基于改进 A*的全局路径规划算法研究

摘　　要

为了降低无人车在多场景中路径规划的时间消耗和冗余节点数量，同时提升实体车的移动效率，保证其在移动过程中的安全性，本研究在即时修复 A*(Anytime Repairing A*, ARA*)算法的基础上，提出了一种双向搜索的修复 A*(Bidirectional Repairing A*, BRA*)算法。该算法融合了双向搜索策略和 ARA*算法，有效地弥补了各自在路径规划过程中的局限性和不足。随后，提出了一种新的优化收敛性判定准则，以防止算法在冗余区域进行过度优化，从而提升了算法的时间效率。为了降低无人车与障碍物发生碰撞的风险，本研究对地图中的障碍物进行了膨胀处理，并基于此设计了自适应代价函数来增强生成路径的安全性。为了最小化生成路径的转角数量，算法采用了 Octile 距离作为启发式函数估计值的计算方法，这种方法更能准确地反映无人车最佳移动路径的实际代价。此外，算法在总代价函数中引入了转角损失项，进一步减少了非必要的转角。

模拟实验结果表明，与传统 A*算法、ARA*算法和双向搜索相比，BRA*算法在路径规划的质量和效率上更具优势，其规划出的路径转角更少，离障碍物距离更远，具有更好的安全性能和移动效率，更适合解决多场景中无人车的路径规划问题。

关键词：无人车；路径规划；改进 A*；自适应代价函数；转角损失

Research on global path planning algorithm based on improved A*

Abstract

To reduce time consumption and redundant node count in path planning for unmanned vehicles across multiple scenarios, while simultaneously improving the movement efficiency of actual vehicles and ensuring their safety during transit, this study proposes a Bidirectional Repairing A*(BRA*) algorithm based on Anytime Repairing A*(ARA*) algorithm. BRA* algorithm integrates the strategies of bidirectional search and the ARA* algorithm, effectively compensating for the limitations and deficiencies of each during the search process. Subsequently, a novel optimization convergence criterion is proposed to prevent excessive optimization in redundant areas, thereby enhancing the algorithm's time efficiency. To mitigate collision risks between unmanned vehicles and obstacles, this study inflates the obstacles in the map and designs an adaptive cost function based on it to enhance the safety of the generated paths. To minimize the number of turns in the generated paths, the algorithm adopts Octile distance as the heuristic function calculation method, which more accurately reflects the actual cost of the optimal path for unmanned vehicles. Additionally, the algorithm introduces a penalty for turning in the total cost function, further reducing unnecessary turns.

The results of simulation experiments show that compared to the traditional A* algorithm, ARA* algorithm and bidirectional search, the BRA* algorithm exhibits superior performance in terms of the quality and efficiency of path planning, and it generates paths with fewer corners, greater distance from obstacles, improved safety and driving efficiency, making it more suitable for solving path planning problems of unmanned vehicles in various scenarios.

Key Words： **Unmanned vehicle; Path planning; Improved A*; Adaptive cost function; Corner loss**

目　　录

1 绪论

1.1 课题背景及目的

近年来，无人车技术已被广泛应用于众多场景，包括但不限于货物配送、安防巡检以及地理测绘等。尤其在仓储物流和生产线中，无人车的需求呈现出显著的增长趋势。在无人车技术的细分领域中，自主导航尤具挑战性，而路径规划技术又是实现高效自主导航的核心组成部分[1]。具体而言，路径规划要求搜索给定区域内某起点到某目标点的最优路径[2]，此过程需确保生成路径的高效与稳定，并尽量最小化总移动代价及无人车与障碍物碰撞的概率。

路径规划通常分为全局规划和局部规划[3]两大类。全局路径规划利用 SLAM 技术[4]生成的静态地图，规划一条从起点到目标点的最佳路径，这一过程侧重于确定一条全局的、避开已知障碍的大致通行路线。而局部路径规划则关注动态环境下的实时导航，需要无人车在沿着全局路径行驶的过程中，不断更新局部地图信息，调整局部路径来避开动态障碍物，从而实现安全有效的导航。

本研究旨在设计一种改进的全局路径规划算法，使之在多种常见的地图场景中均能规划出适合实体无人车运动的路径，进而指导无人车实现自主导航。本研究还将通过与其他路径规划算法进行对比实验，在栅格地图中测试并分析各算法的性能，以验证和评估所提出的改进路径规划算法的可行性及有效性。

1.2 国内外研究现状

1.2.1 路径规划算法概述

为了优化到达目标点的路线，学术界已经发展出各种全局路径规划算法，这些算法根据其设计特点和应用环境各有优势和局限性。目前，全局路径规划算法大致可归纳为两大范畴：一是基于经典理论的传统路径规划算法，二则是借鉴自然界行为策略的智能仿生算法[5]。

在传统路径规划算法领域，Dijkstra 算法由荷兰科学家 Edsger W. Dijkstra 于 1959 年提出[6]，它结合广度优先搜索和贪心策略，解决了权值非负的带权图中单源最短路径问题。该算法因能稳定计算出最优路径而被广泛应用。然而，它的时间复杂度和空间复杂

度会随着地图尺寸的增加而显著提高。A*算法最初由 Stanford 研究院的 Peter Hart 等人于 1968 年提出[7]，该算法在 Dijkstra 算法的基础上引入了启发式代价，用于引导搜索方向，大幅提高了算法的搜索效率。尽管如此，A*算法在实际应用中仍然面临存在大量冗余节点的问题。PRM 算法由 M.H. Overmars 等人于 1996 年首次提出[8]，这是一种将连续空间离散化后再进行路径搜索的方法。此算法显著提高了搜索效率，但在应对地图中的狭窄空间时，需进行大量随机采样来确保这些区域被有效覆盖，这就导致搜索的不稳定性。RRT 算法由 Steven M. LaValle 和 James J. Kuffner Jr.于 1998 年首次提出[9]，该算法采用随机采样和构建树结构的方法来处理局部最优解的问题，其特点是复杂度不受地图离散程度的影响。然而，算法采用了随机采样节点的方法，这通常会导致生成的路径不够平滑。人工势场法由 Khatib 于 1986 年提出[10]，该方法通过构建障碍物斥力场和目标点引力场，利用力场合力指导路径规划以实现避障。方法简洁直观，但容易出现目标不可达的情况和陷入局部最优解的问题。

在智能仿生算法领域，遗传算法[11]最早由 Bremermann 等人基于达尔文的进化论和自然选择理论，于 1958 年提出。该算法通过模拟生物遗传、交叉、变异和选择等机制，实现了对解的持续优化，有效克服了传统算法易陷入局部最优的问题。但该算法在处理复杂优化问题时，其效率可能受到影响。在 1992 年，意大利学者 Marco Dorigo 与 Andrea Colorni 首次提出蚁群算法[12]，该算法灵感源自于蚂蚁在觅食过程中信息素的使用，通过正反馈机制，即信息素的累积，来引导整个蚂蚁群体高效地搜索出最优的觅食路径。虽然此方法加快了算法的收敛速度，但算法的性能极大地依赖于多个相互关联的参数，不当的参数设置可能显著削弱算法的优化能力。另外，强化学习算法也应用到路径规划问题当中[13]，该方法通过直接定义规划目标，采用神经网络对无人车路径规划任务的输入和输出之间的复杂关系进行建模，不断实现自主迭代、更新网络模型[14]。然而，神经网络模型参数众多，这一方法需要大量的数据支撑和较长时间的训练过程，而这些因素又直接影响算法的性能和效率。

1.2.2 路径规划改进方法

目前，A*算法仍被广泛应用于无人车的路径规划问题当中，不过该算法仍面临一些挑战，许多研究者致力于对 A*算法进行改进[15,16]，以增强其效率和在复杂环境下的适应能力。例如，Vadim Bulitko 和 Greg Lee 所提出的算法[17]采用了分级寻径策略，在搜索

过程中基于邻接节点，动态调整节点启发式代价，一定程度上缓解了陷入局部最优解的挑战。然而，这种方法的使用往往伴随着不稳定的时间成本，有时甚至可能导致无法到达目标点的情况出现。秦锋等人[18]提出了一种双向预处理改进搜索算法，旨在解决传统A*算法存在较多的冗余节点和较长搜索时间的问题。然而，在非对称地图场景下，该算法的效果却并不理想，反而可能导致算法性能下降的情况发生。Maxim Likhachev, Geoff Gordon 等人[19]提出了 ARA*算法，该算法引入加速系数，并结合非一致列表快速生成一条次优路径，然后逐步减小加速系数并重新扩展非一致列表中的节点，以此持续优化次优路径直至获得较优路径。然而，由于加速系数的作用，算法更偏向于调整接近目标点的节点，这就导致起点附近的节点优化程度不足。Xing Xu 等人[20]引入了一种动态加速系数的方法，该方法通过在地图中的狭窄通道周围应用更高的加速系数，使得算法更倾向于优先扩展宽阔区域的节点。这一方法不仅提高了算法效率，同时显著增强了算法的安全性能。

综上，尽管现有的某些路径规划算法在特定性能上优于传统 A*算法及其改进版本，但它们往往在场景适应性、路径可行性和算法稳定性等方面存在不足。此外，还有一些算法效果虽然良好，但要求无人车系统具备较高的软硬件性能，具体情况如表 1.1 所示。因此，改进 A*算法的相关研究仍然具有重要的意义。综合上述参考，改进 A*算法的主要途径包括总代价函数的构建、加速系数的设计、搜索策略的融合，以及对多样化场景的适应性调整等方面。

表 1.1 常用路径规划算法在无人车上的性能指标

算法	搜索效率	路径优化性	路径顺滑度	路径安全性	软硬件要求
A*	弱	强	中	弱	弱
双向搜索	强	弱	弱	中	中
LRTA*	中	弱	弱	中	中
ARA*	中	中	中	弱	中
PRM	弱	弱	弱	中	中
RRT	强	弱	弱	中	中
人工势场	弱	弱	中	中	中
遗传	弱	弱	弱	弱	中
蚁群	弱	弱	弱	中	中
强化学习	弱	中	弱	弱	强

1.3 本研究主要工作与创新点

本研究针对无人车在移动过程中可能遭遇的碰撞风险与频繁减速问题，提出了一种改进的 A*算法。该算法旨在生成路径时，尽量避开狭窄区域或者远离障碍物，并在条件允许的情况下，最小化路径的转角个数，从而提高无人车的安全性和移动效率。本研究的主要贡献如下：

(1) 算法将双向搜索策略与 ARA*算法相结合，以弥补 ARA*算法在路径优化中偏向于目标点附近节点的不足。双向搜索策略能够减少冗余节点，提高路径搜索效率，但难以保证两棵搜索树前沿的交点是最优的，而 ARA*算法恰好弥补了这一不足。

(2) 采用 Octile 距离计算当前节点到目标节点的估计代价，在总代价函数中引入了转角损失项，减少了路径长度和转角个数。

(3) 为了降低无人车与障碍物发生碰撞的可能性，对地图中的障碍物进行了膨胀处理，采用自适应函数设计动态权重，降低了搜索列表中离障碍物较近的节点的搜索优先级。

(4) 针对无人车安全性和移动效率，专门设计了新的优化收敛性判定准则，提升了算法效率，并改善了双向搜索中搜索树交点的质量。

1.4 本文组织架构

第一章为绪论，主要介绍了课题的研究背景及其目的。通过综述相关文献，全面概括了国内外路径规划算法的发展现状，并进一步引出了本研究的侧重点与创新点。随后介绍了章节安排与组织结构。

第二章深入探讨总体规划设计及相关技术。首先，对研究课题进一步细分，明确了本文路径规划研究的细分领域和具体场景，并就可视化栅格地图的制定规范展开讨论，明确设计目标，选择 Octile 距离作为启发式函数估计值的计算方法。随后，详细阐述了后续算法设计所需的理论基础，即 ARA*算法的原理。

第三章着眼于改进路径规划算法的设计，针对 ARA*算法在效率和实用性等方面存在的问题，详细阐述了本文核心内容——BRA*算法的设计方案及创新点，其中包括双向搜索策略的融合、新型优化收敛性判定准则的构建、动态加速系数和转角损失项的引入。

第四章中搭建了路径规划模拟测试环境，将 BRA*算法与传统 A*算法、双向搜索策略、ARA*算法在多种地图中进行了对比实验，并从搜索时间、搜索节点数、转角个数、安全系数、路径长度几个方面验证了 BRA*算法的有效性。

第五章为讨论与展望部分，主要对本研究的主要发现和存在的局限性进行了总结，另外还对本课题后续研究工作进行了展望。

2　总体规划设计及相关技术

2.1 总体规划设计

本文专注于在静态平面地图下，无人车自主导航的全局路径规划模块的算法设计问题，即在给定起点、目标点和固定障碍物情况下的路径规划。尽管在实际环境的应用当中，路径规划需要考虑到环境的动态变化、局部障碍物的突发出现以及无人车的尺寸、速度等各种参数，然而鉴于问题的复杂性，本文将重点集中于静态地图下点对点的路径规划。

在本文的研究中，将采用不同类型的栅格地图[21]作为算法对比测试的数据。这些栅格地图都由障碍物单元和可通行单元两个部分组成，单元间以 8 连通方式组织，若任意两个连通节点都不处于障碍物单元内，则这两个节点之间可以相互移动。图 2.1 是某次路径规划的结果示意图，每个色块在图示中都代表一个节点单元，其中，黑色区域表示障碍物，白色区域表示可通行部分[22]，蓝色色块为路径规划的起点，而绿色色块则为路径规划的目标点，灰色区域代表路径搜索过程中探索过的节点，而红色路线则为最终规划的路径。

所有栅格地图数据均被对应大小的矩阵存储，其中矩阵中的点 p(x, y)位于栅格地图的第 y 行、第 x 列。在栅格地图中，任意一个节点可直线或对角地移动到任意一个非障碍物的连通节点上。

图 2.1 还给出了几种启发式距离的路线示例，其中橙色路线对应曼哈顿距离，黄色路线对应欧几里得距离，紫色路线对应 Octile 距离[23]。

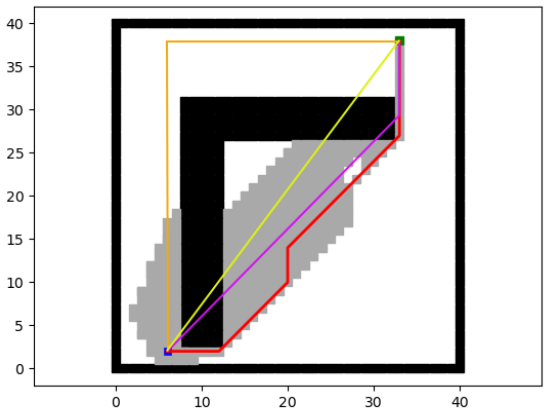


图 2.1　路径规划结果示意图及启发式距离示例

可以发现，在曼哈顿距离的路线示例，只考虑到水平和垂直方向的移动，忽略了对角线方向的移动，其距离代价估计通常会过大，而在欧几里得距离的路线示例，完全忽略了障碍物对路线轨迹的影响，这其实与实际运行路径不可能保持一致，其距离代价估计通常会过小。

相比而言，Octile 距离的路线示例更有可能保证无人车运动的安全性和移动效率，更符合本算法的设计需求，路径代价估计更为准确。因此，本研究采用 Octile 距离作为点与点之间估计移动代价的计算方法。其中，直线移动一个栅格的代价设定为 1，而对角线移动一个栅格的代价设定为$\sqrt{2}$[24]。两节点 $p_1(x_1,y_1)$和 $p_2(x_2,y_2)$之间的 Octile 移动代价计算方法见式(2.1)：

$$Octile(p_1,p_2) = \Delta x + \Delta y + (\sqrt{2}-2) \times \min(\Delta x, \Delta y) \qquad (2.1)$$

其中，$\Delta x = |x_1 - x_2|$，而$\Delta y = |y_1 - y_2|$。

2.2 Anytime Repairing A* 算法原理

2.2.1 传统 A*简概

传统 A*算法结合了 Dijkstra 算法和贪婪最佳优先搜索的特点。Dijkstra 算法是一种基于图搜索的路径规划算法，该算法优先探索已知最短路径节点的邻接节点，更新其最短路径，并以这种方式逐步向外扩展，直至到达目标点。而 A*算法在 Dijkstra 算法的基础上，采纳了贪婪最佳优先搜索的策略，即启发式代价估计值，使得算法更倾向于探索那些距离目标点更近的节点，有效引导了搜索过程，从而大大减少了冗余节点，降低了算法的时间消耗[25]。

节点的搜索优先级由总代价函数值决定，传统 A*算法的总代价函数由两部分组成：一是从起始节点到当前节点的实际代价，二是从当前节点到目标节点的估计代价，该总代价函数值越大，其对应节点的优先级就越低。算法每次会从待扩展集合中，选取优先级最高的节点，同时更新该节点所连通节点的最短路径代价，直至找到目标节点的最短路径。

2.2.2 加速系数

ARA*算法是对传统 A*算法的改进，首先，算法在启发式函数中引入加速系数[26]，

其总代价函数见式(2.2)：

$$f(p) = g(p) + \varepsilon \cdot h(p) \qquad (2.2)$$

其中，f(p)表示地图中节点 p 的总代价估计值，决定了算法扩展节点 p 的优先级，g(p)表示起始节点到当前节点的实际代价，由节点 p 的连通节点扩展节点 p 时计算得到，$h(p)$表示节点 p 到目标节点的预估代价，其准确性对算法的搜索效率产生直接影响，ε为加速系数，通常将其设置为大于 1 的某值来调整算法的性能，主要是路径最优性和算法时间效率之间的平衡。

在传统 A*算法中，g(p)部分的作用是确保搜索的最优性，而 h(p)部分则使得搜索过程有明确的方向性，ARA*算法对 h(p)部分施加了一个大于 1 的加速系数，使得总代价函数的值更偏向于受到 h(p)值大小的影响。换言之，这种加权的启发式函数使得搜索过程更容易朝着目标点的方向扩展节点，从而有效减少了冗余节点的产生，提高了搜索的时间效率。然而，有时这也会使得搜索过程中忽略扩展一些有效节点，导致算法最终得到的路径可能不是最优的。

如图 2.2 所示，引用了 ARA*算法原文献中不同加速系数下的搜索结果，黑色区域表示障碍物，而灰色区域表示已经搜索过的区域，左边三幅分图是使用不同加速系数的情况下，采用 A*算法策略搜索路径得到的结果，右边三幅分图是在对应的加速系数下，采用 ARA*算法策略对路径进行优化的结果（关于 ARA*算法如何对路径进行优化操作，2.2.3 中有详细介绍）。可以发现，加速系数越大，首轮被搜索区域就较小，但最终得到的路径就较次。

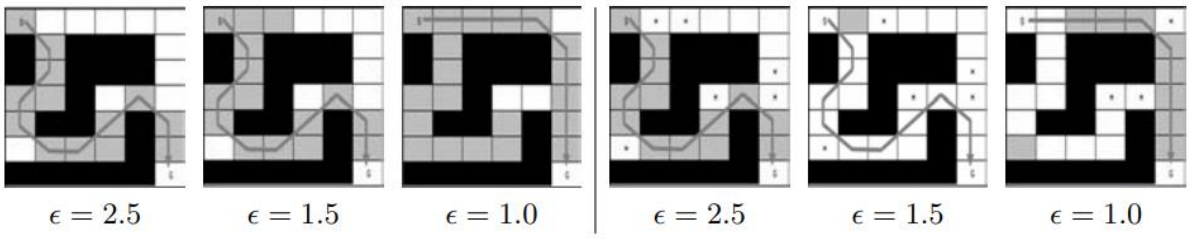


图 2.2　不同加速系数下的 ARA*搜索[19]

2.2.3　ARA*流程介绍

ARA*算法首先要根据地图尺寸，使用一个较大的加速系数进行路径搜索，这样就可以迅速找到一条次优路径，直接指导无人车运动，然后，算法会逐渐减小加速系

数，直至其值不大于 1。每次加速系数的减小也伴随着对上一轮生成路径的优化，即算法会基于上一轮优化剩余的信息，即非一致状态的节点，以及当轮的加速系数进一步优化路径[27]。当加速系数接近 1 时，ARA*算法退化为 A*算法，从理论上来讲，此时得到的路径应该是最优的。

除了加速系数，ARA*算法还介绍了非一致状态的概念，指的是若节点 p 的当前 g(p)值大于算法扩展完所有节点后的$g'(p)$值，即该节点的最短路径搜索还未达到最优，那么就可以认为节点 p 处于不一致状态。在算法中则使用了三个集合：OPEN set、CLOSED set 和 INCONS set。OPEN set 用于存放处于不一致状态的节点，所有被扩展的节点都会被放入此集合。CLOSED set 用于存放已经扩展过的节点，这些节点最终都会达到一致状态。INCONS set 则用于存放被扩展过但搜索过程中发现其仍处于不一致状态的节点。

下面是 ARA*算法搜索的基本流程：

第一步，初始化，设置一个合适且大于 1 的加速系数ε，以用于启发式函数值的计算，设置起始节点的 g 值为 0，地图中其他节点的 g 值为无穷大，将起始节点放入 OPEN set 中。

第二步，搜索或优化路径直至收敛，只要 OPEN set 不为空，就不断从其中取出 f 值最小的节点 p，并将其加入 CLOSE set，若节点 p 的 f 值大于目标节点的 g 值，则认为该轮迭代收敛完成，否则对节点 p 的所有连通节点进行判定：若通过节点 p 到达某连通节点的实际代价大于当前该节点的 g 值，则认为通过节点 p 到达该节点不是一条更好的路径，不做任何操作，否则更新该连通节点的 g 值，并将其父节点设置为节点 p，接着进一步判断该连通节点是否已经在 CLOSE set，若不在则加入 OPEN set，否则加入 INCONS set 暂缓扩展。

第三步，优化前处理，减小加速系数，并将 INCONS set 中所有节点移动到 OPEN set 中。

第四步，重复第二步和第三步，直到加速系数不大于 1，或者找到了合适的路径，或者地图中所有可行路径或节点都被搜索过。

2.3 本章小结

本章阐明了算法设计的基本约束和相关技术，首先说明了算法研究和应用的主要情

形，然后对用于测试的栅格地图及其可视化做了规定，并采用 Octile 距离作为节点间距离代价的计算方法。随后，回顾了传统 A*算法的基本搜索机制，引入了加速系数，分析了系数对总代价函数值大小和算法搜索过程效果的影响，最后还介绍了非一致状态的概念和 ARA*算法搜索的基本流程。

3 Bidirectional Repairing A* 算法

在 ARA*算法中，由于加速系数大于 1，算法会过度依赖于总代价函数中 h(p)的估计值，导致其主要探索目标点附近的节点，这种偏向显著地体现在路径优化阶段，使得算法很容易忽略起点附近可能潜在存在的更优路径。更进一步，在处理复杂地图任务时，ARA*算法可能还会过度优化一些冗余区域，再加上算法需要多次迭代以改进路径，每次迭代的深度又难以精准控制，从而使总体的搜索时间被大大延长。此外，目前大多路径搜索算法过分强调路径的最优性，而未充分考虑无人车与障碍物之间的安全距离，这就很容易导致算法生成的路径中，存在过多靠近障碍物的节点和不必要的转角，不仅降低了无人车的移动效率，也增加了潜在的安全风险。基于以上问题，本研究提出了一系列改进方案。

3.1 双向搜索策略

在路径规划算法的设计过程中发现，尽管 A*算法利用启发式函数，使得其在扩展节点时减少了一部分冗余节点的探索，但当地图中目标点前的障碍物排布复杂时，很容易导致启发式函数估计值与其实际值存在较大的偏误，进一步引致搜索过程中存在另一部分冗余节点。虽然加速系数在一定程度上能够缓解此问题，但要想在不同地图环境中确定其最优值尤为困难，而加速系数设置不合理反而可能导致搜索路径质量不佳。本研究从 ARA*算法借鉴了路径修复方法，并将算法中的节点扩展策略改为双向探索，有效降低了生成冗余节点的可能性，从而提升了路径规划的效率。

单向搜索策略需要从起点开始，沿着任何可能的路径向目标点不断扩展节点，这种方式有时会导致算法向与目标点方向背离的区域进行无效扩展，如图 3.1(a)所示，由于目标点附近障碍物的阻碍，算法对地图左下角和右上角的冗余区域进行了扩展，如图 3.1(c)所示，ARA*算法在搜索首段次优路径时扩展了冗余区域的部分节点，这就导致算法在之后的优化过程中，对这部分冗余区域（浅灰色区域）进行了大量无效的优化。然而，双向搜索策略从两个不同的起点同时进行搜索，并在两棵搜索树前沿相交时完成搜索任务。从效果上看，任一搜索树都为另一搜索树提供了方向上的指引，从而能够快速定位两个路径段的连接节点，有效避免了算法在地图中进一步扩展其他冗余节点，显著提高了算法的时间效率，如图 3.1(b)所示，与图 3.1(a)和图 3.1(c)相比，被扩展节点（灰色区域）数量显著减少。

但双向搜索策略的问题在于，两棵搜索树前沿的首个交点所确定的两个路径段，不一定构成整体上的最优路径，这一问题尤其在非对称地图中表现显著。值得注意的是，这首个交点通常位于最优交点附近。而 ARA*算法存在局限性，即过分优化目标点周围的路径，而忽略了起点附近的路径。然而，该局限性恰好可以被用来优化两棵搜索树的交点。

双向搜索策略下的修复算法伪代码如算法 3.1 所示。

算法 3.1：双向搜索策略下的修复算法

输入：起点 s；目标点 g；初始加速系数ε

输出：规划路径 path

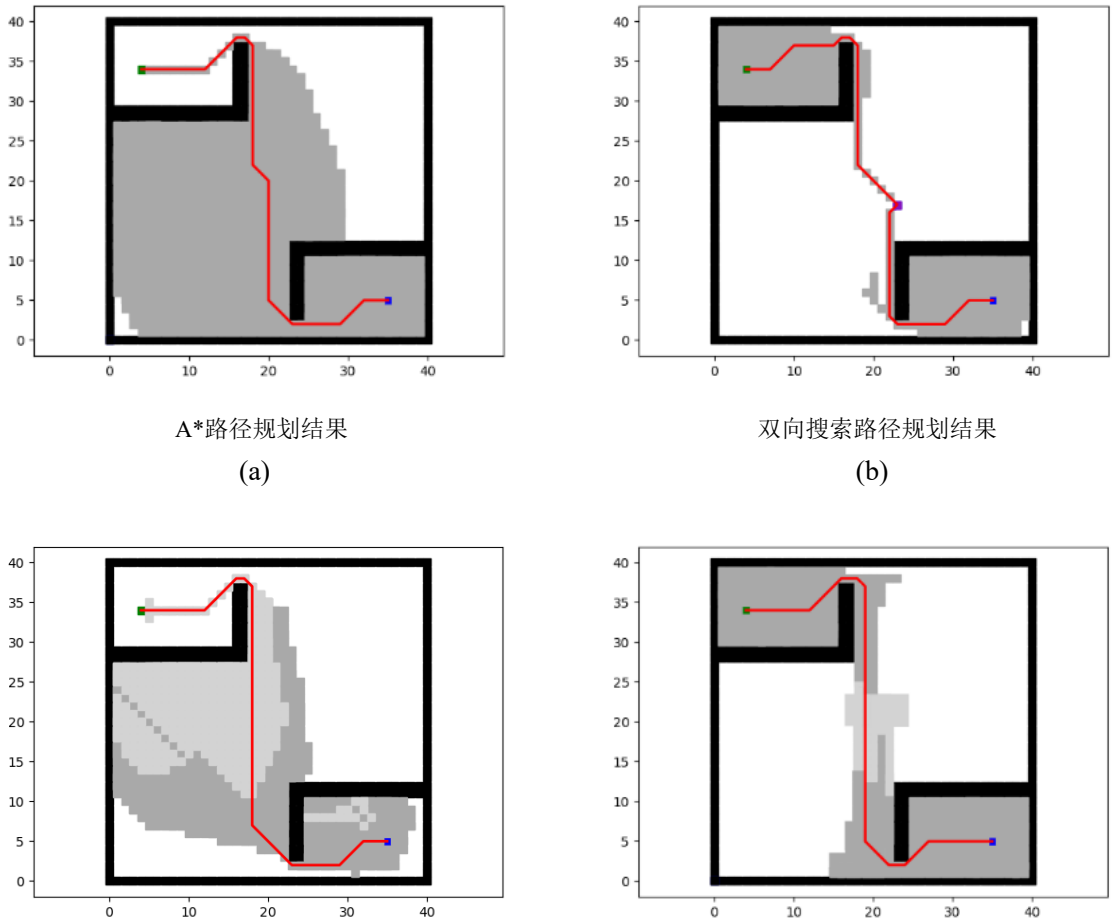
```
1  init OPEN_for with s;
2  init OPEN_back with g;
3  tree1 = forward weighted A* search from s;
4  tree2 = backward weighted A* search from g;
5  m = the intersection of tree1 and tree2;
6  if m is None:
7        return “path not found”;
8  else:
9        publish a suboptimal combined path;
10 while update(ε) > 1 or m not changed:
11       decrease ε;
12       move points, costs from INCONS_for into OPEN_for;
13       move points, costs from INCONS_back into OPEN_back;
14       clear INCONS_for、INCONS_back;
15       update tree1 and tree2;
16       m = new intersection of tree1 and tree2;
17       publish current ε -suboptimal path;
```

总之，将 ARA*算法和双向搜索策略结合起来，在发挥各自搜索优势的同时，也弥补了各自的缺点：一方面，ARA*算法在路径优化规则上的局限性，恰好弥补了双向搜索策略中两棵搜索树前沿的首个交点并非最优的不足，另一方面，双向搜索策略的高效性，又解决了 ARA*算法过分优化冗余区域的问题。图 3.1(b)中紫色色块代表双向搜索策略中两棵搜索树前沿的首个交点。如图 3.1(d)所示，图中的浅灰色区域是 ARA*算法

逻辑对双向搜索前沿交点的优化，可以看到，其得到的路径显然比图 3.1(b)所示路径更加平滑。

3.2 优化收敛性判定的研究

ARA*算法在ε小于 1 时结束优化迭代，尽管每轮迭代中，由于加速系数的作用，降低了探索节点的数量，但算法在几轮迭代中，可能存在大量无效的重复探索节点和冗余节点，如图 3.1(c)所示，这就很难达到如图 3.1(d)中，被优化区域（灰色区域）面积较小的效果。



A*路径规划结果
(a)

双向搜索路径规划结果
(b)

ARA*路径规划结果
(c)

BRA*路径规划结果
(d)

图 3.1　A*、ARA*、双向搜索和 BRA*路径规划结果对比

虽然 ARA*算法的设计旨在平衡路径最优性和搜索效率，但现有的收敛性判定准则仍然存在冗余节点较多的问题。为进一步解决这一问题，BRA*算法提出了一种新的优化收敛性判定准则。

为了在增强算法执行效率的同时，提高无人车实际移动效率，算法不再以ε小于 1 作为优化迭代结束的标准，而是直接将当轮次迭代生成的路径与上一轮次的路径进行比较，若新路径长度更短或转角更小，则认为算法还存在优化的空间；反之，若新路径并未显著改善，则判定算法已经足够收敛，迭代结束。

为了进一步确定双向搜索的最佳交点，算法不再通过当前扩展节点的总估计代价与当前已知最优路径的实际代价的比较，来判定该轮优化是否结束，而是直接判断新扩展的节点是否是双向搜索树的一个更好的交点。具体来讲，就是根据双向搜索中两棵搜索树的历史记录，计算当前扩展节点 p 对应两段路径的实际代价之和，计算方法见式(3.1)，并以此值与上轮迭代中的最佳交点做比较，来得到一个实际代价更小的搜索树前沿交点，作为当前迭代轮次的最佳交点。

$$total(p) = g_{for}(p) + g_{back}(p) \qquad (3.1)$$

其中，$total(p)$代表在该轮迭代中，节点 p 的实际代价，$g_{for}(p)$代表由前向搜索得到的从起始节点到节点 p 的实际代价，$g_{back}(p)$代表由后向搜索得到的从节点 p 到目标节点的实际代价。

优化收敛性判定准则的伪代码如算法 3.2 所示。

算法 3.2：优化收敛性判定准则

输入：无

输出：规划路径 path

```
fp = best pathway of the first iteration of search;
fc = the count of corners in fp;
while true:
    p = best pathway next iteration of search;
    c = the count of corners in p;
    if c/fp >= 1 or length(p)/length(fp) >= 1:
        break;
    else:
        fp = p;
        fc = c;
return fp;
```

3.3 自适应代价函数

ARA*算法中代价函数如式(2.2)所示，该函数适用于地图中的任何节点，并具有统一的加速系数，即使对于靠近障碍物的节点也是如此。有时，算法为了尽可能找到最优路径，可能会优先探索地图中较窄的线路，但这样的线路在实际环境中可能导致无人车与障碍物发生碰撞。为了提高无人车运行的安全性，算法对地图中的障碍物进行了膨胀处理，并根据膨胀处理后的地图数据设计了自适应代价函数，有效地降低了生成路径贴近障碍物的可能性。

首先，算法对地图矩阵数据进行了膨胀处理。在原始矩阵中，可通行单元被标记为0，而障碍物单元被标记为 1。经过膨胀扩散后，障碍物附近的单元也被赋予了介于 0 到 1 之间的数值，该数值的大小取决于该单元与最近障碍物的距离，距离越近，该数值越大，具体值的计算方法如式(3.2)所示。

$$costmap[p] = \frac{1}{\sqrt{d+1}} \tag{3.2}$$

其中，$costmap[p]$表示地图矩阵中节点 p 对应位置的数值，d代表该节点 p 与最近障碍物的距离。

算法 3.3 给出了地图矩阵膨胀的伪代码。

算法 3.3：地图矩阵膨胀算法

输入：地图数据矩阵 costmap;膨胀半径 r

输出：膨胀后的地图数据矩阵 costmap

```
Procedure update_costmap()
1   for obstacle in costmap:
2       costmap_expasion(1, obstacle);

Procedure costmap_expansion(unit, obs)
1   if uint > r:
2       return;
3   else:
4       for neighbor of obs:
5           costmap[neighbor] = max(costmap[neighbor], 1.0/sqrt(unit + 1));
6           costmap_expansion(unit + 1, obs);
```

这样，地图矩阵中从可通行区域到障碍物单元，便形成了 0~1 之间递增的离散值，算法基于此，对原式(2.2)中的加速系数ε进行了修改，修改后的ε值会根据膨胀后的地图矩阵数据动态调整[28]，这一调整使得总代价函数对于靠近障碍物的节点赋予更高的代价值，从而使得算法更倾向于扩展远离障碍物的节点。

为了合理控制加速系数对总代价函数的作用，防止加速系数过大导致生成路径过长，或者加速系数过小导致生成路径靠近障碍物，本研究对加速系数进行了一系列配比测试，自适应代价函数中加速系数ε的设置表达为式(3.3)：

$$\varepsilon = \alpha + (1-\alpha)\cdot(1+costmap[p]) \tag{3.3}$$

其中，ε表示加速系数，α为权重比，$\alpha \in [0,1]$，$costmap[p]$为节点 p 在膨胀后地图矩阵中的数据。

当权重比α取值为 1 时，总代价函数的加速系数保持为 1，此时表现为既不采用自适应函数，也不使用地图矩阵膨胀方法；而当权重比α取值为 0 时，加速系数直接依赖于膨胀后的地图数据，并且加速系数总大于 1，从而可能导致生成高质量路径的效率大大下降。

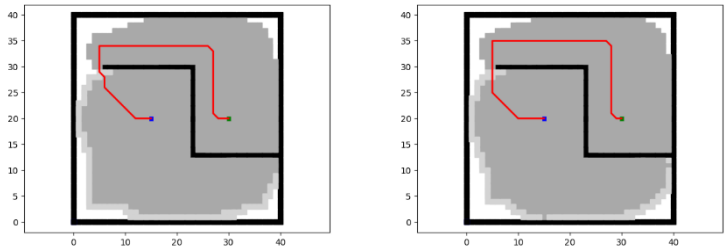
图 3.2 展示了不同膨胀半径下自适应代价函数对 BRA*算法性能的影响。图 3.2(a)是不采用自适应函数和地图膨胀处理的 BRA*算法，其生成路径非常不适合无人车实际运动，安全性较低，图 3.2(b)~(e)是不同膨胀半径下直接依赖地图数据的 BRA*算法，膨胀半径分别为 1、2、3、4 个栅格单位，其生成的路径明显远离了障碍物。

然而，需要注意到的是，当权重比α不够大时，会导致加速系数ε也不够大，部分靠近障碍物的节点由于其到目标节点的估计代价足够小，这就抵消了加速系数的扩大作用，导致算法会优先扩展这一部分节点。这一点在地图的转角处表现较为明显，如图 3.2(b)~(e)中左边的拐角所示，此时就需要稍微增大α值，不过为了保持本文的统一性，这里采用了与后文实验统一的权重值。另外，膨胀半径的增大并非总是有利，当膨胀半径增大，会导致算法的计算复杂度大大增加，增大到一定程度后，规划的路径效果反而变差。

为了使得算法效果最佳，将权重比α在$[0,1]$区间内以 0.05 为步长分为 20 份，分别测试算法路径规划的效果，发现当权重比在$[0.2, 0.4]$时效果较佳。在本文后续实验中，将权重比设置为 0.25。

在本文后续实验中，由于地图障碍物设置比较紧密，将膨胀半径设置为 1.0，实际调用算法时其值需据情况而定。

α=1 (a)　　α=0，r=1 (b)　　α=0，r=2 (c)



α=0，r=3 (d)　　α=0，r=4 (e)

注：α为权重比，r 为膨胀半径

图 3.2　不同膨胀半径下自适应代价函数的 BRA*算法

3.4 转角损失代价

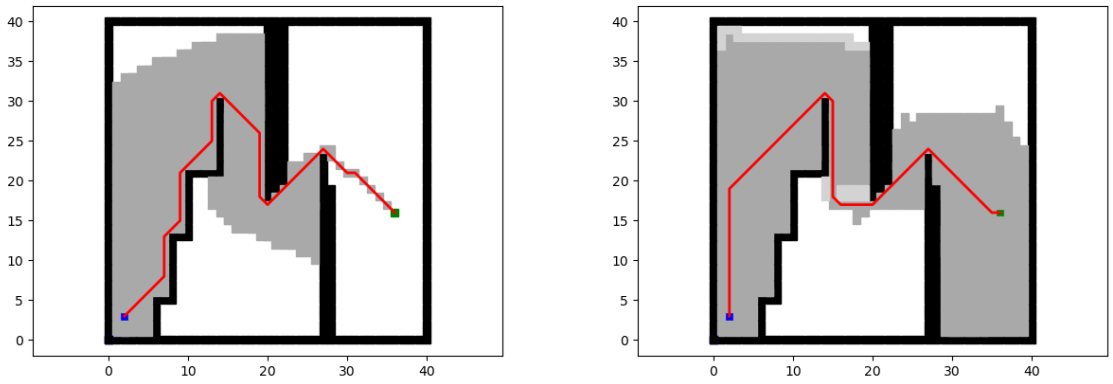
在实际应用中，路径方向每次被改变，通常需要无人车进行减速、转向以及再次加速，而这些步骤往往需要额外的时间和能源，显著降低了无人车的移动效率。传统路径规划算法为了达到最优路径往往会产生大量的转角，特别是在曲折地图场景中，如图 3.3(a)所示，实际上只需一次转角然后沿着对角线移动即可完成任务，如图 3.3(b)左边阶梯式障碍物对应的路径所示。因此，通过最小化路径中转角的数量，可以找到一条更符合实际移动效率的路径。

本研究为此引入了转角损失代价，即算法在扩展节点 p 时，如果被扩展的节点p'引发其当前对应路径的方向发生改变，则到达该节点需要增加额外的代价。

因此，BRA*算法的总代价函数表示为式(3.4)。

$$f(p) = g(p) + \varepsilon \cdot h(p) + loss(p) \tag{3.4}$$

其中，$loss(p)$表示搜索树中到达节点 p 所对应最优路径产生的总转角损失代价，其余项同式(2.2)。在本文实验中，将每次转角的损失均设置为 1.0。



未加入转角代价损失　　加入转角代价损失

(a)　　(b)

图 3.3　转角代价损失算法效果对比图

3.5 BRA*算法

BRA*算法的总体流程如图 3.4 所示。算法创建了三对列表：OPEN set、CLOSE set 和 INCONS set，用于分别存储两个方向上搜索的节点，并建立了两棵搜索树来记录两个方向上的最佳路径。首先，算法将起始节点和目标节点分别加入对应的 OPEN set 中，并将它们的 g 值设置为 0，其他节点的 g 值设置为无穷大。然后，根据 3.3 节中算法 3.3 对地图矩阵数据进行膨胀处理。接着进行路径搜索或者优化（下文简称为 improve path）：首先确定两个 OPEN set 是否为空，如果其中任一集合为空，则认为路径搜索失败；否则两个方向上的搜索交替进行如下操作：从对应 OPEN set 中取出具有最小 f 值的节点 p，并将其放入对应的 CLOSE set 中。然后，判断节点 p 是否存在于另一棵搜索树上，如果是，则认为路径搜索成功；否则遍历节点 p 的连通节点p'，并根据节点 p 的 g 值计算其连通节点p'的 g 值，具体计算方法见式(3.5)：

$$g(p') = g(p) + \varepsilon \cdot cost(p,p') + loss(p,p') \qquad (3.5)$$

其中，$g(p')$表示起始节点到节点p'的实际代价，而$g(p)$表示从起始节点到节点 p 的实际代价。ε代表路径的加速系数，其定义形式同式(3.3)。$cost(p,p')$表示从节点 p 移动到节点p'的代价，而$loss(p,p')$ 表示从节点 p 到节点p'产生的转角损失代价，该值可以通过一搜索树祖先节点组成的路径方向与节点 p 和节点p'的关系来确定[29]，如若节点 p 的祖

先节点的路径方向与节点 p 到节点p'的方向不一致，则需要计上一定损失，否则该损失则计作 0。

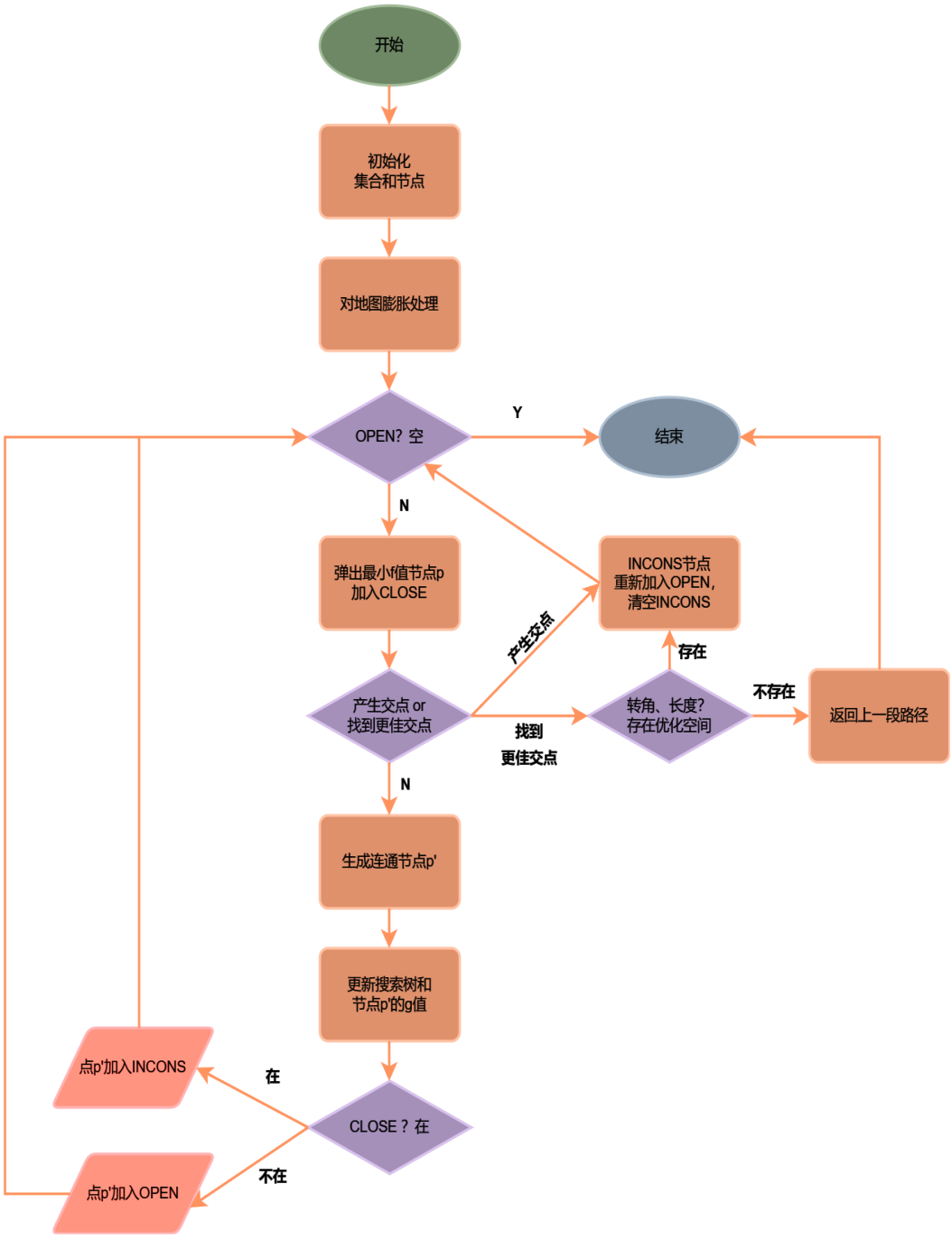


图 3.4　BRA*算法的总体流程

如果新计算的$g(p')$值小于其历史$g'(p')$值，则需要更新其搜索树和其$g(p')$值。如果节点p'不在 CLOSE set 中，则将其加入 OPEN set，否则加入 INCONS set。这样，算法就生成了一条次优路径。

接下来，算法将 INCONS set 重新加入对应 OPEN set 中，并将 INCONS set 清空。

然后不断执行上述 improve path 的步骤优化路径，直到生成的路径转角不再更小、长度不再更短，在此过程中，不再以“节点 p 是否在另一棵搜索树上”作为优化成功的条件，而是采用 3.2 节中的方法来选择双向搜索的最佳交点。

具体细节见 BRA*伪代码——算法 3.4。

算法 3.4：BRA*算法

输入：起点 s；目标点 g；

输出：规划路径 path

```
Procedure improve_path(flag)
1   if OPEN is not empty:
2       p = the min-f point from OPEN;
3       put p into CLOSE;
4       if flag is true:
5           if p in another tree:
6               bm = p;
7               bf = g(p) in tree1 + g(p) in tree2;
8               return bm;
9       else:
10          if bf > g(p) in tree1 + g(p) in tree2:
11              bm = p;
12              bf = g(p) in tree1 + g(p) in tree2;
13              return bm;
14      for p′ in neighbors(p):
15          c = g(p) + ε · cost(p, p′) + loss(p, p′);
16          if direction(tree(p),p) is not direction(p, p′):
17              c += the loss of a corner;
18          if  c< g(p′):
19              g(p′) = c;
20              tree(p′) = p;
21              if  p′ not in CLOSE:
22                  put p′ into OPEN;
23              else:
24                  put p′ into INCONS;
```

```
Procedure main()
1   update_costmap();
2   init OPEN_for with s;
3   init OPEN_back with g;
4   tree1 = imporve_path(true) with forward tree;
5   tree2 = imporve_path(true) with backward tree;
6   m = the first intersection of tree1 and tree2;
7   fp = a suboptimal combined path by m;
8   if m is None:
9         return “path not found”;
10  else:
11        return fp;
12  fc = the count of corners in fp;
13  while true:
14        move points, costs from INCONS_for into OPEN_for;
15        move points, costs from INCONS_back into OPEN_back;
16        clear INCONS_for、INCONS_back;
17        clear CLOSE_for、CLOSE_back;
18        imporve_path(fasle) with tree1;
19        imporve_path(false) with tree2;
20        m = a better intersection of tree1 and tree2;
21        p = a new combined path by m;
22        c = the count of corners in p;
23        if c/fp >= 1 or length(p)/length(fp) >= 1:
24              break;
25        else:
26              fp = p;
27              fc = c;
28  return fp;
```

综合分析表明，BRA*算法成功地将 ARA*算法的修复特性和双向搜索策略的高效性进行了结合，形成了一种新的搜索方法。该方法的复杂度介于 ARA*算法复杂度和双向搜索策略复杂度之间。具体来说，BRA*算法不仅维持了 ARA*算法在环境中调整搜索精度的能力，同时利用了双向搜索的策略，从两个方向同时进行搜索，大幅提升了搜索的有效范围比和速度。在优化收敛性判定准则方面，BRA*算法进行了改进，精细化管

理冗余节点，减少了算法无效节点和重复节点的扩展，搜索时间也随之减少，在搜索效率上相对高效。

3.6 本章小结

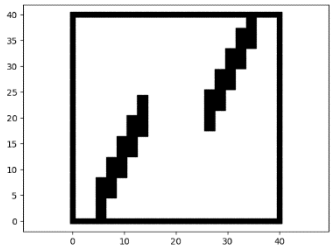
本章主要针对 ARA*算法在路径规划中出现的偏向优化、冗余优化、生成路径安全性低、无人车移动效率差等问题，详细阐述了 BRA*算法的设计和创新。首先，在 ARA*算法中融入了双向搜索策略，弥补了两者的不足，其次，为提高算法搜索效率，专门改进了 ARA*算法的优化收敛性准则，另外，为提高无人车移动的安全性，设计了自适应代价函数，最后，为提高无人车的移动效率，在总代价函数中加入转角损失项。

4　模拟地图实验及其结果分析

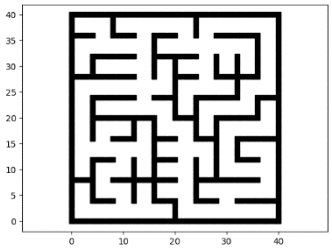
4.1 模拟地图介绍

为了验证 BRA*算法的有效性，本研究在 Python 环境中构建了几种栅格地图模型，这些模型模拟了无人车在实际移动过程中经常遇到的场景。与传统 A*算法、ARA*算法以及双向搜索策略进行对比试验并可视化，来评估 BRA*算法的性能。在试验中，研究也关注算法在处理不同类型地图时的适应能力。

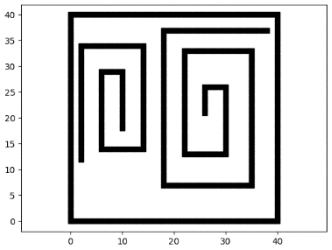
图 4.1(a)～(e)展示了模拟实验中用于测试的五幅栅格地图，大小均为 41*41，分别标记为地图 a～e。地图 a 采用了阶梯状布局，其参差不齐的障碍物配置，容易导致路径为了最优性也变得同样参差不齐，形成大量的转角。地图 b 为迷宫型布局，其复杂的通道结构易使路径搜索陷入死胡同，为路径规划带来极大的挑战。地图 c 展现了螺旋式设计，要求从一螺旋中心移动至另一螺旋中心，此外，该地图周边的狭窄通道增加了路径规划的难度，一般来说，在路径长度相近的情况下，选择较宽通道的路径更具实用性。地图 d 以简单障碍物布局为特征，此类布局在现实世界中出现频次较高，常见于办公室、教室、仓库等环境。地图 e 则是复杂障碍物地图，含有多种杂乱形状的障碍物，分布密集，比较考验算法的全局搜索能力。



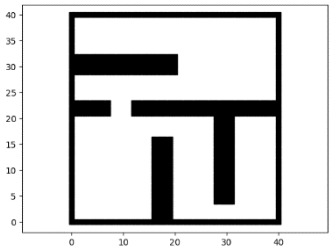
(a)



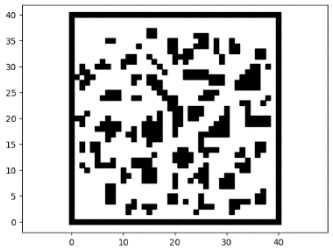
(b)



(c)



(d)



(e)

图 4.1　不同场景类型的栅格地图

4.2 路径规划模拟测试

在本文测试中，设置了五个性能指标来评估路径规划算法的效果，包括搜索时间、搜索节点数、转角个数、安全系数和路径长度[30]。搜索时间指的是算法从开始搜索到确定最终路径的时间消耗，应尽可能保持不长，以保证无人车系统效率和响应速度。搜索节点数表示算法搜索过程中访问或检查过的节点总数，应控制在合理范围内，过多会增加无人车软硬件的要求。转角个数表示路径中转向操作的总次数，无人车在执行转弯操作时需要进行加减速，这增加了能耗但降低了移动效率，因此，转角个数应尽可能保持在一个较低水平。安全系数是指规划路径中位于障碍物膨胀半径范围内节点比例，该值越大说明该路径对无人车来说，安全性越低。路径长度表示路径的实际代价消耗，通常在保证安全性和转角个数的前提下应尽可能较小。综上所述，这五个性能指标越小，则说明算法效果越好。

4.2.1 阶梯地图测试

图 4.2(a)~(d)分别呈现了四种算法 A*、Bidirectional A*、ARA*和 BRA*在阶梯地图上的路径规划结果。A*、Bidirectional A*和 ARA*生成的路径中存在大量的转角，并且这些转角还往往分布在障碍物附近，显著增加了无人车移动的复杂性和无人车与障碍物发生碰撞的风险。因此，BRA*算法所提出的自适应代价函数和转角损失代价策略，可以有效地减少路径中转角数量，并降低路径与障碍物接近的可能性。

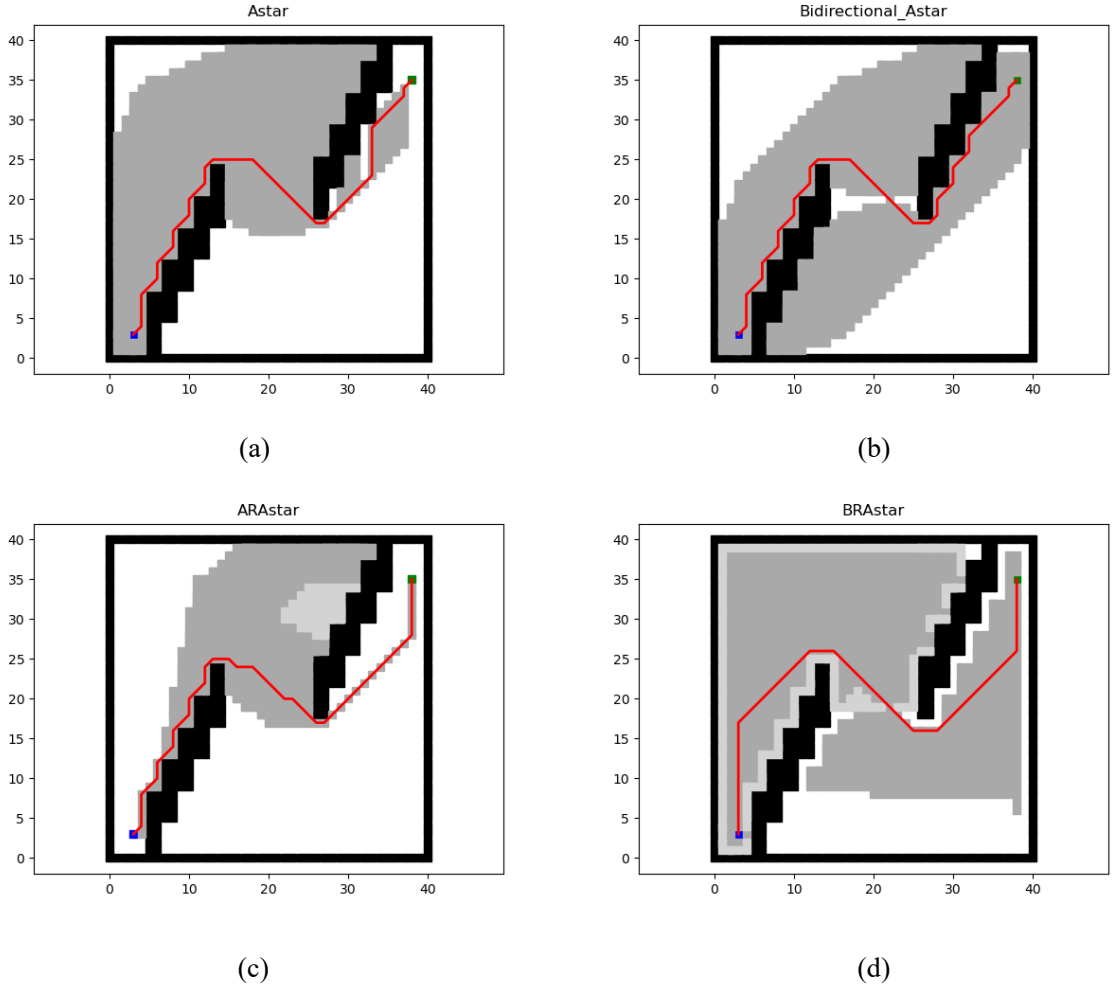
表 4.1 展示了四种算法在阶梯地图上的路径规划性能指标。观察可知，BRA*算法的转角个数为 6 个，与 A*、Bidirectional A*和 ARA*相比分别减少了 66.6%、72.7%和 68.4%，显著提升了无人车运动的稳定性。四种算法在路径长度上相差不大，虽然 ARA*算法的搜索节点数相对较少，但它生成的路径不能保证无人车的安全性和移动效率。而 BRA*算法的安全系数为 0，完全保障了无人车在运动过程中不受碰撞。

总而言之，A*算法具有较长的搜索时间，Bidirectional A*算法生成的路径安全性较低，而 ARA*算法效果比前两者好，但在安全性和转角个数上仍然远不及 BRA*算法。因此可以认为，BRA*算法的综合性能在阶梯地图上远优于 A*、Bidirectional A*和 ARA*算法。

表 4.1 四种算法在阶梯地图上的路径规划性能指标

算法	搜索时间	搜索节点数	转角个数	安全系数	路径长度
A*	116	1170	18	54.5%	55
Bidirectional A*	59	1363	22	72.7%	55
ARA*	65	924	19	50.9%	55
BRA*	61	1311	6	0.0%	59

注：搜索时间单位为 ms.



(a) (b)

(c) (d)

图 4.2 四种算法在阶梯地图上的路径规划结果

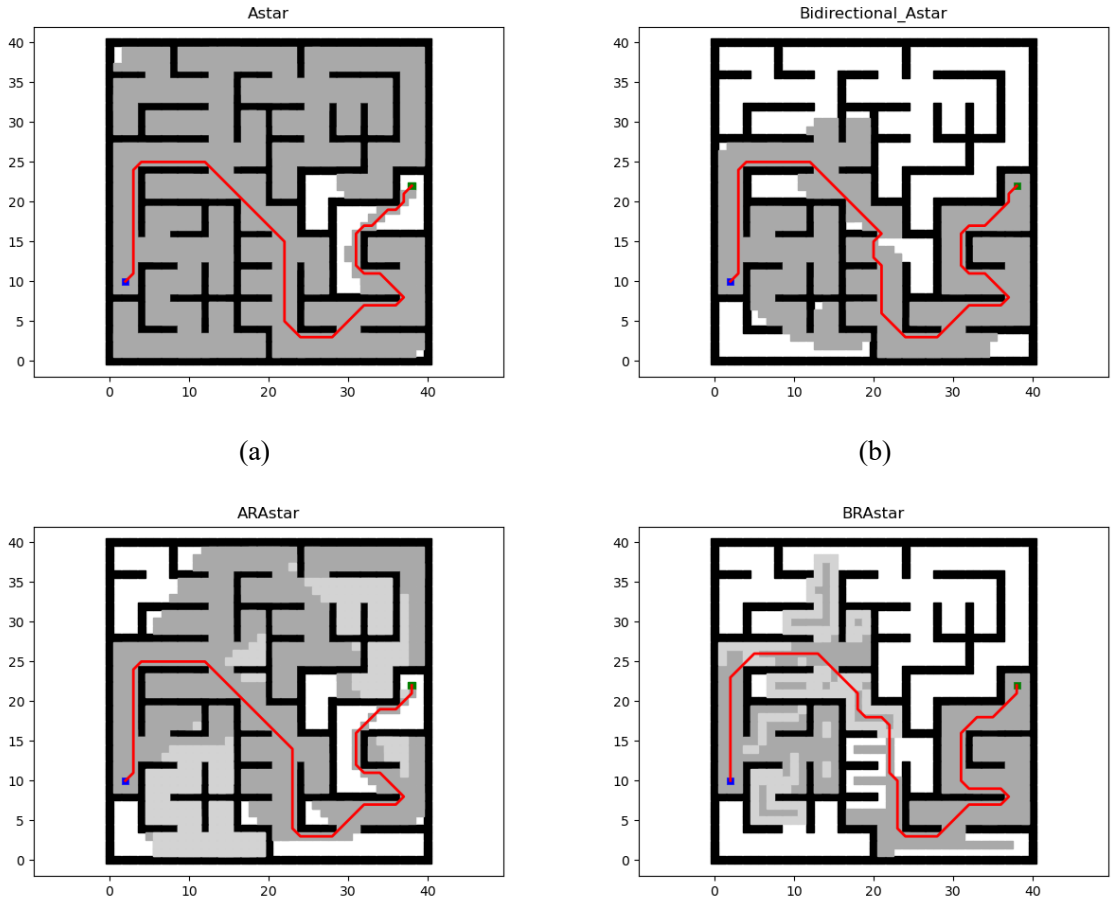
4.2.2 迷宫地图测试

图 4.3(a)~(d)分别呈现了四种算法 A*、Bidirectional A*、ARA*和 BRA*在迷宫地图上的路径规划结果。由于迷宫场景中存在大量的死胡同，A*和 ARA*算法产生了过多的冗余节点，尤其是 ARA*算法，如图 4.3(c)浅灰色区域所示，还对迷宫地图左下角和右上角无关区域进行了无效优化，大大提高了搜索时间和搜索节点数，而 Bidirectional A*

和 BRA*采用了双向搜索策略，巧妙避开了这部分无效搜索，从而降低了算法的搜索时间和搜索节点数。

表 4.2 展示了四种算法在迷宫地图上的路径规划性能指标。观察可知，BRA*算法在搜索时间和搜索节点数上具有很大的优势，分别为 23ms 和 616 个，与 A*、Bidirectional A*和 ARA*相比分别降低了 74.2%、45.2%、79.3%和 56.9%、25.8%、64.5%。另外，尽管 BRA*算法在迷宫地图中的转角数量并不占优势，但观察图 4.3 可知，该迷宫地图通道狭窄，BRA*生成的路径中许多转角实际上是为了避开障碍物而不得不产生，因此这些转角在权衡安全性和移动效率后，是必要的。

总而言之，A*算法和 ARA*算法搜索时间太长、搜索节点数太多，Bidirectional A*算法生成路径过于刻意为了最优性却放弃了无人车的安全性。因此可以认为，BRA*算法的综合性能在迷宫地图上优于 A*、Bidirectional A*和 ARA*算法。



(a) (b)

(c) (d)

图 4.3　四种算法在迷宫地图上的路径规划结果

表 4.2 四种算法在迷宫地图上的路径规划性能指标

算法	搜索时间	搜索节点数	转角个数	安全系数	路径长度
A*	89	1429	21	71.4%	77
Bidirectional A*	42	830	22	80.5%	77
ARA*	111	1733	18	81.8%	77
BRA*	23	616	23	44.6%	83

注：搜索时间单位为 ms.

4.2.3 螺旋地图测试

图 4.4(a)~(d)分别呈现了四种算法 A*、Bidirectional A*、ARA*和 BRA*在螺旋地图上的路径规划结果。四种算法都产生了大量的节点，被搜索区域几乎覆盖全图。除了 Bidirectional A*算法，其他算法生成的路径效果基本达到了地图的极限。然而，Bidirectional A*算法的路径选择受限于两棵搜索树前沿首个交点，不幸选择了地图上部的一条狭窄通道，如图 4.4(b)所示，严重影响了路径的安全系数，而 BRA*算法修改了优化收敛性准则，使得其路径效果依然较好。另外，螺旋地图中左下角空旷区域，仅 BRA*算法生成的路径与 Octile 距离对应，减小了转角的次数或角度。

表 4.3 展示了四种算法在螺旋地图上的路径规划性能指标。观察可知， Bidirectional A*算法搜索时间最短但选择了一条狭窄通道，依然认为不是一个好的算法。另外三类算法效果不相上下，但 BRA*算法在安全性上依然具有压倒性的优势，安全系数为 9.8%，与 A*、Bidirectional A*和 ARA*算法相比分别小 89.7%、90.0%和 89.7%。

总而言之，BRA*算法仅牺牲了少量时间就使得无人车的安全性得到了保障，同时还降低了路径选择狭窄通道的可能性。因此，BRA*算法在螺旋地图上的安全性远优于其他三类算法。

表 4.3 四种算法在螺旋地图上的路径规划性能指标

算法	搜索时间	搜索节点数	转角个数	安全系数	路径长度
A*	135	1424	30	95.9%	172
Bidirectional A*	48	1330	31	97.7%	173
ARA*	103	1590	29	95.3%	172
BRA*	91	1720	25	9.8%	193

注：搜索时间单位为 ms.

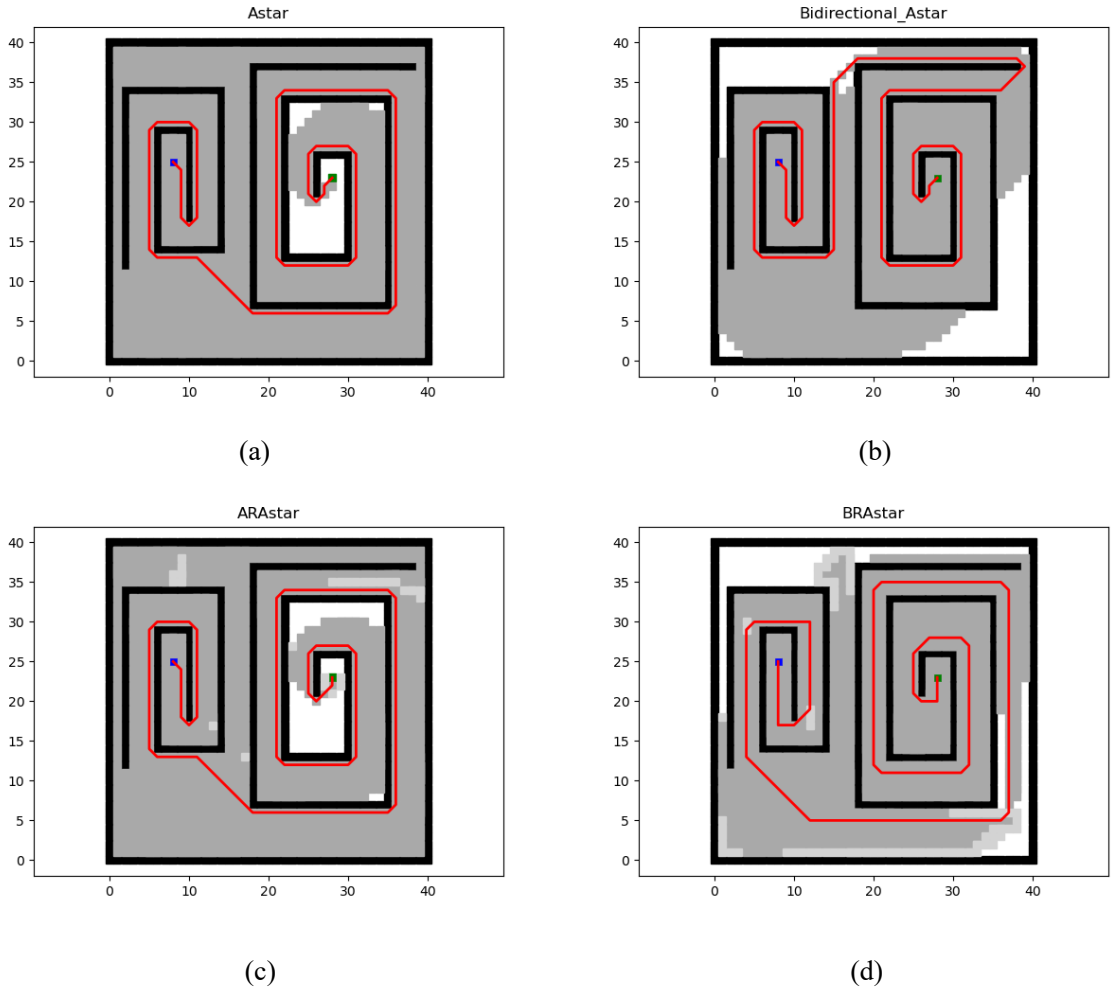


图 4.4　四种算法在螺旋地图上的路径规划结果

4.2.4　简单地图测试

图 4.5(a)~(d)分别呈现了四种算法 A*、Bidirectional A*、ARA*和 BRA*在简单地图上的路径规划结果。传统 A*算法在这种日常场景中反而具有较好的表现，但与 BRA*算法相比还是略逊一筹，如图 4.5(d)所示，BRA*算法生成的路径更加平滑，更适合进一步对曲线优化。而 ARA*算法对中间区域进行了无效优化，白白花费了更多的搜索时间，产生了更多重复的搜索节点，Bidirectional A*算法则在两颗搜索树前沿的交点处形成了几个多余的转角。

表 4.4 展示了四种算法在简单地图上的路径规划性能指标。观察可知，在保证算法搜索时间和搜索节点数与其他算法基本持平或者小有优化的情况下，BRA*算法的转角个数只有 13 个，与 A*、Bidirectional A*和 ARA*算法相比分别减少了 22.2%、40.9%和 23.5%。此外，安全系数仍然远小于其他算法。虽然路径长度指标上稍显劣势，但从图

4.5(d)中可以看出，这都是为了安全性考虑而产生的结果。同时也注意到，双向搜索策略在这种非对称地图中，搜索节点数并不占优势，但时间效率依然能得到保障。

总而言之，A*算法搜索效率太低，Bidirectional A*路径中会存在多余的转角，ARA*算法安全性不足且存在大量重复的搜索节点，因此可以认为，BRA*算法在简单地图上综合性能优于其他三类算法。

表 4.4 四种算法在简单地图上的路径规划性能指标

算法	搜索时间	搜索节点数	转角个数	安全系数	路径长度
A*	96	979	18	43.5%	69
Bidirectional A*	55	1426	22	58.0%	69
ARA*	88	1298	17	49.3%	69
BRA*	66	1322	13	6.0%	83

注：搜索时间单位为 ms.

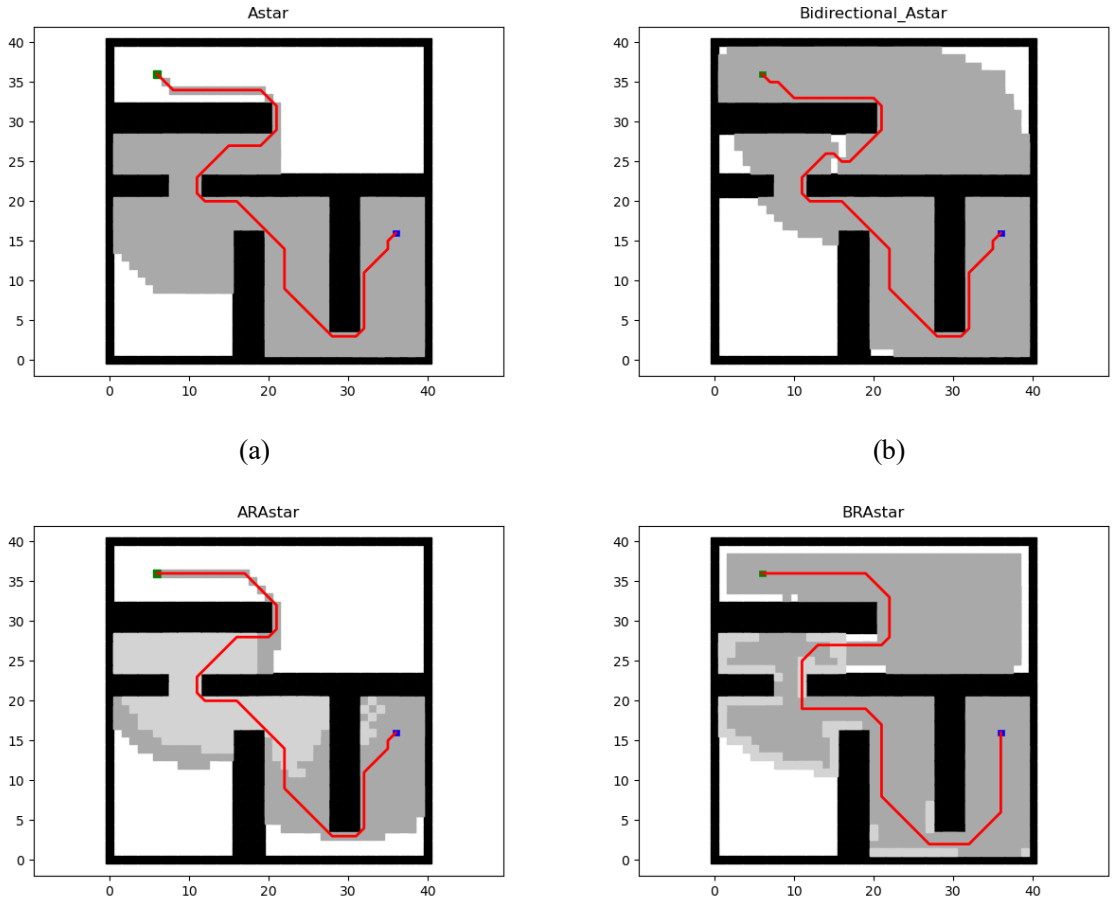
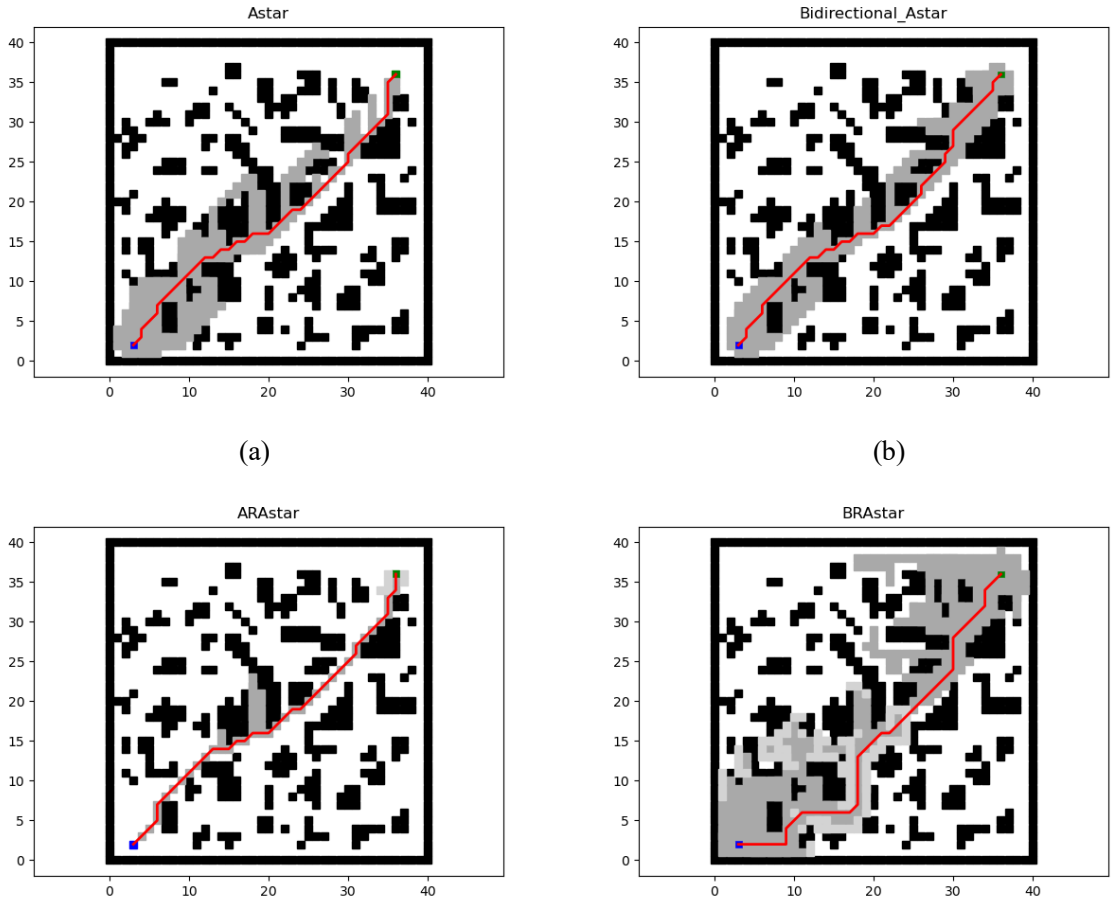


图 4.5 四种算法在简单地图上的路径规划结果

4.2.5 杂乱地图测试

图 4.6(a)~(d)分别呈现了四种算法 A*、Bidirectional A*、ARA*和 BRA*在杂乱地图上的路径规划结果。在杂乱地图中，障碍物排布往往复杂多样，其中总会存在各种各样的狭缝，通过这些狭缝可以找到一条路径长度比较短的路径，如图 4.6(a)~(c)所示，然而，狭缝往往会导致生成的路径具有过多的转角和可能存在安全隐患。在实际应用中，无人车很难在这样狭窄的通道中通过。而 BRA*算法采用了修复方法，它会对初次生成的路径进一步扩展，以更全面地观察地图，从而找到一条更为宽阔、曲折度低的路径。



(a) (b)

(c) (d)

图 4.6 四种算法在杂乱地图上的路径规划结果

表 4.5 展示了四种算法在杂乱地图上的路径规划性能指标。观察可知，BRA*算法的时间性能可能不佳，但其转角个数为 12 个，相比 A*和 Bidirectional A*减少了 33.3%和 45.5%。安全性能依旧远超其他算法。

表 4.5 四种算法在杂乱地图上的路径规划性能指标

算法	搜索时间	搜索节点数	转角个数	安全系数	路径长度
A*	24	245	18	78.0%	41
Bidirectional A*	16	234	22	73.2%	41
ARA*	7	66	15	78.0%	41
BRA*	24	456	12	20.8%	48

注：搜索时间单位为 ms.

总而言之，A*、 Bidirectional A*和 ARA*算法使得无人车安全性和移动效率依旧不足，因此可以认为，BRA*算法在简单地图上综合性能优于其他三类算法。

4.3 本章小结

本章首先介绍了五幅用于算法性能测试的栅格地图，并分析了每幅图在路径规划中的挑战，然后介绍了五个用来评估路径规划算法性能的指标，收集不同算法在不同地图中的不同性能指标并进行对比。结果表明，A*算法搜索过程中通常会产生过多的冗余节点，ARA*算法易在冗余区域无效优化，而双向搜索 A*算法生成的路径并不一定最优，最重要的是，这三种算法生成的路径在安全性和移动效率方面表现不佳。然而，BRA*算法在不增加的时间和空间资源的情况下，能够生成具有更好安全系数和更少转角数目的路径。BRA*的综合性能优于 A*、ARA*和双向搜索 A*。

5 结论与展望

本研究针对传统 A*算法路径规划搜索时间长、生成冗余节点多、路径安全性低、转角数量多等问题，提出了一种改进的 A*算法——BRA*算法。该算法与传统 A*算法相比，将双向搜索策略与 ARA*的修复方法相结合，取长补短，减少冗余节点的同时，改善了双向搜索树前沿的交点。提出了一种新的优化收敛性判定准则，防止了修复方法在冗余区域内过度优化，同时提高了无人车的安全性和移动效率。对地图矩阵进行膨胀处理，基于此设计了自适应的代价函数，大大降低了无人车与障碍物碰撞的风险。采用 Octile 距离作为启发式函数估计值的计算方法，并在总代价函数中引入转角损失代价以最小化路径规划中的转角数量。

实验结果表明，在多场景下，BRA*算法在路径规划上的综合性能优于传统 A*算法、ARA*算法和双向搜索，在保持搜索时间、搜索节点数和路径长度相对稳定的情况下，其规划出的路径转角总是更少，离障碍物距离总是更远，具有更好的安全性和移动效率，更适合解决无人车在多场景下的路径规划问题。

然而，本研究尚未完善，研究仅涉及静态地图下的全局路径规划，在未来的研究中，可以考虑引入局部路径规划方法，以完成对动态地图的路径规划。此外，尽管算法在模拟地图测试中取得了不错的效果，但尚未在实体无人车上进行测试，也未在 ROS 系统中进行仿真验证。

参考文献

[1] Qin H, Shao S, Wang T, et al. Review of autonomous path planning algorithms for mobile robots[J]. Drones, 2023, 7(3): 211.

[2] 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.

[3] Marin-Plaza P, Hussein A, Martin D, et al. Global and local path planning study in a ROS-based research platform for autonomous vehicles[J]. Journal of Advanced Transportation, 2018, 2018: 1-10.

[4] Durrant-Whyte H, Bailey T. Simultaneous localization and mapping: part I[J]. IEEE robotics & automation magazine, 2006, 13(2): 99-110.

[5] 孟范润, 李士心, 周立明, 等. 无人机路径规划算法研究综述[J]. Computer Science and Application, 2023, 13: 1390.

[6] Dijkstra E W. A note on two problems in connexion with graphs[M]//Edsger Wybe Dijkstra: His Life, Work, and Legacy. 2022: 287-290.

[7] 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.

[8] Kavraki L E, Svestka P, Latombe J C, et al. Probabilistic roadmaps for path planning in high-dimensional configuration spaces[J]. IEEE transactions on Robotics and Automation, 1996, 12(4): 566-580.

[9] LaValle S M, Kuffner Jr J J. Rapidly-exploring random trees: progress and prospects. Algorithmic and Computational Robotics: New Directions[J]. 2000.

[10] Khatib O. Real-time obstacle avoidance for manipulators and mobile robots[J]. The international journal of robotics research, 1986, 5(1): 90-98.

[11] Holland J H. Genetic algorithms[J]. Scholarpedia, 2012, 7(12): 1482.

[12] 杨剑峰.蚁群算法及其应用研究[D].浙江大学,2007.

[13] Singh R, Ren J, Lin X. A Review of Deep Reinforcement Learning Algorithms for Mobile Robot Path Planning[J]. Vehicles, 2023, 5(4): 1423-1451.

[14] 周紫瑜.无人车路径规划优化算法研究[D].西安建筑科技大学,2023.DOI:10.27393/d.cnki.gxazu.2023.000276.

[15] 龚鹏, 李文博, 马庆升, 等. 基于改进 A 算法的无人车路径规划研究[J]. 组合机床与自动化加工技术 (3): 17-20, 24.

[16] 缪殷俊,施卫.基于改进A星算法的无人车路径规划研究[J].电脑知识与技术,2024,20(03):4-7.DOI:10.14004/j.cnki.ckt.2024.0161.

[17] Bulitko V, Lee G. Learning in real-time search: A unifying framework[J]. Journal of Artificial Intelligence Research, 2006, 25: 119-157.

[18] 秦锋, 吴健, 张学锋, 等. 基于 A~* 的双向预处理改进搜索算法[J]. 计算机系统应用, 2019, 28(5): 95-101.

[19] Likhachev M, Gordon G J, Thrun S. ARA*: Anytime A* with provable bounds on sub-optimality[J]. Advances in neural information processing systems, 2003, 16.

[20] Xu X, Zeng J, Zhao Y, et al. Research on global path planning algorithm for mobile robots based on improved A[J]. Expert Systems with Applications, 2023: 122922.

[21] Thrun S. Learning occupancy grid maps with forward sensor models[J]. Autonomous robots, 2003, 15: 111-127.

[22] 林彬, 韩光辉, 宋晨晨, 等. 基于辐射扫描算法的机器人路径规划与仿真[J]. 系统仿真学报, 2021, 33(1): 84.
[23] Ulloa C H, Baier J A, Asin-Acha R. Multipath Adaptive A*: Factors That Influence Performance in Goal-Directed Navigation in Unknown Terrain[J]. IEEE Access, 2020, 8: 116724-116732.
[24] 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]. Journal of Frontiers of Computer Science & Technology, 2021, 15(11).
[25] 崔航.基于深度学习的移动机器人路径规划算法设计[D].哈尔滨工业大学,2022.DOI:10.27061/d.cnki.ghgdu.2022.000917.
[26] 李擎, 徐银梅, 张德政, 等. 基于粒子群算法的移动机器人全局路径规划策略[J]. 工程科学学报, 2010, 32(3): 397-402.
[27] 黄梦涛, 李智伟. 改进 ARA* 算法的移动机器人路径规划[J]. Journal of Computer Engineering & Applications, 2022, 58(24).
[28] 沈克宇, 游志宇, 刘永鑫. 基于拟合优先搜索的多场景自适应改进 A* 算法[J]. Computer Engineering & Science/Jisuanji Gongcheng yu Kexue, 2024, 46(1).
[29] 沈克宇, 志宇, 刘永奢, 等. 基于改进 A 算法的移动机器人路径规划[J]. Application Research of Computers/Jisuanji Yingyong Yanjiu, 2023, 40(1).
[30] Güllü A I, Alam M S, Güneş A. Performance Evaluation of Reinforcement Learning and Graph Search-based Algorithm for Mobile Robot Path Planning[C]//2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME). IEEE, 2023: 1-6.