

## 基于改进 A\* 算法和动态窗口法的移动机器人路径规划<sup>①</sup>

王军晓<sup>②\*</sup> 黄美琴\* 冯建涵\* 汪显博<sup>③\*\*</sup>

(\* 浙江工业大学信息工程学院 杭州 310023)

(\*\* 浙江大学海南研究院 三亚 572025)

**摘要** 在大型场景中,针对传统 A\* 算法存在的内存开销大、搜索时间长以及动态窗口法(dynamic window approach, DWA)容易陷入局部最优、找到的最终路径不是全局最优路径等问题,提出了一种基于融合改进 A\* 算法和 DWA 的混合路径规划方法。首先,将传统 A\* 算法的 24 邻域扩展减少为 10 邻域扩展。其次,引入同步双向搜索策略,并在此基础上提出同步直连的方法,即检查 2 个当前动态定义的目标节点之间是否存在障碍物,若无障碍物,则直接生成最终路径。然后,将改进 A\* 算法生成的全局路径中提取出的关键路径点作为 DWA 的局部目标点,并改进 DWA 的评价函数。仿真和实验结果表明,相比于传统 A\* 算法,改进的 A\* 算法有效地将遍历节点数量减少 51.42%、搜索时间降低 63.32%;改进的 DWA 可以完美避开凹形障碍物并找到全局最优路径。

**关键词** 移动机器人; 路径规划; 改进 A\* 算法; 改进动态窗口法; 栅格地图

近几年来,移动机器人不断拓展新的应用领域,从工业、医疗、安防等传统领域,到家庭、教育、娱乐、公共服务等新兴领域,移动机器人的功能和用途日益丰富。而在移动机器人自主导航领域,路径规划被视为实现该技术的一个关键因素,也是衡量移动机器人智能化水平的一个重要指标。所谓路径规划就是移动机器人根据自身传感器对环境的感知<sup>[1]</sup>,找到一条从起点到终点的安全、可行、高效的途径。关于路径规划的方法有很多,根据机器人对外部地图信息的掌握程度可以分为全局路径规划<sup>[2-4]</sup>和局部路径规划<sup>[5-7]</sup>。全局路径规划需要对整个环境有准确的了解,而局部路径规划则需要实时、精确的环境感知。环境建模为路径规划算法提供了基础框架,它是机器人能否高效、智能地规划路径的关键。

栅格法是环境建模中的一种经典方法,能够将移动机器人的外部环境信息存储在相互连接且大小

相同的对应位置的栅格之中<sup>[8]</sup>,使机器人控制器能够方便地处理、存储、使用和更新信息<sup>[9]</sup>。因此,本文将采用栅格法对环境进行建模。已有很多路径规划算法的设计是以栅格地图作为基础进行导航的,如 A\* 算法<sup>[10]</sup>、动态窗口法(dynamic window approach, DWA)<sup>[11]</sup>等。其中,当外部环境中存在动态未知障碍物时,主要采用动态窗口法;当外部环境中只存在静态已知障碍物时,往往采用 A\* 算法才能够更快且更有效地求出从起点到终点的最短路径<sup>[12]</sup>。

但是 A\* 算法也存在一些问题。A\* 算法在搜索路径时需要使用一个开放列表和一个关闭列表来存储被访问的节点,当搜索场景较大时,需要访问大量的节点,从而消耗大量内存,且搜索效率低。针对这些问题,文献[13]提出了在 A\* 算法中引入双向交替搜索策略,再通过指数衰减对启发式函数进行加权,有效减少了搜索节点的数量,提高了算法的搜

① 国家自然科学基金(62273306)资助项目。

② 男,1986 年生,博士,副教授;研究方向:机电伺服/自主系统抗干扰控制与优化;E-mail: wjx2017@zjut.edu.cn。

③ 通信作者,E-mail: xbwang@zju.edu.cn。

(收稿日期:2024-08-23)

索效率;但它生成的路径不是最优路径,需要进一步对路径节点进行过滤处理,增加了计算量。文献[14]提出了方向约束自适应扩展双向 A\* 算法,利用自适应扩展方法和方向约束的最优节点扩展方法,有效减少了扩展节点和搜索时间;但它只是在仿真环境中进行了大量实验,并没有应用到实际环境中。而动态窗口法也存在容易陷入局部最优解、找不到最终路径,或者找到的最终路径不是全局最优路径等问题。为了解决上述问题,文献[15]提出了参数自适应的动态窗口法,该算法根据障碍物的密集程度动态调整轨迹评价函数中的速度函数权值,提高了算法在障碍物密集环境中的运行效率以及安全避障性能。文献[16]研究了轨迹评价函数中目标方位角函数评价因子和速度函数评价因子之间的矛盾,在评价函数中引入了关于姿态变化的评价因子,有效抑制某些特定情况下某个特定因素对评价函数的过度影响,减少了机器人不必要的转向次数。文献[17]提出了一种基于 Q-learning 的改进动态窗口法。首先改进动态窗口法轨迹评价函数中的每一项取值,然后再增加 2 个新的评价函数,最后提出了一种基于 Q-learning 的自适应算法来调整动态窗口法评价函数的权值,有效提高了该方法在复杂未知环境下的导航效率和成功率。

然而在实际环境中,地图更加复杂且不可能一成不变,随时会出现未知的静态或动态的障碍物。而当环境发生变化时,只依靠全局路径规划方法已经无法有效地避开障碍物,甚至可能发生碰撞,不能保障移动机器人的安全。此时,若选择只利用局部路径规划方法来实现对未知障碍物的避障,往往会导致规划时间长、效率低,规划出来的路径不是全局最优路径。因此,本文在改进 A\* 算法的基础上加入了改进的动态窗口法,将全局路径规划方法与局部路径规划方法融合到一起。混合算法可以有效避开环境中的未知静态或动态障碍物,并且可以避免陷入凹形障碍物中,使规划出来的路径趋近于最优路径。

## 1 改进 A\* 算法

### 1.1 环境模型描述

本文采用栅格法构建地图,将外部地图环境分

割为大小相同且相连的小方块,如图 1 所示。图中的黑色圆圈表示不规则障碍物,为了方便计算,没有占满整个方格的障碍物将其视为占满整个方格,如灰色区域所示;并且为了安全考虑,本文对障碍物进行膨胀处理,膨胀范围为一格,图中绕障碍物一圈的灰色方格即为膨胀区域;没有障碍物的地方不做任何处理,即方格为空白。移动机器人在栅格地图中作为质点处理,不可以直接穿越障碍物,可以在无障碍物区域自由活动。

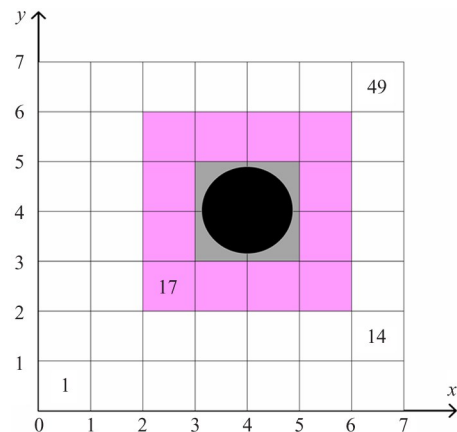


图 1 栅格地图及障碍物

图 1 中每一个方格不仅可以由坐标表示,也可以由索引号表示。例如第 1 个方格的坐标为(1,1),它的索引号为 1。索引号与坐标之间的转换公式如式(1)所示。

$$\begin{cases} x_n = \begin{cases} w & \text{mod}(n, w) = 0 \\ \text{mod}(n, w) & \text{mod}(n, w) \neq 0 \end{cases} \\ y_n = \text{ceil}(\frac{n}{w}) \end{cases} \quad (1)$$

式中: $n$  表示索引, $w$  表示栅格地图宽度, $x_n$  和  $y_n$  分别表示索引  $n$  的  $x$ 、 $y$  坐标, $\text{mod}(n, w)$  表示  $n$  对  $w$  的取余操作, $\text{ceil}$  表示数学运算符向上取整。

### 1.2 传统 A\* 算法

A\* 算法<sup>[10]</sup>是一种用于在图中寻找从一个位置到另一个位置最短路径的算法。因其具有完备性、最优性和最优效率,它被广泛应用于计算机科学的许多领域。A\* 算法的核心思想是使用一个启发式函数来预估节点至目标节点的移动成本,从而指导搜索的方向。为确保 A\* 算法寻得最优解, $h(m)$  须满足特定条件。其代价函数如下:

$$f(m) = g(m) + h(m) \quad (2)$$

式中: $m$  表示路径中的当前节点,  $g(m)$  表示从起点到当前路径节点  $m$  的实际代价,  $h(m)$  表示从当前路径节点  $m$  到目标点的预估代价,  $f(m)$  表示当前路径节点  $m$  的总代价。

A\* 算法的性能依赖于启发式函数的选择,不同的启发式函数适用于不同的问题和地图,常见的启发式函数有曼哈顿距离、欧几里得距离和对角距离等。A\* 算法的一个主要缺点是它需要存储所有生成的节点,这可能占用大量的内存空间。因此,在实际的应用中,有时需要使用一些预处理或记忆限制的方法来优化 A\* 算法。

### 1.3 改进 A\* 算法

传统 A\* 算法采用 4 邻域搜索方式进行扩展,每次扩展从起始点和目标点 2 个方向分别遍历上下左右 4 个邻域,使搜索方向具有局限性,搜索范围小、搜索路径较为曲折。若将 4 邻域扩展为 8 邻域或者 24 邻域,虽然增加了搜索方向使搜索路径更加平滑,但随着搜索范围的增大,内存开销变大、路径规划效率变低。为了解决上述问题,本文在搜索邻域上以及同步双向搜索上对传统 A\* 算法进行改进。

#### 1.3.1 改进搜索邻域

在  $5 \times 5$  的栅格范围内,以 24 邻域搜索为基础,文献[18]提出了根据方向判断和扇形邻域扩展的原则,去除了 9 个冗余扩展方向。借鉴其扩展思想,本文结合当前节点与目标结点的相对方位关系,对候选扩展方向的保留规则进行了重新定义,并进一步给出了扩展方向的筛选条件。

定义当前节点 *current* 的  $3 \times 3$  栅格内的节点为内层, $5 \times 5$  栅格的最外层节点为外层。如图 2 所示。

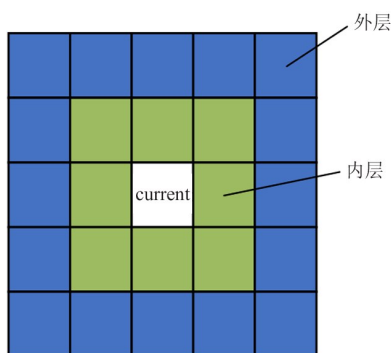


图 2 层级定义

邻域扩展方向如图 3 所示,其中 *current* 表示栅格地图上的任意当前节点, *goal* 表示此节点的目标节点。若将以 *current* 点向 *goal* 点进行搜索视为正向搜索,则 *current* 点扩展方向的筛选条件如下所述。

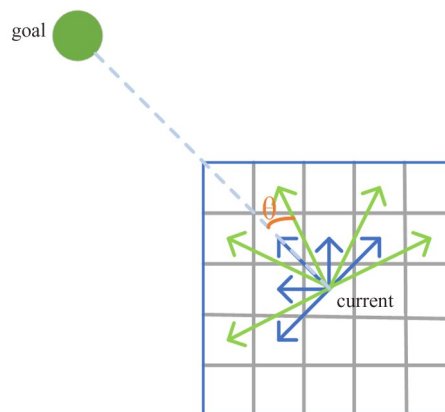


图 3 领域扩展方向

(1) 将 24 邻域扩展减少为 16 邻域扩展,即将内层的 8 个扩展方向保留,将外层 16 个扩展方向中与内层扩展方向相同的 8 个扩展方向删除。

(2) 以 *goal* 点到 *current* 点的连线作为一条边, *current* 点为顶点。假设任意一个扩展方向与已知边的夹角为  $\theta$ , *goal* 点的坐标为  $(x_{goal}, y_{goal})$ , *current* 点的坐标为  $(x_{current}, y_{current})$ , 扩展点的坐标为  $(x_k, y_k)$ , 那么:

$$\cos\theta_h = \frac{\mathbf{CG} \cdot \mathbf{CK}}{|\mathbf{CG}| \cdot |\mathbf{CK}|} \quad (3)$$

$$\theta_d = \frac{180^\circ}{\pi} \times \theta_h \quad (4)$$

式中:  $\mathbf{CG} = (x_{goal} - x_{current}, y_{goal} - y_{current})$  表示点 *goal* 到点 *current* 的向量,  $\mathbf{CK} = (x_k - x_{current}, y_k - y_{current})$  表示任意扩展点  $k$  到点 *current* 的向量。根据式(3)求出 16 个扩展方向  $\mathbf{CK}$  与  $\mathbf{CG}$  的弧度角  $\theta_h$  后,再根据式(4)求出对应的角度  $\theta_d$ 。通过比较  $\theta_d$  角的大小,找出最小的 10 个  $\theta_d$  值,并保留其对应的扩展点。

(3) 对通过筛选条件式(1)和式(2)保留下来的 10 个扩展点进行障碍物判断。若内层 8 个扩展点均不是障碍物,那么外层 8 个扩展点只需要判断自身是否为障碍物。若内层 8 个扩展点中存在障碍物,那么,如图 4 所示,当搜索外层扩展点  $e_4$  时,需

要判断  $d_3$  和  $d_4$  点是否有障碍物,若有障碍物,则删除该扩展点;若无障碍物,则保留该扩展点。同理,每次在搜索其他外层扩展点时,也需要进行如上障碍物判断。

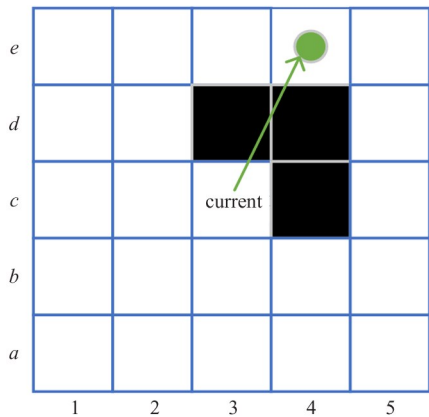


图4 障碍物判断

最终满足上述筛选条件的扩展方向组成的图案类似半个烟花的形状,如图3所示。反向搜索邻域扩展的筛选条件与正向搜索邻域扩展的筛选条件相似,原理同上。

### 1.3.2 同步双向搜索策略

同步双向搜索是指从给定的起点和终点同时采用A\* 算法进行搜索,并且称起点以终点为目标的搜索为正向搜索,终点以起点为目标的搜索为反向搜索。除此之外,同步双向搜索策略还动态定义正向搜索和反方向搜索的起点和终点,即将正向搜索的 openList1 中评价函数值最小的节点设为正向搜索的当前节点,也就是当前时刻正向搜索的起点,同时也将其设为反向搜索的目标节点。将反向搜索的 openList2 中评价函数值最小的节点设为反向搜索的当前节点,即此次循环中反向搜索的起点,同时也将其设为正向搜索的目标节点。每次进入循环,目标节点都将更新,从而使得每次正、反向搜索的起点和终点都不一样,直到正向搜索的当前节点的扩展节点与反向搜索的当前节点的扩展节点相遇时,循环结束。此时查找路径成功。

### 1.3.3 同步直连方法

本文结合同步双向搜索策略提出同步直连的方法,所谓同步直连就是当正向搜索的当前节点与反向搜索的当前节点之间的直接连线上不存在障碍物

时,判定已经找到从起点到终点的最终路径。不同于传统A\* 算法每次寻路只能规划一小步,改进A\* 算法通过直接连线2个当前节点,实现跨越较大的距离,直接找到最终路径。这有效降低了寻路过程中访问的节点数量,减少了内存开销。

改进A\* 算法就是在上一次循环即将结束且下一次循环开始之前,加入同步直连的判断方法。根据正向搜索的当前节点与反向搜索的当前节点之间是否有障碍物,分以下2种情况进行讨论。

#### 案例1: 2个当前节点之间不存在障碍物

如图5所示,current 1表示正向搜索的当前节点,current 2表示反向搜索的当前节点。点current 1和点current 2之间的连线称为线段  $c1c2$ ,将线段  $c1c2$  分成  $n$  个点,检查每一个点的8个邻居节点  $n_i\_neighbor$  和该点本身  $n_i$  是否为障碍物。

如果2个当前节点之间不存在障碍物,如图6所示,那么线段  $c1c2$  上的每一个  $n_i$  以及  $n_i\_neighbor$  均不是障碍物,此时寻路成功,循环终止,找到从起点到终点的路径。

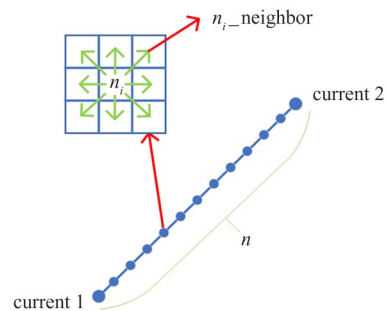


图5 每一个点的检查范围

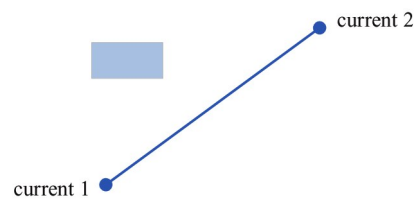


图6 2个节点之间不存在障碍物点

#### 案例2: 2个当前节点之间存在障碍物

如果2个当前节点之间的线段  $c1c2$  上存在障碍物,如图7(a)所示,那么线段  $c1c2$  上的某一个或多个节点是障碍物,此时寻路失败,进入下一次循环。

如果 2 个当前节点之间的线段  $c_1c_2$  上不存在障碍物,如图 7(b)所示,但障碍物距离线段  $c_1c_2$  很近,且某一个或多个点的邻居节点存在障碍物,此时寻路也失败,进入下一次循环。

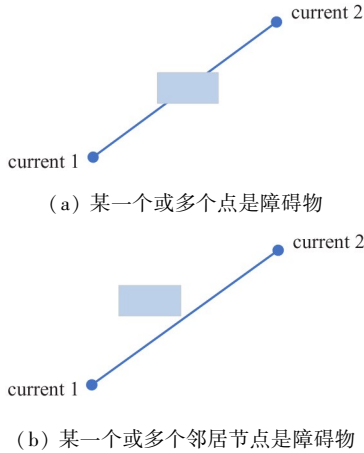


图 7 2 个节点之间存在障碍物点

#### 1.3.4 平滑路径

由于本文是在栅格地图上采用 A\* 算法进行路径规划,并且将 24 个可扩展方向优化为 10 个可扩展方向。因此其产生的最终路径在通过狭窄通道时可能会产生尖锐转折角,所以本文将采用以下方法来减小转折角度、缩短最终路径,具体如下。

(1) 首先将规划出来的路径点都存在一个数组中,例如:  $\{A, S_1, S_2, \dots, S_n, B\}$ , 其中  $A$  是起点,  $B$  是终点,  $S_1, S_2, \dots, S_n$  是路径上经过的其他点。

(2) 从起点  $A$  开始,连接  $A$  到  $S_1$ , 检查  $AS_1$  连线上是否存在障碍物,若不存在,则继续连接  $A$  到  $S_2$ , 继续检查  $AS_2$  连线上是否存在障碍物,以此类推。直到连接  $A$  到  $S_m$  且发现  $AS_m$  连线上存在障碍物时,说明从起点  $A$  到  $S_{m-1}$  点之间不存在障碍物,可以直接到达,此时只保留  $A$  点和  $S_{m-1}$  点,其余中间的  $S_1, S_2, \dots, S_{m-2}$  点都删除。

(3) 接下来从  $S_{m-1}$  点开始,连接  $S_{m-1}$  到  $S_m$ , 转到(2)继续进行后续检查;检查完所有的节点后,将终点  $B$  也加入数组中。

(4) 若从起点  $A$  到终点  $B$  之间都不存在障碍物,则数组中只保留  $A, B$  2 点,其余所有路径节点都删除。

## 2 改进动态窗口法

### 2.1 移动机器人速度采样

动态窗口方法将路径规划问题转化为速度空间的约束优化问题,通过输出实时的最优速度来控制机器人的运动。由于速度矢量空间中存在多种速度组合,因此有必要从以下 3 个方面进行限制,以降低速度采样的复杂度。

#### (1) 运动学约束

由于移动机器人受其自身最大线速度、角速度和最小线速度、角速度的限制,采样速度  $V_s$  被限制如下。

$$V_s = \{(v, \omega), v \in [v_{\min}, v_{\max}] \wedge \omega \in [\omega_{\min}, \omega_{\max}]\} \quad (5)$$

式中:  $v_{\max}$  和  $\omega_{\max}$  是机器人的最大线速度和最大角速度,  $v_{\min}$  和  $\omega_{\min}$  是机器人的最小线速度和最小角速度。

#### (2) 动力学约束

由于电机性能的影响,移动机器人的最大加减速速度受到电机转矩的限制。因此,在移动机器人一个模拟周期  $\Delta t$  期间,其实际速度集合  $V_d$  为

$$V_d = \{(v, \omega) \mid v \in [v_c - \dot{v}_d \Delta t, v_c + \dot{v}_d \Delta t] \wedge \omega \in [\omega_c - \dot{\omega}_d \Delta t, \omega_c + \dot{\omega}_d \Delta t]\} \quad (6)$$

式中:  $v_c$  和  $\omega_c$  分别是移动机器人当前线速度和当前角速度,  $\dot{v}_d$  和  $\dot{\omega}_d$  是相应的最大线性减速度和最大线性加速度,  $\dot{\omega}_d$  和  $\dot{\omega}_a$  是相应的最大角减速度和最大角加速度。

#### (3) 安全性约束

在实际应用中,基于移动机器人的安全考虑,为了使移动机器人能够在制动后不与障碍物发生碰撞,移动机器人的速度  $V_a$  被限制在最大范围内,该最大范围可以表示为

$$V_a = \{(v, \omega) \mid v \leq \sqrt{2 \cdot \text{dist}(v, \omega) \cdot \dot{v}_d} \wedge \omega \leq \sqrt{2 \cdot \text{dist}(v, \omega) \cdot \dot{\omega}_d}\} \quad (7)$$

式中:  $\text{dist}(v, \omega)$  是当前速度矢量  $(v, \omega)$  对应的轨迹与最近障碍物的距离。

生成的较小速度矢量空间  $V_r$  由上述 3 个速度

产生,表示如式(8)所示。

$$V_r = V_s \cap V_d \cap V_a \quad (8)$$

## 2.2 轨迹评价函数

在动态窗口法中,轨迹评价函数是用于评估每一组速度组合  $(v, \omega)$  好坏的关键工具。轨迹评价函数通常综合考虑多个因素,包括目标方向评分、障碍物距离评分和速度评分,以便在实现避障的同时尽可能快速安全地接近目标。评价函数的形式如下所示。

$$G(v, \omega) = \varepsilon(\alpha \cdot \text{heading}(v, \omega)) + \beta \cdot \text{dist}(v, \omega) + \gamma \cdot \text{vel}(v, \omega) \quad (9)$$

式中:  $\alpha, \beta, \gamma$  分别表示评价函数的目标方向角函数  $\text{heading}(v, \omega)$ 、障碍物距离函数  $\text{dist}(v, \omega)$  和速度函数  $\text{vel}(v, \omega)$  对应的权重系数,取值范围为  $[0, 1]$ ;  $\varepsilon$  表示对上述3个部分进行归一化处理。

## 2.3 融合改进A\*算法和DWA算法

本文将改进的A\*算法与DWA算法相融合,提取改进A\*算法生成的全局路径中的关键节点,将这些提取到的关键路径点作为DWA算法的临时目标点,使局部路径规划遵循全局路径规划的轮廓。通过将两者相结合,移动机器人能够在路径规划过程中避开未知静态或动态障碍物,同时不与已知静态障碍物碰撞,并且可以避免陷入局部最优,绕开凹形障碍物。将A\*算法与DWA算法融合后的轨迹评价函数如式(10)所示。

$$G(v, \omega) = \varepsilon(\alpha \cdot \text{PointHeading}(v, \omega) + \beta \cdot \text{dist}(v, \omega) + \gamma \cdot \text{vel}(v, \omega)) \quad (10)$$

式中:  $\text{PointHeading}(v, \omega)$  是在当前速度  $(v, \omega)$  模拟轨迹末端朝向与阶段性目标点之间的夹角  $\theta$ , 阶段性目标点是指改进A\*算法规划出来的全局最优路径中提取出的关键路径点。如图8所示,五角星为移动机器人起点,笑脸为目标点,实线表示生成的全局最优路径,圆圈表示提取出的关键路径点,  $\theta$  为移动机器人当前模拟轨迹末端与关键路径点之间的夹角。

## 2.4 改进DWA算法的评价函数

为了使移动机器人能更精确地到达目标点,且

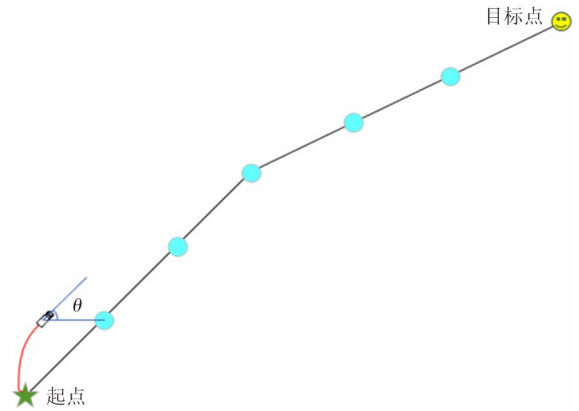


图8 移动机器人的阶段性目标点

以更平稳、尽可能快的速度行驶,本文在引入全局路径规划为局部路径规划作参考的基础上,综合考虑方向角和距离的效应,对目标方向角函数进行优化。除此之外,再添加一个新的子函数,计算在一个模拟周期内从预测轨迹端点到目标点的最小距离。增加该函数的目的是用于评估预测轨迹接近目标点的程度,以增强移动机器人向目标点的运动趋势,特别是当目标点周围有障碍物时的复杂情况。改进后的动态窗口法的轨迹评价函数如式(11)所示。

$$G(v, \omega) = \text{DH}(v, \omega) \cdot (\varepsilon(\alpha \cdot \text{PointHeading}(v, \omega))) + \varepsilon(\beta \cdot \text{dist}(v, \omega)) + \varepsilon(\gamma \cdot \text{vel}(v, \omega)) + \varepsilon(\varphi \cdot \text{goal\_dist}(v, \omega)) \quad (11)$$

式中:  $\text{DH}(v, \omega)$  表示距离系数,具体公式为

$$\text{DH}(v, \omega) = d(v, \omega) / D_{\max} \quad (12)$$

式中:  $d(v, \omega)$  表示速度对应的轨迹终点与当前目标位置之间的距离,  $D_{\max}$  表示从上一个目标位置到当前目标位置的距离。如图9所示,当移动机器人在位置1时,  $D_{\max}$  表示从起点到第1个关键路径点之间的距离,  $d(v, \omega)$  表示从移动机器人当前轨迹末端到第1个关键路径点之间的距离;当移动机器人在位置2时,  $D_{\max}$  表示从第2个关键路径点到第3个关键路径点之间的距离,  $d(v, \omega)$  表示从移动机器人当前轨迹末端到第3个关键路径点之间的距离。

式(11)中,  $\varphi$  为  $\text{goal\_dist}(v, \omega)$  函数的系数;  $\text{goal\_dist}(v, \omega)$  为目标点距离函数,具体公式为

$$\text{goal\_dist}(v, \omega) = \frac{\text{dist\_stgo} - \text{dist\_go}}{\text{dist\_stgo}} \quad (13)$$

式中:  $dist\_stgo$  表示从起点到目标点的距离,  $dist\_go$  表示在一个模拟周期内移动机器人预测轨迹末端到目标点的距离,如图 9 所示。可以看出,当预测轨迹端点距离目标点越近,  $goal\_dist(v,\omega)$  函数得分越高。但是,  $goal\_dist(v,\omega)$  函数不是从起点开始就起作用的,而是需要满足一定条件。只有当移动机器人的当前位置与目标点之间的距离小于 2 时,  $goal\_dist(v,\omega)$  函数才开始工作,有效帮助移

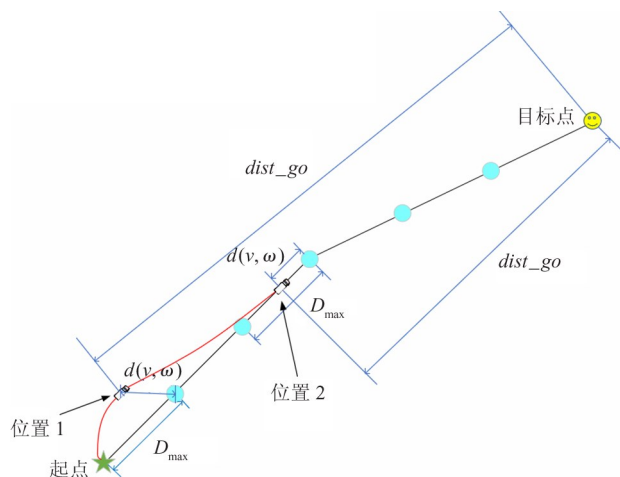


图 9 移动机器人与阶段性目标点以及终点的距离

动机器人以稳定的速度转向目标。

### 3 仿真结果与分析

本文为了验证改进算法的有效性,下面将借助 Matlab,在 CPU 为 i7-8550U、系统为 Windows 10 的电脑上进行仿真。

#### 3.1 改进 A\* 算法仿真对比分析

##### 仿真 1:简单环境

如图 10 所示,在  $30 \times 30$  的栅格地图上进行仿真实验,其中五角星表示起点,坐标为 (3,2),圆圈表示终点,坐标为 (29,29)。在寻路算法执行过程中,被探索过一次的节点由格子标识,方格表示算法在寻路过程中曾多次搜索的节点,折线表示最终生成的路径。由仿真结果可以看出,同样的地图环境中,传统 A\* 算法访问的节点数量最多,如图 10(a) 所示;基于半烟花形邻域扩展的同步直连的双向 A\* 算法访问的节点数量最少,且生成的最终路径的转折点数也最少,路径更加平滑,如图 10(c) 所示。

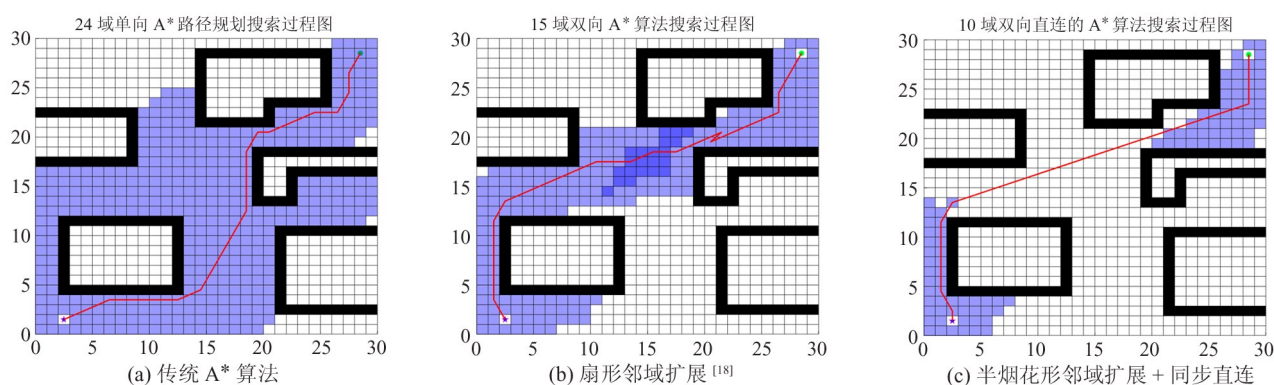


图 10 简单环境下传统 A\* 和改进 A\* 算法的仿真结果

由表 1 可以看出,传统 A\* 算法的寻路时间最长,搜索节点个数最多;本文提出的半烟花形邻域扩展法生成的最终路径与扇形邻域扩展法生成的最终路径长度相同,寻路时间减少 21.40%,搜索节点个数减少 18.50%;当将半烟花形邻域扩展法和同步直连法结合到一起时,寻路时间最短,搜索节点数最少,转折点数量也最少,与传统 A\* 算法相比,寻路时间减少了 85.00%,搜索节点数减少了 76.00%。

仿真结果表明,在障碍物比较聚集的简单环境

中,改进后的 A\* 算法不仅大大减少了搜索时间,提高了算法效率,而且还有效地减少了寻路过程中搜索节点的数量,大大降低了对内存的消耗,最终生成的路径转折角更小,路径也更加平滑美观。

##### 仿真 2:随机复杂环境

在  $30 \times 30$  栅格环境中设置障碍物密度为 0.4,随机生成障碍物地图如图 11 所示。由图 11(a) 可以看出,传统 A\* 算法访问的节点数最多;观察图 11(c)和图 11(d)可知:基于半烟花形邻域扩展

表 1 相同环境中改进 A\* 算法与传统 A\* 算法路径规划结果对比

算法	寻路时间/s	路径长度/m	搜索节点/个
传统 A* 算法	6.919 3	43.832 8	420
扇形邻域扩展 <sup>[18]</sup>	2.800 5	47.719 2	259
半烟花形邻域扩展	2.200 5	47.719 2	211
同步直连	2.386 2	44.985 8	231
扇形扩展 + 同步直连	1.639 6	47.358 5	159
半烟花形扩展 + 同步直连	1.040 3	45.328 9	101

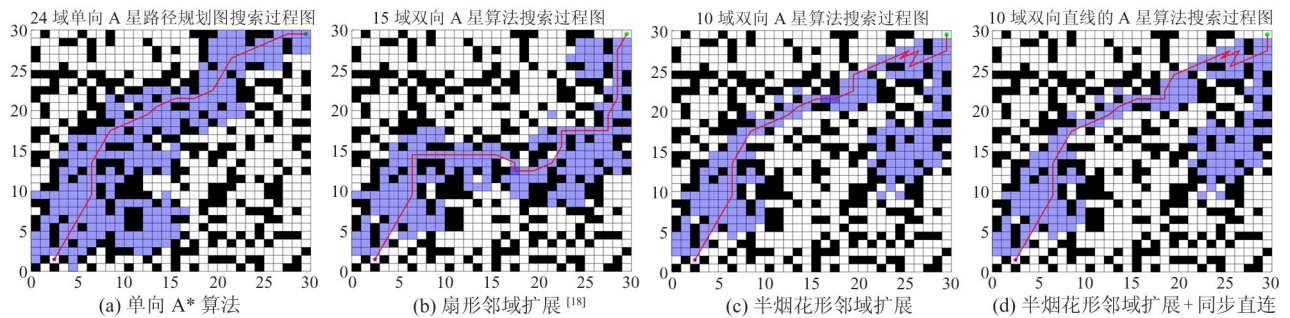


图 11 随机复杂环境下传统 A\* 和改进 A\* 算法的仿真结果

的同步直连的双向 A\* 算法与基于半烟花形邻域扩展的双向 A\* 算法,无论是生成的最终路径还是访问的节点数都大致相同。

由表2可以看出,基于半烟花形邻域扩展的同

步直连的双向 A\* 算法搜索节点数只比基于半烟花形邻域扩展的双向 A\* 算法减少了 3 个节点,寻路时间也只减少了 0.035 7 s;然而比传统 A\* 算法寻路时间短了 41.70%,搜索节点数少了 26.90%。

表 2 复杂环境中改进 A\* 算法与传统 A\* 算法路径规划结果对比(多次实验取平均值)

算法	寻路时间/s	路径长度/m	搜索节点/个
传统 A* 算法	4.201 2	42.955 2	253
扇形邻域扩展 <sup>[18]</sup>	2.689 8	52.124 6	214
半烟花形邻域扩展	2.486 7	50.841 6	188
同步直连	2.913 2	42.955 2	220
扇形扩展 + 同步直连	2.689 8	52.124 6	214
半烟花形扩展 + 同步直连	2.451 0	50.841 6	185

由仿真结果可知,在障碍物分散且复杂的随机环境中,同步直连法几乎不能起到优化作用。而基于半烟花形邻域扩展的双向 A\* 算法可以有效地减少搜索时间和访问节点数量,提高了算法的寻路效率,但是增加了路径长度,寻到的最终路径不是最优路径。因此,在复杂环境中,本文对最终生成的路径进行平滑处理,优化后的路径如图 12(b)所示,可以看出:优化后的路径转折角度大大减少,优化前的路径

长度为 50.841 6,优化后的路径长度为 44.741 3,路径长度减短了 12.00%。

### 3.2 改进 DWA 算法仿真对比分析

如图 13 所示,在 20 × 20 的栅格地图上,设置移动机器人的起点坐标为 (1, 1), 终点坐标为 (20, 20)。黑色方块表示已知的静态障碍物,灰色方块表示未知的静态障碍物,实线表示动态障碍物的运行轨迹,虚线表示利用改进 A\* 算法规划出来的全

局最优路径。

路径规划结果如图 14 所示,实线表示局部路径规划的结果,虚线上的星号表示提取出来的全局路径关键节点。可以看出,3 种算法都能在较复杂的环境中避开未知的静态或动态障碍物。但是,

传统 DWA 算法生成的路径比较曲折且几乎不与全局路径重合,如图 14(a) 所示;而融合 A\* 算法与改进 DWA 算法和融合 A\* 算法与传统 DWA 算法生成的路径大致相同,都比较平滑,并且基本上与全局路径重合,如图 14(b) 和 14(c) 所示。

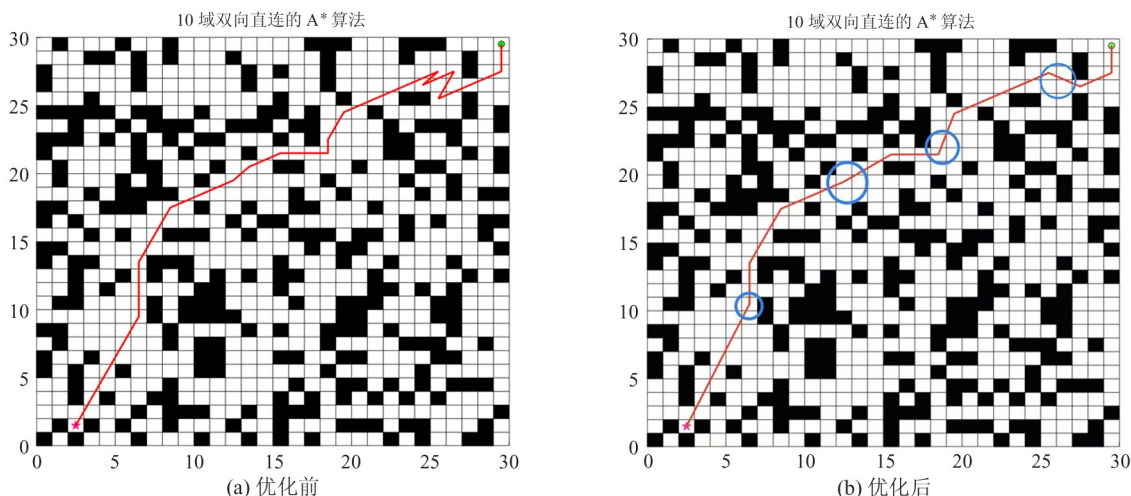


图 12 优化改进 A\* 算法生成的路径

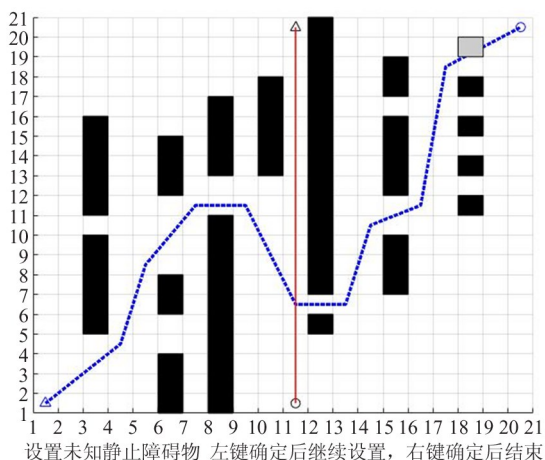


图 13 复杂随机地图以及全局最优路径

线速度和角速度变化情况如图 15 所示。在迭代次数为 100 ~ 400 时,传统 DWA 算法的角速度波动幅度明显比 2 种融合算法的角速度波动幅度大,并且在迭代次数为 300 ~ 310 时,传统 DWA 算法的线速度也发生了一些变化,如图 15(a) 所示。但 2 种融合算法的线速度并没有发生变化,如图 15(b) 和图 15(c) 所示。在迭代次数为 500 ~ 830 时,传统 DWA 算法的角速度变化幅度更大,这是因为此时移动机器人所处的位置旁有一些离目标点较近的分散

障碍物;而 2 种融合算法由于已知全局静态障碍物的分布,因此角速度的波动幅度相对较小。在迭代次数为 830 ~ 950 时,融合 A\* 算法与传统 DWA 算法的线速度开始减小,传统 DWA 算法和融合 A\* 算法与改进 DWA 算法的线速度没有发生改变。在迭代次数为 900 位置时,融合 A\* 算法与改进 DWA 算法到达目标点,停止运动;而传统 DWA 算法和融合 A\* 算法与传统 DWA 算法仍继续前进。

姿态角度变化如图 16 所示。可以看出,传统 DWA 算法的姿态角度变化幅度比融合 A\* 算法与改进 DWA 算法波动幅度大,尤其是在迭代次数为 100 ~ 250 和 500 ~ 800 时。

具体仿真数据如表 3 所示。可以看出,传统 DWA 算法的寻路时间和路径长度都是最长的;融合 A\* 算法与传统 DWA 算法的路径长度是最短的;融合 A\* 算法与改进 DWA 算法的寻路时间最短,比传统 DWA 算法减少 15.21%,比融合 A\* 算法与传统 DWA 算法减少 12.45%。

上述仿真验证了改进 DWA 算法的有效性和可行性,并且改进 DWA 算法对环境的适用性也更强。

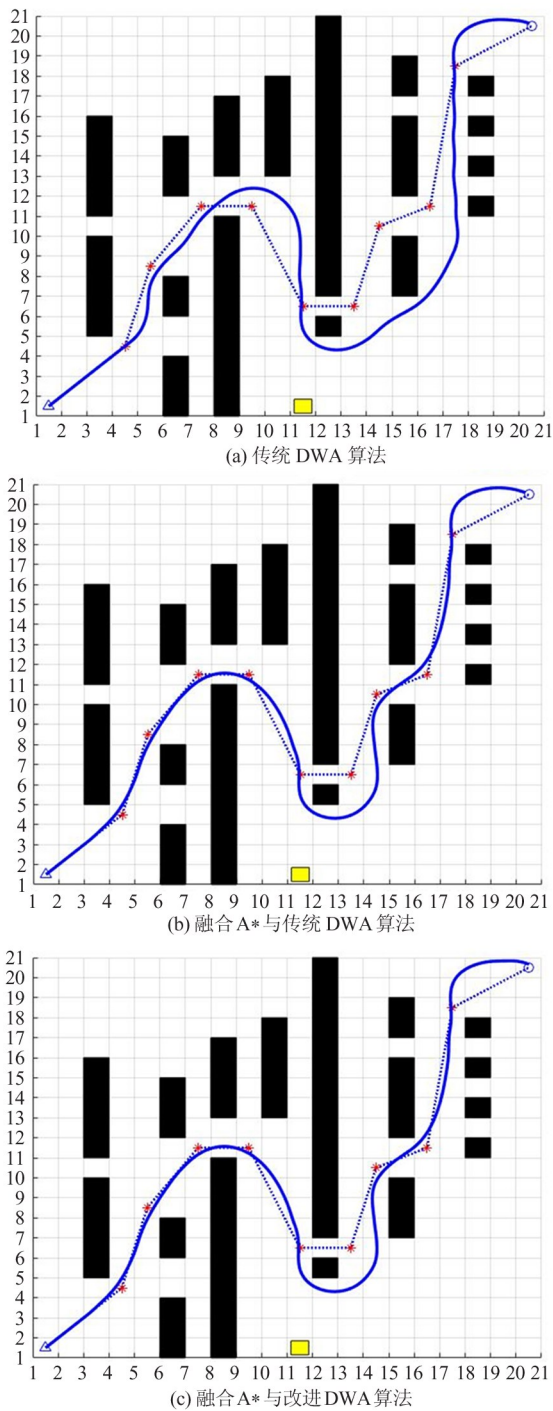


图 14 传统 DWA 算法和改进算法在复杂随机环境中的仿真结果

表 3 传统 DWA 算法和改进算法在复杂随机环境中的路径规划结果对比

算法	路径长度/m	寻路时间/s
传统 DWA 算法	44.959 3	203.179
融合 A* 与传统 DWA 算法	42.618 4	196.756
融合 A* 与改进 DWA 算法	42.673 7	172.270

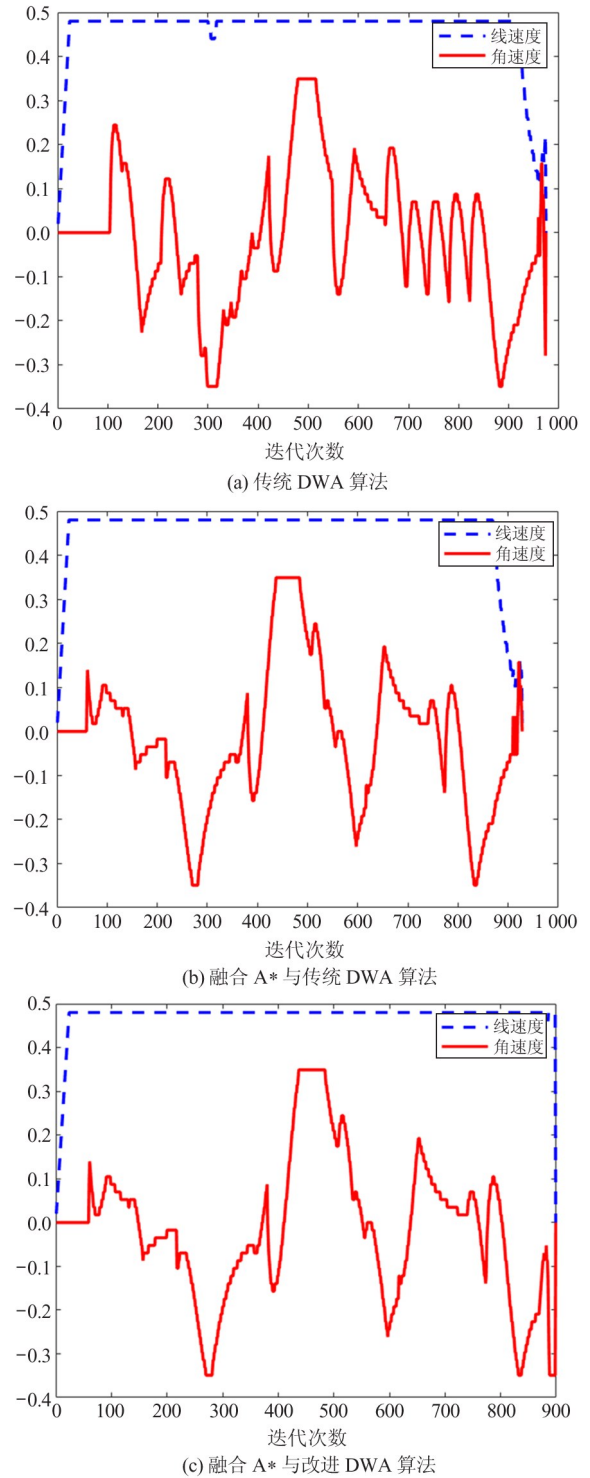


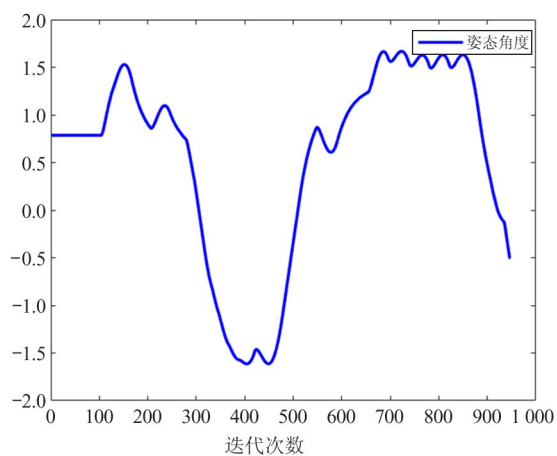
图 15 传统 DWA 算法和改进算法在复杂随机环境中的线速度和角速度

## 4 实验验证

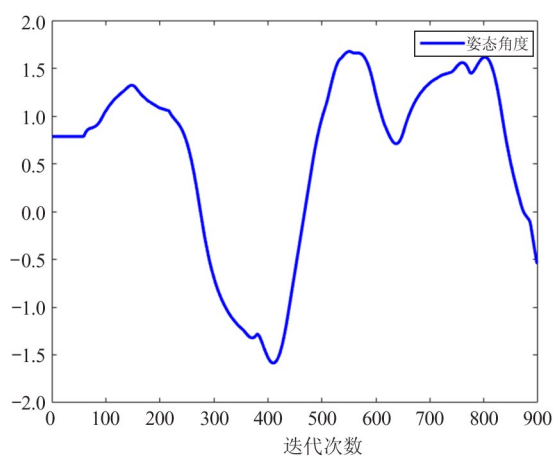
为了验证改进算法在实际应用中的有效性,将改进算法用在基于机器人操作系统的移动机器

人 Turtlebot3-Burger 上,如图 17 所示。该机器人采用 LDS-02 二维激光雷达扫描外部环境,获得地图信息,然后利用 Gmapping 建图方式构建 2 维栅格地图,再利

用 amcl 模块完成对移动机器人的实时定位,最后通过利用 move\_base 功能包中的 global\_planner 路径规划器实现移动机器人的全局路径规划。



(a) 传统 DWA 算法



(b) 融合 A\* 与改进 DWA 算法

图 16 传统 DWA 算法和改进算法在复杂随机环境中的姿态角度

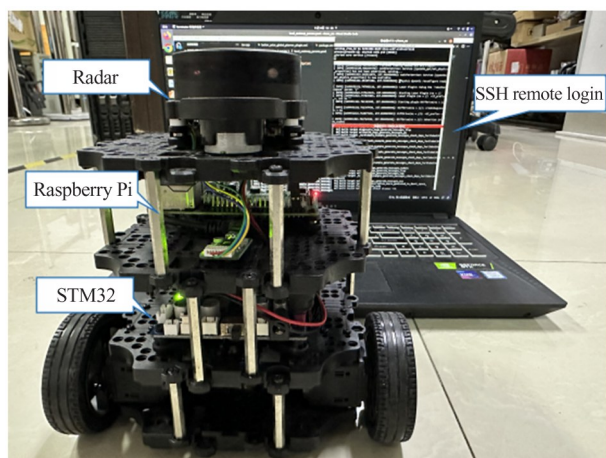


图 17 Turtlebot3-Burger 移动机器人

终点位置和朝向,曲线表示改进 A\* 算法最终生成的路径。



图 18 实验场景

### 实验 1: 静态环境

本文中的实验场景如图 18 所示,在实验室的楼道内搭建静态实验环境,通过远程操作来操控 Turtlebot3-Burger 机器人构建二维地图,然后将 A\* 算法和改进后的 A\* 算法分别在该地图上进行实验,通过分析实验数据的结果,进而证实改进后的算法效果。

图 19 为基于半烟花形邻域扩展的同步直连的双向 A\* 算法路径规划结果图,图中的黑色区域为障碍物,其他区域为障碍物的膨胀范围,小车的位置为设定的起点位置,粗箭头的位置和方向为设定的

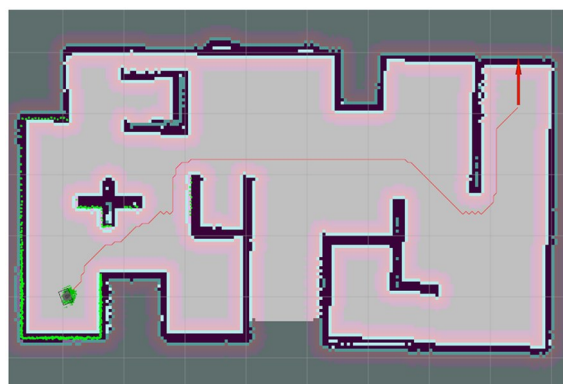


图 19 改进双向 A\* 算法的路径规划结果

实验结果如表4所示,可以看出,在同一地图上A\*算法和改进A\*算法的路径长度大致一样,且改进A\*算法的搜索节点数比传统A\*算法减少了97.67%,寻路时间减少了82.76%。因此,可以说明本文提出的改进A\*算法能够生成最优路径,且寻路速度更快,占用内存更小。

表4 同一地图上2种算法实验结果对比

	传统 A* 算法	改进 A* 算法
路径长度/m	10.985	10.829
搜索节点/个	5 151	120
寻路时间/s	53.022	9.141

实验2:动态环境

动态实验场景是在静态凹形障碍物中不仅加入

静态未知障碍物,还加入动态未知障碍物。实验中的动态障碍物均由一辆移动小车代替。

动态实验场景的实验过程如图20所示。首先移动机器人从起点出发,规划出全局最优路径,如图20(a)所示。然后移动机器人遇到了动态障碍物开始转向,并利用局部路径规划算法重新规划路径,由图20(b)和20(c)可以看出,移动机器人开始发现小车时,先向左前方行驶躲避障碍,一段时间后发现继续前进可能会发生碰撞,立即开始向右前方行驶,最终完美避开动态障碍物。接着移动机器人识别到前方出现了未知障碍物,再次重新规划路径直至避开静态障碍物回到全局路径上,如图20(d)和20(e)所示。最后成功到达目标点,如图20(f)所示。

在上述实验过程中,移动机器人的线速度和角速度随时间的变化情况如图21所示。图中的虚线

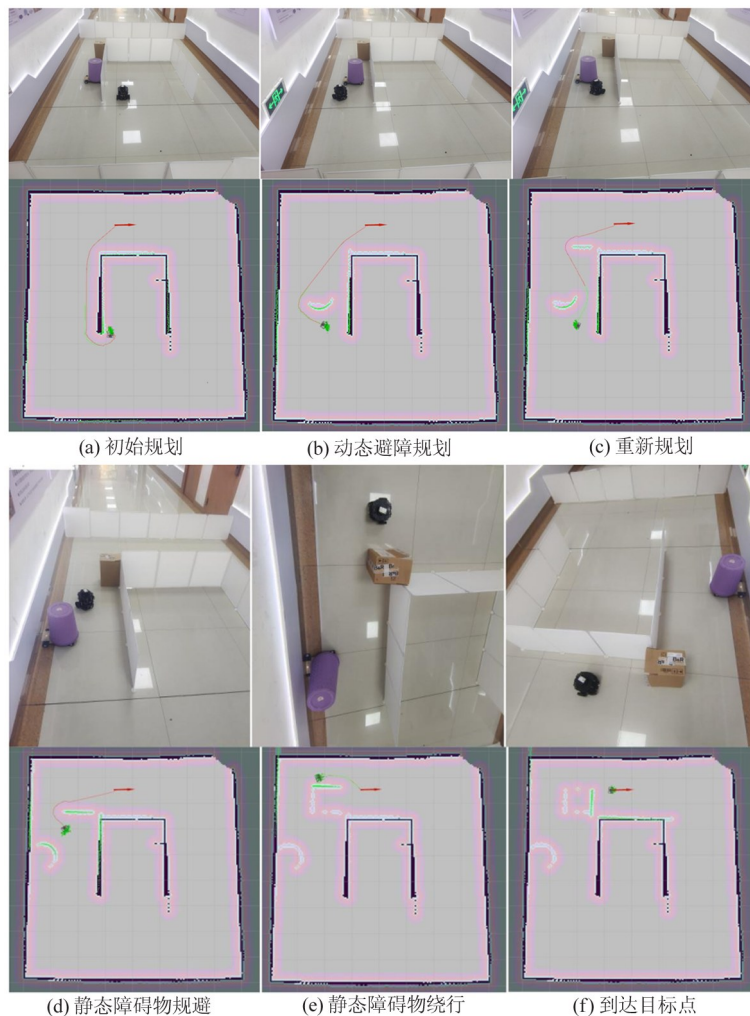


图20 动态实验场景

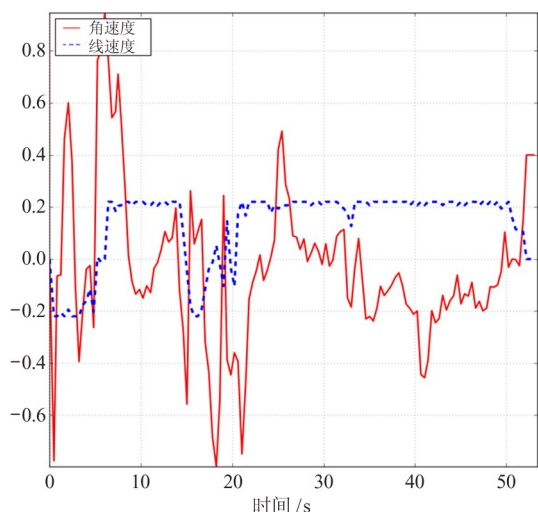


图 21 动态场景中的线速度、角速度变化图

表示线速度变化情况,实线表示角速度变化情况。可以看出,当环境中出现动态障碍物时,移动机器人的角速度波动较大,且线速度也会受到动态障碍物的影响;当前方只存在静态未知障碍物时,线速度几乎不受影响。

通过上述实验表明,混合路径规划方法能够有效地解决移动机器人陷入局部最优解、无法找到目标点的问题,并且可以避开环境中出现的未知静态或动态障碍物,满足移动机器人的自主导航要求。

## 5 结论

为了改善A\*算法在较大场景中寻找路径时,存在的内存开销大、搜索时间长以及DWA算法容易陷入局部最优、找不到最终路径或生成的最终路径不是最优路径等问题,本文提出了一种基于融合改进A\*算法和改进DWA算法的混合路径规划方法。该算法首先通过对每一个当前节点的邻居节点进行筛选,将24邻域扩展变为10邻域扩展,从而达到减少搜索节点数量的目的;然后引入同步双向搜索策略,同时提出同步直连的方法搜索最终路径,从而提高搜索效率。接着从改进A\*算法生成的全局路径中获取关键路径节点,并将这些关键节点作为DWA算法的局部目标点。最后改进DWA算法的评价函数,其一是给目标方向角函数增加一个距离系数,其二是在轨迹评价函数中增加一个目标距离子函数。通过对改进A\*算法和改进DWA算法的多组仿真

和实验表明,本文提出的改进算法满足自主导航要求,将寻路节点减少了51.42%、搜索时间减少了63.32%,展示了改进算法的可行性和有效性。

## 参考文献

- [1] 朱大奇,颜明重. 移动机器人路径规划技术综述[J]. 控制与决策, 2010,25(7):961-967.
- [2] Tu H, Deng Y, Li Q, et al. Improved RRT global path planning algorithm based on bridge test[J]. Robotics and Autonomous Systems, 2024,171:104570.
- [3] Pang R, Zhang C, Sheng X, et al. Global path planning with lifetime constraint model-based offline optimized loading strategy for vehicle fuel cell system[J]. Applied Energy, 2023, 347:121401.
- [4] Mac T T, Copot C, Tran D T, et al. A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization[J]. Applied Soft Computing, 2017,59:68-76.
- [5] Tagor H, Habibullah H, Rafiqul I, et al. Local path planning for autonomous mobile robots by integrating modified dynamic-window approach and improved follow the gap method[J]. Journal of Field Robotics, 2021,39(4):371-386.
- [6] Yao M, Deng H, Feng X, et al. Improved dynamic windows approach based on energy consumption management and fuzzy logic control for local path planning of mobile robots[J]. Computers and Industrial Engineering, 2024, 187:109767.
- [7] Lavelle S M. Rapidly-exploring random trees: a new tool for path planning[J]. Research Report,1998,98:1-10.
- [8] 王洪斌,尹鹏衡,郑维,等. 基于改进的A\*算法与动态窗口法的移动机器人路径规划[J]. 机器人, 2020, 42(3):346-353.
- [9] 王殿君. 基于改进A\*算法的室内移动机器人路径规划[J]. 清华大学学报(自然科学版), 2012,52(8):1085-1089.
- [10] 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.
- [11] Fox D, Burgard W, Thrun S. The dynamic window approach to collision avoidance[J]. IEEE Robotics and Au-

- tomation Magazine, 1997,4(1):23-33.
- [12] 辛煜,梁华为,杜明博,等. 一种可搜索无限个邻域的改进 A\* 算法[J]. 机器人,2014,36(5):627-633.
- [13] Li C, Huang X, Ding J, et al. Global path planning based on a bidirectional alternating search A\* algorithm for mobile robots[J]. Computers and Industrial Engineering, 2022,168:108123.
- [14] Chen J, Li M, Su Y, et al. Direction constraints adaptive extended bidirectional A\* algorithm based on random two-dimensional map environments[J]. Robotics and Autonomous Systems, 2023,165(C):104430.
- [15] 王永雄,田永永,李璇,等. 穿越稠密障碍物的自适应动态窗口法[J]. 控制与决策, 2019,34(5):927-936.
- [16] Liu T, Yan R, Wei G, et al. Local path planning algorithm for blind-guiding robot based on improved DWA algorithm[C]//2019 Chinese Control and Decision Conference. Nanchang, China: IEEE, 2019:6169-6173.
- [17] Chang L, Shan L, Jiang C, et al. Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment[J]. Autonomous Robots, 2021,45(1):51-76.
- [18] 陈万通,刁天茹,贾吉庆,等. 基于扇形领域扩展的同步双向 A\* 算法[J]. 计算机应用研究, 2022,39(1):118-122,127.

## Path planning of mobile robot based on improved A\* algorithm and dynamic window approach algorithm

Wang Junxiao<sup>\*</sup>, Huang Meiqin<sup>\*</sup>, Feng Jianhan<sup>\*</sup>, Wang Xianbo<sup>\*\*</sup>

(<sup>\*</sup> College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023)

(<sup>\*\*</sup> Hainan Institute, Zhejiang University, Sanya 572025)

### Abstract

In large scenarios, a hybrid path planning method based on the fusion of the improved A\* algorithm and the dynamic window approach (DWA) is proposed in response to the problems of the traditional A\* algorithm, such as the large memory overhead, the long search time, and the ease of DWA to fall into the local optimum, and the final path found is not a globally optimal path. Firstly, the 24-neighbourhood expansion of the traditional A\* algorithm is reduced to 10-neighbourhood expansion. Secondly, a synchronous bidirectional search strategy is introduced, and based on this, a synchronous direct connection method is proposed to check whether there is an obstacle between two current dynamically defined target nodes, and if there is no obstacle, the final path is generated directly. Then, the critical path points extracted from the global path generated by the improved A\* algorithm are used as the local target points of DWA, and the evaluation function of DWA is improved. Finally, the results of simulation and experiment show that compared with the traditional A\* algorithm, the improved A\* algorithm effectively reduces the number of traversed nodes by 51.42% and the search time by 63.32%; and the improved DWA can perfectly avoid concave obstacles and find the globally optimal path.

**Key words:** mobile robot, path planning, improved A\* algorithm, improved dynamic window approach, grid map