因此采用固定分段步长，分段后的贝塞尔曲线在分段处并不能保证路径的平滑性，这是因为两段贝塞尔

曲线在连接处的一阶导不连续。

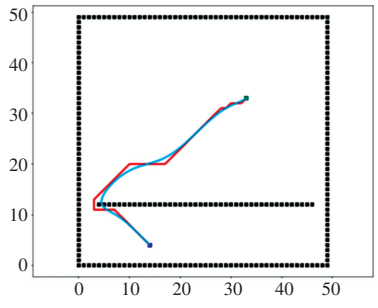


图 6 高阶贝塞尔曲线

Fig. 6 Higher-order Bezier curve

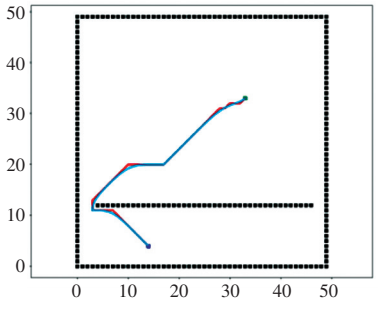


图 7 固定步长分段高阶贝塞尔曲线

Fig. 7 Fixed step segmented higher-order Bezier curve

对式(20)进行一阶求导得式(22)、式(23):

$$P'(t) = \sum_{i=0}^{n} P_i B'_{i,n-1}(t) \tag{22}$$

$$P'(t) = \sum_{i=0}^{n} P_i n(B_{i-1,n-1}(t) - B_{i,n-1}(t)) \tag{23}$$

将式(23)进行展开得式(24):

$$P'(t) = \sum_{i=1}^{n} nP_i B_{i-1,n-1}(t) - \sum_{i=0}^{n-1} nP_i B_{i,n-1}(t) + n(P_0 B_{0-1,n-1}(t) - P_n B_{n,n-1}(t)) \tag{24}$$

当 $i=0$ 和 $i=n$ 时,由式(21)可得 $B_{0-1,n-1}(t) = B_{n,n-1}(t) = 0$, 化简式(24)得式(25),整理得到高阶贝塞尔曲线的一阶导,如式(26):

$$P'(t) = \sum_{i=0}^{n-1} nP_{i+1} B_{i,n-1}(t) - \sum_{i=0}^{n-1} nP_i B_{i,n-1}(t) + nP_{n+1} B_{n,n-1}(t) \tag{25}$$

$$P'(t) = \sum_{i=0}^{n-1} n(P_{i+1} - P_i) B_{i,n-1}(t) \tag{26}$$

当 $t=0$ 时,式(26)展开后除 $B_{0,n-1}=1$ 以外,其他控制点的系数均等于 0;当 $t=1$ 时,式(26)展开后除 $B_{n,n-1}=1$ 以外,其他控制点的系数均等于 0。因此,得到式(27):

$$\begin{cases} P'(0) = n(P_1 - P_0) \\ P'(1) = n(P_n - P_{n-1}) \end{cases} \tag{27}$$

分段后的高阶贝塞尔曲线由 ζ 个控制点构成,设在分段处左右两段高阶贝塞尔曲线分别为 $P_{\text{left}}(t)$ 和 $P_{\text{right}}(t)$, 如图 8 所示。要保证分段高阶贝塞尔曲线的平滑性,则需要 $P'_{\text{left}}(1) = P'_{\text{right}}(0)$。从式(27)可知,当 $P_{\text{left},\zeta-2} - P_{\text{left},\zeta-1} = P_{\text{right},0} - P_{\text{right},1}$ 时,能实现分段处的平滑。由于左边贝塞尔曲线的最后一个控制点也是右贝塞尔曲线的第一个控制点,所以在分段后,左边贝塞尔曲线的最后一个控制点的路径点应该与其邻近的两个路径点满足式(28)中的关系。若 path_list[ζ] 不满足式(28)中的关系,则令 $\zeta = \zeta + 1$ 进行条件循环。找到满足条件的路径点后,将 ζ 个路径点都放入 Bezier_list 中作为高阶贝塞尔曲线的控制点,调用式(20)和式(21)计算得到平滑处理后的路径。一次分段结束后,将 ζ 设置为初始 ζ 继续进行下一次分段,直到遍历完 path_list 中的所有路径点。

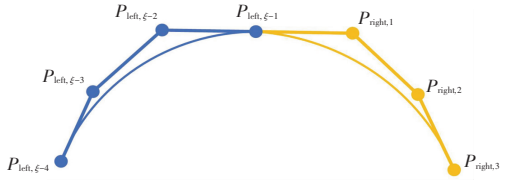


图 8 分段高阶贝塞尔曲线

Fig. 8 Segmented higher-order Bezier curve

$$\begin{cases} \text{path\_list}[\zeta-1].x - \text{path\_list}[\zeta].x = \\ \quad \text{path\_list}[\zeta].x - \text{path\_list}[\zeta+1].x \\ \text{path\_list}[\zeta-1].y - \text{path\_list}[\zeta].y = \\ \quad \text{path\_list}[\zeta].y - \text{path\_list}[\zeta+1].y \end{cases} \tag{28}$$

2 融合路径规划

2.1 速度采样

DWA 算法实现原理主要是在速度空间 (v,w) 中采样多组速度,其中 v 表示移动机器人的线速度, w 表示移动机器人的角速度。通过这些速度根据移动机器人的运动模型预测移动机器人在一定时间内的运动轨迹,通过评价函数对这些轨迹进行打分,选择最优轨迹对应的一组速度发给移动机器人的底盘,控制移动机器人向目标位置移动,因此对移动机器人的位置控制问题最终转换成对移动机器人的速度控制问题。在实际应用中,由于自身硬件的限制,移动机器人的速度存在最大上限,因此采样速度应该控制在自身的最大速度与最小速度范围内。定义 V_m 为移动机器人最大最小速度约束的集合,其表达如式(29):

$$V_m = \{(v,w) \mid v \in [v_{\min}, v_{\max}],\ w \in [w_{\min}, w_{\max}]\} \tag{29}$$

受移动机器人动力的限制,其线加速度 $\dot{v}$ 和角

加速度 $\dot{w}$ 均存在限制。其中 v_c 表示当前时刻的线速度，w_c 表示当前时刻的角速度，定义 V_d 为以 Δt 为预测时间间隔内能产生的最大最小的速度集合，其表达如式(30)所示：

$$V_d = \{(v,w) \mid v \in [v_c - \dot{v}\Delta t, v_c + \dot{v}\Delta t], w \in [w_c - \dot{w}\Delta t, w_c + \dot{w}\Delta t]\} \tag{30}$$

出于对移动机器人的安全性考虑，为了防止其因速度因素导致与环境中的障碍物发生碰撞，需要其在最大减速度下与障碍物发生碰撞前将速度减为零。定义 V_a 为制动距离约束的速度集合，$dist(v,w)$ 表示当前轨迹距离最近障碍物的距离，如果当前采样的速度可以在碰到障碍物之前停止，则该速度为允许的速度。V_a 表达如式(31)所示：

$$V_a = \left\{(v,w) \mid v \leqslant \sqrt{2dist(v,w)\dot{v}}, w \leqslant \sqrt{2dist(v,w)\dot{w}}\right\} \tag{31}$$

综上所述，定义 V_r 为三项约束条件下的交集，则其表达如式(32)所示，$v_{\max}$、$v_{\min}$ 分别表示满足约束条件的最大线速度和最小线速度，$w_{\max}$、$w_{\min}$ 分别表示满足约束条件的最大角速度和最小角速度。

$$V_r = V_m \cap V_a \cap V_d = \{(v,w) \mid v \in [v_{\min}, v_{\max}] \cap w \in [w_{\min}, w_{\max}]\} \tag{32}$$

2.2　轨迹预测与评价函数

本文选用的移动机器人为二轮差分底盘，其运动学模型如式(33)所示，v 和 w 分别表示移动机器人在世界坐标系下的线速度和角速度，在速度采样周期 Δt 内，假设速度恒定。(x,y,θ) 表示移动机器人在世界坐标系中的位姿。

$$\begin{cases} x_t = x_{t-1} + v\Delta t\cos(\theta_{t-1}) \\ y_t = y_{t-1} + v\Delta t\sin(\theta_{t-1}) \\ \theta_t = \theta_{t-1} + w\Delta t \end{cases} \tag{33}$$

利用 DWA 算法进行移动机器人的路径规划时，需要对多个二维速度组下的运动轨迹进行预测。在速度采样空间 V_r 中，设线速度分辨率和角速度分辨率分别为 d_v 和 d_w，在线速度取值范围 $[v_{\min}, v_{\max}]$ 中每隔 d_v 取一个值，在角速度取值范围 $[w_{\min}, w_{\max}]$ 中每隔 d_w 取一个值，每一个线速度值对应一个角速度值共同组成二维速度组。设采样速度组的个数为 n，其计算如式(34)所示。已知线速度和角速度，根据式(33)可以计算出机器人预测轨迹和位姿。

在获取移动机器人的预测轨迹后，通常有多组速度对应的多条轨迹可行，因此需要利用评价函数对这些轨迹进行评估，选择出最优的轨迹作为移动机器人的运动路径。评价标准为：在局部导航过程中，移动机器人能有效避开障碍物且快速向目标所在位置移动。评价函数的定义如式(35)所示，其中 α、β、γ 表示权重系数。

$$n = \frac{v_{\max} - v_{\min}}{d_v} \times \frac{w_{\max} - w_{\min}}{d_w} \tag{34}$$

$$G(v,w) = \alpha.head(v,w) + \beta.velocity(v,w) + \gamma dist(v,w) \tag{35}$$

$$head(v,w) = \frac{\pi - \theta_i}{\sum_{i=1}^{n}(\pi - \theta_i)} \tag{36}$$

$$velocity(v,w) = \frac{velocity(i)}{\sum_{i=1}^{n} velocity(i)} \tag{37}$$

$$dist(v,w) = \frac{dist(i)}{\sum_{i=1}^{n} velocity(i)} \tag{38}$$

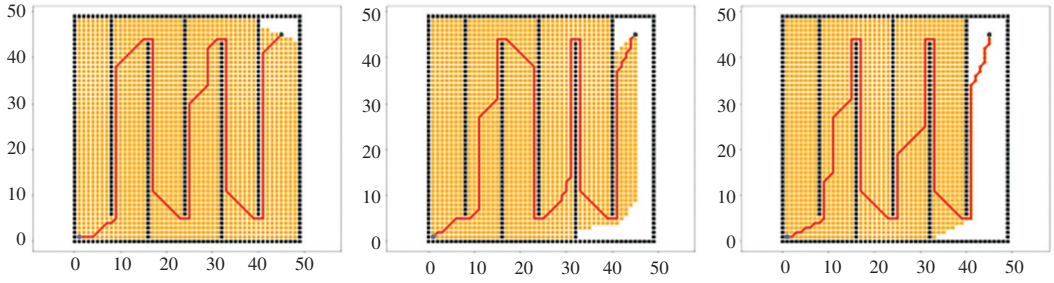
其中，$head(v,w)$ 是方位角评价函数，表示移动机器人到达预测轨迹末端时，其朝向与目标位置之间的角度差距。在预测轨迹末端机器人与目标位置的夹角为 θ，$head(v,w) = \pi - \theta$，θ 的值越小评价函数得分越高。$velocity(v,w)$ 是速度评价函数，表示当前轨迹对应移动机器人的速度，速度越快评价函数得分越高。$dist(v,w)$ 是距离评价函数，表示移动机器人在当前轨迹上与最近障碍物之间的距离，移动机器人距离障碍物越近，其值越低，评价函数得分越低。如果当前轨迹上有障碍物，则将其标记为不可使用。不同评价函数的评价标准不同，会出现评价函数不连续的问题，因此需要对各评价函数进行归一化处理。其计算如式(36)～式(38)所示，n 表示预测轨迹的总数，i 表示第 i 条轨迹。

2.3　融合算法

全局路径规划是在已知的静态环境中，计算出移动机器人起始位置到目标位置的最优路径。由于需要预先知道环境中的障碍物分布，当环境发生变化后原有路径就不能保证移动机器人能够顺利到达目标点。局部路径规划可以应对环境信息未知的情况，其重点关注的是移动机器人当前所处位置的局部环境信息，移动机器人利用自身携带的传感器实时收集环境信息，从而获取障碍物的具体位置和几何特征，使得移动机器人能够避开环境中的未知障碍物。因为缺少对全局环境的掌握，局部路径规划算法规划出的路径可能不是最优甚至找不到完整路径，因此将改进的 A-star 算法和 DWA 算法相结合，保证移动机器人在找到最优路径的同时能够避开场

景中的动态障碍物。

在全局栅格地图已知的情况下,设定目标位置,利用改进 A-star 算法进行全局最优路径规划,在得到全局最优路径后,将全局路径上的路径节点作为局部路径规划的目标位置,如图 9 所示,利用 DWA 算法根据移动机器人当前采样的线速度和角速度进行轨迹预测,通过评价函数选取出最优轨迹作为移动机器的局部路径,将对应的速度发给移动机器人的控制中心,从而完成在动态环境中的避障。



(a) Dijkstra 算法 (b) A-star 算法 (c) 改进的 A-star 算法

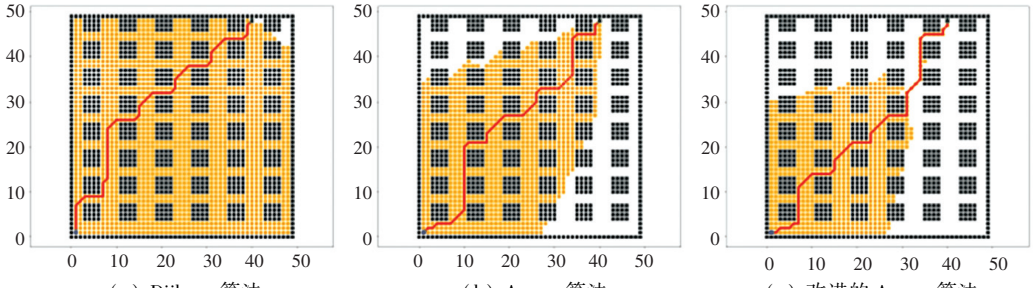
图 9 巷道环境路径规划效果对比

Fig. 9 Comparison of path planning in roadway environment

3 实验与分析

3.1 改进 A-star 算法实验与分析

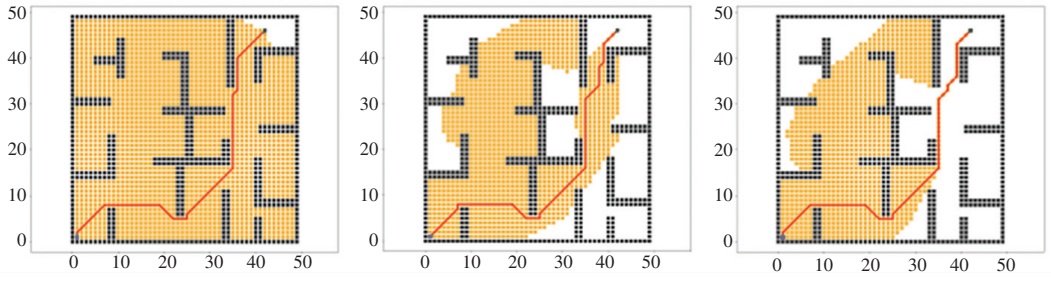
为了形象展示改进 A-star 算法的路径节点搜索过程和路径优化前后的效果对比,利用 python 构建栅格地图进行仿真实验,所用平台为 Windows10,搭载处理器为 i7-6500U,运行内存为 4 G。构建栅格地图模拟不同的现实场景进行仿真实验,定义每个栅格边长为 1 m 的正方形,黑色小栅格表示障碍物,为不可行驶区域,白色区域为可自由行驶区域,黄色小栅格表示已经搜索过的节点。场景 1 为常见的巷道环境(图 9),场景 2 为常见的仓库环境(图 10),场景 3、4 为迷宫环境(图 11、图 12),其设计思路来源于文献[20]。不同场景的障碍物分布和起止位置均有所不同,具体参数设置见表 1。改进的 A-star 算法分别对比传统 A-star 算法和 Dijkstra 算法。算法性能的评价指标主要有搜索节点数、路径长度、算法耗时等。其中搜索节点数的统计以 close_list 中的节点数为主。算法耗时以传入参数开始到输出 path_list 为止。



(a) Dijkstra 算法 (b) A-star 算法 (c) 改进的 A-star 算法

图 10 仓库环境路径规划效果对比

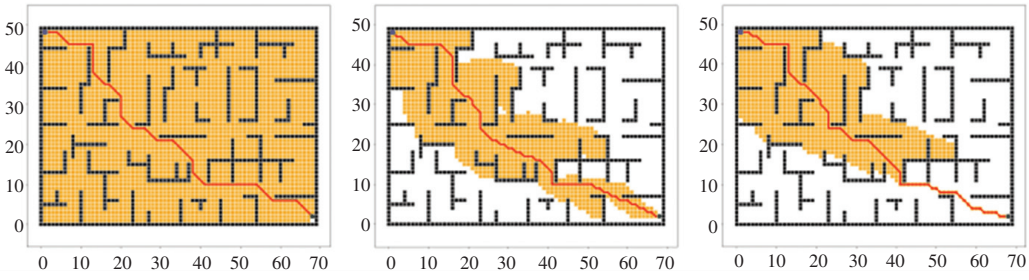
Fig. 10 Comparison of path planning in warehouse environmental



(a) Dijkstra 算法 (b) A-star 算法 (c) 改进的 A-star 算法

图 11 简单迷宫环境路径规划效果对比

Fig. 11 Comparison of path planning in simple maze environment



（a）Dijkstra 算法　（b）A-star 算法　（c）改进的 A-star 算法

图 12　复杂迷宫环境路径规划效果对比

Fig. 12　Comparison of path planning in complex maze environment

表 1　地图参数

Table 1　Map parameters

地图场景	起点位置	目标位置	障碍物栅格点数	总栅格点数	k
巷道环境	(1,1)	(45,45)	411	2 500	0.16
仓库环境	(1,1)	(40,48)	940	2 500	0.38
简单迷宫环境	(1,1)	(42,46)	574	2 500	0.23
复杂迷宫环境	(1,48)	(68,2)	695	3 500	0.20

从表 2 中的数据可以看出，改进的 A-star 算法对比传统 A-star 算法，在不同的场景中算法性能均有明显提升，且均能保证找到的路径最优。在巷道环境中，本文改进的 A-star 算法比传统 A-star 算法搜索节点数减少了 8.2%，算法耗时减少了 10.1%。在仓库环境中，改进的 A-star 算法比传统 A-star 算法搜索节点数减少了 29.5%，算法耗时减少了 47.4%。在简单迷宫环境中，改进的 A-star 算法比传统 A-star 算法搜索节点数减少了 13.8%，算法耗时减少了 21%。在复杂迷宫环境中，改进的 A-star 算法比传统 A-star 算法搜索节点数减少了 14.7%，算法耗时减少了 37%。从图 11(c)和图 12(c)中可以明显看出，在搜索的节点靠近起点位置时，由于改进 A-star 算法的启发函数的权重系数小于 1，改进 A-star 算法搜索的节点数大于 A-star 算法。随着搜索节点向目标位置靠近，改进 A-star 算法的启发函数的权重系数逐渐增大，此时搜索的节点数逐渐减少。

值得注意的是，算法耗时并未随着搜索节点数的减少等比例的减少，这是因为算法搜索节点数越多，在 close_list 节点数越多，close_list 中每增加一个节点都伴随着一次 open_list 中节点的遍历。虽然启发函数中引入了环境的障碍物栅格率，但障碍物最少的场景中，改进 A-star 算法的提升效果反而不明显，这是因为在实际的路径搜索中 A-star 算法的整体性能除了和启发函数的权重系数有关还与障碍物的分布结构有关。如果障碍物总是分布在当前节点朝向目标位置的方向上，且只有向回绕行才能避开环境中的障碍物，其启发函数的权重无论大小均不能引导搜索节点快速向目标位置靠近。从图 9(a)和图 9(b)中算法搜索过的节点可以看出，此时 A-star 算法整体性能与 Dijkstra 算法无异。

表 2　全局路径规划实验结果

Table 2　Experimental results of global path planning

地图场景	算法	搜索节点数	算法耗时/ms	路径长度/m
巷道环境	Dijkstra	2 063	459.57	225.25
	A-star	1 872	427.34	225.25
	改进 A-star	**1 730**	**384.16**	225.25
仓库环境	Dijkstra	1 543	270.29	74.87
	A-star	964	180.83	74.87
	改进 A-star	**680**	**95.1**	74.87
简单迷宫环境	Dijkstra	1 883	421.84	76.77
	A-star	1 158	297.03	76.77
	改进 A-star	**998**	**234.56**	76.77
复杂迷宫环境	Dijkstra	2 804	866.46	94.84
	A-star	1 140	347.55	94.84
	改进 A-star	**994**	**218.81**	94.84

3.2　轨迹优化实验与分析

分段步长 ζ 的值设置太大，优化后的路径在转弯较大的地方会直接穿过障碍物区域；分段步长 ζ 的值设置太小，优化后的路径会由于与原路径的拟合程度太高，导致路径出现频繁的转弯。因此，在 4 个不同的场景中，贝塞尔曲线初始分段步长 ζ 的值通过调试最终都设置为 10。路径优化前后的效果对比如图 13 所示，优化前的路径为图中的折线，优化后的路径为图中平滑的曲线。从图中可以看出，优化前的路径存在较多的转折点，优化后的路径整体较为平滑无明显转折点。虽然在图 13(a)、(b)、(d)中优化后的路径与障碍物区域有少量接触，但在实际的移动机器人导航应用中可以通过设置障碍物的膨胀区域进行避免。

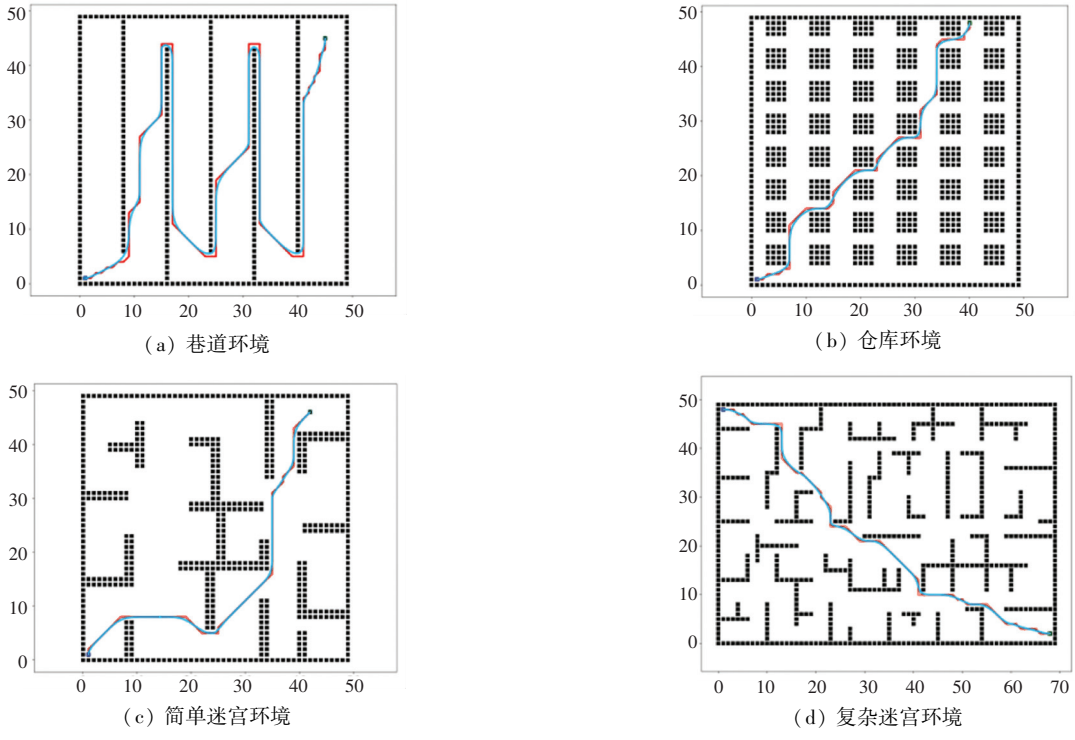


图 13 改进 A-star 算法路径优化效果

Fig. 13 Path planning of improved A-star algorithm

3.3 融合路径规划实验与分析

电脑系统为 64 位 ubuntu20.04,通过 ROS(kinetic)系统和 Gazebo 平台,搭建 3D 仿真环境模拟现实的实验室环境,长 40 m、宽 20 m(图 14)。搭建仿真移动机器人为二轮差分底盘模型,机器人半径为 0.15 m 有两个主动轮,两个从动万向轮,搭载有 10 Hz 的 2D 激光雷达用于感知周围环境。移动机器人最快移动速度 1.5 m/s,最大加速度1 m/s^2,最快旋转速度 1 rad/m,最快旋转加速度1 rad/s^2,线速度分辨率为 0.02,角度分辨率为 0.02,局部路径规划速度采样周期为 2 s。通过 cartographer 算法[21]构建二维栅格地图。为了保证移动机器人在行驶过程中与障碍物保持足够的安全距离,为栅格地图设置膨胀区域半径为 0.15 m(图 15),然后将其用于路径规划。

在全局地图中设置移动机器人的起始位置与目标位置,利用融合路径规划算法进行移动机器人路径规划和导航。在移动机器人向目标位置前进的过程中,在仿真实验室场景中添加未知的障碍物。局部路径规划过程如图 16 所示,当环境中出现未知的障碍物时,通过激光雷达扫描将障碍物信息加入二维栅格地图中,通过局部路径规划算法完成避障,然后再回到全局路径中,继续向目标位置前进。融合路径规划算法下的机器人运动轨迹如图 17,全局最优路径和移动机器人的运动轨迹均较为平滑无明显转折点,且移动机器人能避开环境中的未知障碍物顺利到达目标位置。

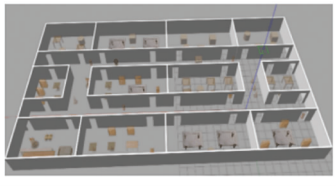


图 14 仿真实验环境

Fig. 14 Simulation environment

图 15 栅格地图

Fig. 15 Grid map

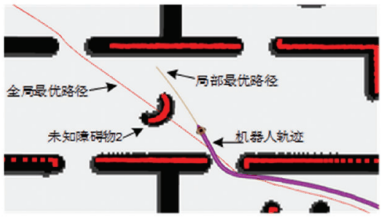


图 16 局部路径规划

Fig. 16 Local path planning

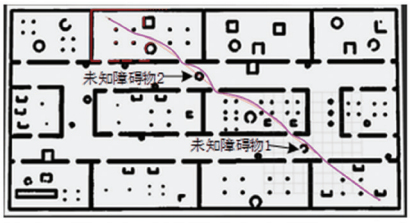


图 17　融合算法路径

Fig. 17　Path planning with fusion algorithm

4　结束语

本文将障碍物栅格率和搜索节点到起始位置的距离信息引入传统 A-star 算法的启发函数，使启发函数的权重信息随着搜索环境和节点位置的变化动态调整，从而减少算法搜索节点的数量，提升算法的速度。利用分段高阶贝塞尔曲线对路径进行优化，使路径平滑无明显转折点。仿真实验结果表明，改进的 A-star 算法能适用于各种不同的应用场景，且相比传统 A-star 算法搜索节点数明显减少，耗时更短。将改进的 A-star 算法搜索全局最优路径中的节点，作为 AWD 算法局部路径规划的中间目标位置实现算法融合。仿真实验结果表明，本文提出的融合路径规划算法能实现移动机器人在复杂环境中的动态路径规划，有效避开环境中未知的障碍物且平稳到达目标位置。

参考文献

[1] OROKO J A, NYAKOE G N. Obstacle avoidance and path planning schemes for autonomous navigation of a mobile robot: a review [C]//Proceedings of the Sustainable Research and Innovation Conference. IEEE, 2022: 314-318.

[2] ZHOU C, HUANG B, FRÄNTI P. A review of motion planning algorithms for intelligent robots[J]. Journal of Intelligent Manufacturing, 2022, 33(2): 387-424.

[3] DIJKSTRA E W. A note on two problems in connexion with graphs[J]. Numerische Mathematik, 1959, 1: 269-271.

[4] ZHOU Y, HUANG N. Airport AGV path optimization model based on ant colony algorithm to optimize Dijkstra algorithm in urban systems [J]. Sustainable Computing: Informatics and Systems, 2022, 35: 100716.

[5] 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.

[6] GUO B, KUANG Z, GUAN J, et al. An improved a-star algorithm for complete coverage path planning of unmanned ships [J]. International Journal of Pattern Recognition and Artificial Intelligence, 2022, 36(3): 2259009.

[7] STENTZ A. Optimal and efficient path planning for partially-known environments[C]//Proceedings of the 1994 IEEE International Conference on Robotics and Automation. IEEE, 1994: 3310-3317.

[8] KOENIG S, LIKHACHEV M, FURCY D. Lifelong planning A* [J]. Artificial Intelligence, 2004, 155(1-2): 93-146.

[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] 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.

[11] PARK M G, JEON J H, LEE M C. Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing[C]//Proceedings of 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No. 01TH8570). IEEE, 2001: 1530-1535.

[12] VOSE M D. The Simple Genetic Algorithm: Foundations and Theory[M]. Cambridge: MIT Press, 1999.

[13] ZAFAR M A, ZHENG Z, WENKAI Y. Mobile robots path planning based on A* algorithm improved with jump point search [C]//Proceedings of 2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST). IEEE, 2021: 536-544.

[14] XIANGRONG T, YUKUN Z, XINXIN J. Improved A-star algorithm for robot path planning in static environment[J]. Journal of Physics: Conference Series IOP Publishing, 2021, 1792(1): 012067.

[15] 卞永明，马逍阳，高飞，等. 基于改进 A-Star 算法的 AGV 全局路径规划[J]. 机电一体化，2019 (6)：9-15.

[16] 谢春丽，高胜寒，孙学志. 融合改进 A* 算法和贝塞尔曲线优化的路径规划算法[J]. 重庆理工大学学报（自然科学），2022，36(7)：177-187.

[17] 张恒瑞，孟海涛. 融合改进人工势场法的 A* 算法优化[J]. 软件工程与应用，2022，11：994.

[18] 槐创锋，郭龙，贾雪艳，等. 改进 A* 算法与动态窗口法的机器人动态路径规划[J]. 计算机工程与应用，2021，57(8)：244-248.

[19] SAEED R A, RECUPERO D R, REMAGNINO P. A boundary node method for path planning of mobile robots[J]. Robotics and Autonomous Systems, 2020, 123: 103320.

[20] CHEN J, STURTEVANT N R. Necessary and sufficient conditions for avoiding reopenings in best first suboptimal search with general bounding functions[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2021: 3688-3696.

[21] HESS W, KOHLER D, RAPP H, et al. Real-time loop closure in 2D LIDAR SLAM [C]//Proceedings of 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2016: 1271-1278.