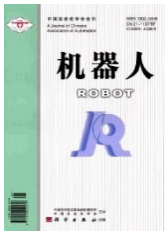


机器人
Robot
ISSN 1002-0446,CN 21-1137/TP

《机器人》网络首发论文

题目：基于改进A*算法的移动机器人路径规划
作者：赵晓，王铮，黄程侃，赵燕伟
DOI：10.13973/j.cnki.robot.170591
收稿日期：2017-10-27
网络首发日期：2018-05-02
引用格式：赵晓，王铮，黄程侃，赵燕伟．基于改进A*算法的移动机器人路径规划．机器人. https://doi.org/10.13973/j.cnki.robot.170591

DOI：10.13973/j.cnki.robot.170591

基于改进 A* 算法的移动机器人路径规划

赵 晓[1]，王 铮[2]，黄程侃[1]，赵燕伟[1,2]

（1. 浙江工业大学特种装备制造与先进加工技术教育部重点实验室，浙江 杭州 310014；
2. 浙江工业大学计算机科学与技术学院，浙江 杭州 310023）

摘 要：为了解决较大场景下 A* 寻路算法存在的内存开销大、计算时间长等问题，本文在 A* 算法的基础上，结合跳点搜索算法，提出一种改进的 A* 算法．该算法通过筛选跳点进行扩展，直到生成最终路径，扩展过程中使用跳点代替 A* 算法中大量可能被添加到 OpenList 和 ClosedList 的不必要节点，从而减少计算量．为了验证改进 A* 算法的有效性，分别在不同尺寸的 2 维栅格地图中进行仿真，仿真结果表明，相比 A* 算法，改进 A* 算法在寻路过程中扩展更少的节点，寻路速度更快，且加速效果随环境地图的增大更加明显．最后将改进 A* 算法应用于移动机器人 Turtlebot2 进行对比实验．实验结果表明，在生成相同路径的基础上，改进 A* 算法的寻路速度较 A* 算法提高了约 200%，能够满足移动机器人路径规划的要求．

关键词：移动机器人；路径规划；A* 算法；跳点搜索
中图分类号：TP242.6 **文献标识码：**A

Mobile Robot Path Planning Based on an Improved A* Algorithm

ZHAO Xiao[1]，WANG Zheng[2]，HUANG Chengkan[1]，ZHAO Yanwei[1,2]

(1. *Key Laboratory of Special Purpose Equipment and Advanced Manufacturing Technology, Ministry of Education, Zhejiang University of Technology, Hangzhou* 310014, *China*;
2. *College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou* 310023, *China*)

Abstract: Aiming at the problems of A* algorithm in the large scene, such as large memory overhead and long computation time, an improved A* algorithm is proposed based on A* algorithm and jump point search algorithm. The algorithm expands by selecting jump points until the final path is generated. During the expansion process, a lot of unnecessary nodes in A* algorithm which are added to OpenList and ClosedList can be replaced by jump points to reduce computation. In order to prove the validity of the improved A* algorithm, the simulation is carried out in 2D grid map of different sizes. The simulation result shows that the improved A* algorithm expands fewer nodes in the pathfinding process, and the pathfinding speed is faster. The acceleration effect becomes more apparent with the increase of environment map. Finally, the improved A* algorithm is implemented on the mobile robot Turtlebot2 to conduct comparative experiments. The experimental result shows that the improved A* algorithm can speed up by about 200% compared with the A* algorithm when generating the same path, and it can meet the requirements of path planning for mobile robots.

Keywords: mobile robot; path planning; A* algorithm; jump point search

1 引言（Introduction）

路径规划是移动机器人实现自主导航的关键技术之一．路径规划是指，在有障碍物的环境中，按照一定的评价标准（如距离、时间、代价等），寻找到一条从起始点到目标点的无碰撞路径[1-3]．

路径规划的核心是路径规划算法，常用的路径规划算法有人工势场法、模糊逻辑算法、栅格法、自由空间法等[4-5]，这些都属于传统算法的范畴．近几年来随着人工智能技术的不断发展，一些智能算法在路径规划中也得到了广泛的应用，如蚁群算法、遗传算法、神经网络算法等[6-7]．由于栅格法建模简单，易于编程实现，本文采用栅格法构建环境模型，进行路径规划实验．基于栅格法的路径规划算法有很多，如 Dijkstra 算法[8]、A* 算法[9]、D* 算法[10]、D* Lite 算法[11] 等，其中 D* 和 D* Lite 算法主要应用于环境未知的动态场景，针对已知的静态场景，A* 算法往往能更快更有效地求解出最优路径[12]．

基金项目：国家自然科学基金（61572438）；浙江省公益技术应用研究计划（2017C31072）．
通信作者：赵燕伟，zyw@zjut.edu.cn 收稿/录用/修回：收稿/录用/修回：2017-10-27/2018-03-21/2018-03-23

A* 算法存在一些不足．由于 A* 算法在寻路过程中需要维护 2 个列表：OpenList 和 ClosedList，时刻对扩展的每个节点进行检测，并寻找代价最小的节点，因此 A* 算法的主要计算量在于对节点的状态检测和代价最小节点的选择．当路径规划的场景较大时，A* 算法往往由于计算量巨大而导致内存占用严重，且计算时间长，寻路效率不高．针对以上问题，Korf 等人[13] 提出一种改进的 A* 算法——IDA*，在 A* 算法的基础上融合了迭代加深算法，无需进行状态判重和估价排序，从而优化了 A* 算法的寻路策略；文 [14] 提出了 HPA* 算法，该算法通过抽象地减小搜索空间来提高搜索速度，有效地提高了寻路效率，但它往往得不到最优路径，还需要进一步搜索来实现对路径的优化，进一步增加了计算量；文 [15] 提出了一种优化方法，通过提高指导搜索的启发式功能的准确性进行加速[15-16]，这种方法通常可以得到最优路径，速度相较 A* 算法也得到大幅提高，但在追求较快速度的同时，存在较高的空间复杂度，使得算法运行占用较多的存储空间；Harabor 等人提出一种跳点搜索算法，筛选有代表性的跳点进行计算，从而避免了对不必要节点的计算，有效提高了寻路效率[17]．为了解决 A* 算法在较大场景寻路过程中存在的内存消耗大、计算时间长等问题，本文结合跳点搜索算法对 A* 算法加以改进，首先对寻路过程中的相关节点进行预处理，筛选出一系列有代表性的跳点进行扩展，从而减少了对大量不必要节点的计算．改进后的 A* 算法不仅减少了计算过程中对内存的占用，还能有效提高计算效率．最后将改进后的 A* 算法应用到移动机器人 Turtlebot2 上，替代原有的 Dijkstra 算法，大大提高了路径规划的效率．

2 问题的描述（Problem description）

2.1 环境的表示方法

在移动机器人路径规划中，首先需要将各种传感器获得的环境信息进行融合，建立一个表示周围环境的地图模型．采用栅格单位描述环境信息的方法由于简单有效、易于实现等特点，在路径规划中得到了广泛的应用．本文采用由边长为 1 的栅格组成的地图来表示机器人周围的环境信息，如图 1 所示．栅格法将机器人工作环境用一系列具有相同尺寸的栅格表示，并且根据环境信息将栅格分成了 2 种状态：占据状态和空闲状态．图中空白栅格表示移动机器人可以通过的区域，体现为空闲状态；黑色栅格表示环境中的障碍物，对应于占据状态，此时机器人不能通过．

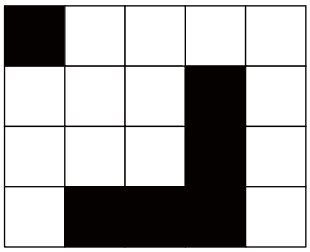


图 1 栅格地图

Fig.1 Grid map

为了便于研究和开展实验，本文做出以下合理假设：

(1) 假设地图和障碍物边界都是在考虑机器人安全距离的情况下建立的，因此可以将机器人视为一个质点，且只能在网格范围内移动；

(2) 在不存在边界和障碍物的情况下，机器人可以向周围 8 个方向的栅格移动，如图 2 所示，并且不考虑自身高度的影响；

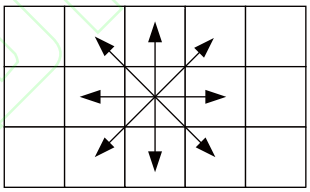


图 2 机器人运动方向

Fig.2 Motion directions of the robot

(3) 在机器人运动过程中，周围环境保持不变．

2.2 A* 寻路算法

A* 算法是静态环境中用于求解最优路径的最有效的直接搜索算法，它结合了 best-first search 和 Dijkstra 算法的思想，在保证得到最优路径的基础上，采用启发式搜索[18]．A* 算法通过一个估价函数来确定搜索方向，从起点开始向周围扩展，通过估价函数计算得到周围每个节点的代价值，选择最小代价节点作为下一个扩展节点，重复这一过程直到到达目标点，生成最终路径．在搜索过程中，由于路径上的每个节点都是具有最小代价的节点，因此得到的路径代价是最小的．A* 算法的估价函数 $f(n)$ 表示为

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

式中的 $f(n)$ 表示从起始点经由任意节点 n 到达目标点的估价函数，$g(n)$ 表示起始点到节点 n 的实际代价，$h(n)$ 表示节点 n 到目标点的估计代价．本文采用欧几里得距离度量两点之间的移动代价：

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

其中，(x_1,y_1)、(x_2,y_2) 分别表示两节点 n_1、n_2 的坐标．对于估价函数 $f(n)$，当 $g(n)=0$ 时，原式变成了计算节点 n 到目标点的估价函数 $h(n)$，A* 算法变为使用贪心策略的 best-first search，计算速度快，但不一定得到最优解；当 $h(n)=0$ 时，原式变为计算起始点到节点 n 的估价函数 $g(n)$，此时 A* 算法转化为 Dijkstra 算法，需要计算大量节点，效率低下．A* 算法在搜索过程中同时计算 $g(n)$ 和 $h(n)$，在考虑搜索效率的同时保证找到最优路径．

A* 算法朝着趋近目标的方向进行搜索，搜索过程中对搜索方向上的每个节点进行考察．当到达某一节点时，该节点的周围节点会被添加到 OpenList，A* 算法会选择 OpenList 中具有最小估价值的节点作为下一个扩展节点，同时将该节点加入到 ClosedList，重复执行这一过程，直到目标节点添加到 OpenList，寻路成功．A* 算法的寻路过程如图 3 所示．

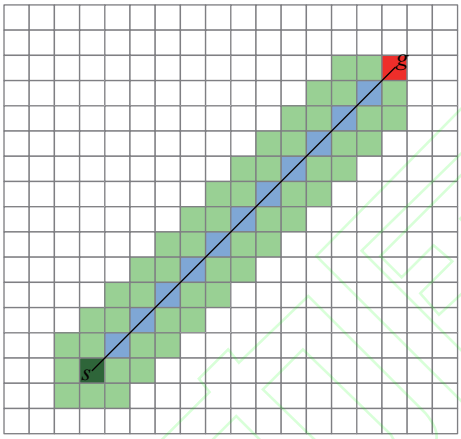


图 3　A* 算法寻路过程

Fig.3　Path-finding process of A* algorithm

A* 算法在寻路过程中存在一些问题．如图 3 所示，从起始节点 s 到目标节点 g，通过 A* 算法生成了最短路径．其中，浅绿色栅格表示在寻路过程中访问过的节点．可以看出，大多数绿色栅格与最终生成的路径并无关联，在 A* 算法寻路过程中，这些栅格构成了大量不必要操作的邻节点，算法本身的搜索策略使得这些节点在寻路过程中不断被维护和访问，直接导致了传统 A* 算法在对较大场景进行路径规划时存在一些缺陷，如计算量大、内存消耗严重且效率不高等．

3　改进 A* 算法（Improved A* algorithm）

为了解决 A* 算法寻路过程中存在的计算量大、内存占用严重等问题，本文结合跳点搜索算法对传统 A* 算法进行改进．

3.1　跳点搜索算法

跳点搜索算法（jump point search，JPS）是一种新的搜索策略，所谓跳点就是搜索过程中可以直接跳跃的节点．如图 4 所示，从 s 节点有 3 条路径可以到达 g 节点，分别为：$s\to a_2\to g$，$s\to b_2\to g$，$s\to c_2\to g$．A* 算法在寻路过程中，对路径上每个节点的估价值进行比较，挑选出代价最低的节点（b_2），这个过程需要访问并操作每个相邻节点（a_2、b_2、c_2）．显然，对 a_2、c_2 两个节点的评估是没有意义的，因为在这种情况下，路径 $s\to b_2\to g$ 的代价永远是最低的．为了进一步提高搜索速度，可以直接根据父节点（即 g 节点）的方向进行搜索，这就是跳点搜索算法的搜索策略．跳点搜索就是在扩展节点的过程中通过筛选某些特定的节点，而不是评估所有相邻节点进行扩展，这些被选中的节点被称为跳点，图中的 g 节点就是一个跳点（这里路径够短，忽略其他路径段）．2 个跳点所组成路径上的中间节点不会被扩展，这也使得扩展过程中计算和存储量大大降低，从而有效减少了算法在寻路过程中的计算量和内存消耗．

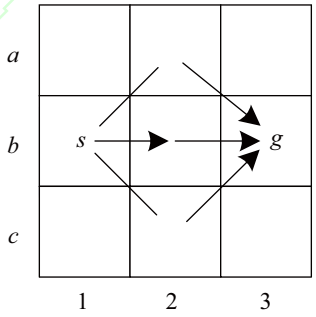


图 4　寻路过程中节点的扩展过程

Fig.4　Node expansion during path-finding process

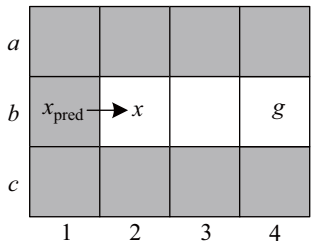
3.2　改进 A* 算法

结合跳点搜索的改进 A* 算法是在 A* 算法的基础上增加了一个预处理的过程，所谓预处理就是通过跳点搜索算法挑选出一批有代表性的跳点，然后将跳点分别添加到 OpenList 进行计算，这样一来避免了 A* 算法寻路过程中对大量中间节点的计算．不同于 A* 算法每次只能规划一小步的策略，改进 A* 算法通过连接跳点，可以实现较长距离的跳跃．因此，在寻路过程中只需要很短的时间进行预处理，从而大大减少了对 OpenList 和 ClosedList 操作的时间，同时有效降低了寻路过程中的内存开销．

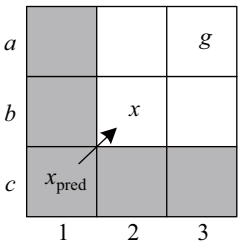
3.2.1　跳点的筛选

改进 A* 算法主要包括两部分，一是对跳点的筛选，二是最终路径的生成．对于跳点的筛选，

文 [19] 提出了几条修剪规则去除不必要节点，本文根据实际情况稍作修改，提出跳点的筛选条件．根据节点周围是否存在障碍物，对以下 2 种情况分别进行讨论．



(a) 水平方向



(b) 对角线方向

图 5 节点周围无障碍物时跳点的筛选条件

Fig.5 Filtering criteria of jump points without an adjacent obstacle

case 1 节点周围不存在障碍物

如图 5(a) 所示，x 和 g 表示栅格地图中的任意节点，x_{pred} 表示节点 x 的父节点，由图可知，$x_{\text{pred}} \to x$ 的方向是水平向右的，要规划一条以 x_{pred} 为起始节点，g 为目标节点的路径．对所有可能产生的结果进行分析可知，规划出来的路径存在 2 种情况，一种是以 x_{pred} 作为起始点，经过节点 x，继续向右扩展到达节点 g，另一种是由 x_{pred} 节点开始，不经过节点 x 到达节点 g，如路径 $x_{\text{pred}} \to c_2 \to c_3 \to c_4 \to g$，其中，经过 x 节点的路径明显更短且减少了对周围邻节点的重复访问，此类重复操作在 A* 算法寻路过程中经常出现，计算量大且毫无意义，因此为了筛选跳点，需要删掉这些不必要的节点，即图中的灰色栅格．根据图 5(a) 可以看出，在到达 b_3 节点之前，$x_{\text{pred}} \to x$ 扩展方向上 x 的邻节点，包括 a_1、c_1、a_2、c_2，都可以由 x_{pred} 直接到达，这时，若经过 x 节点到达，路径将变得复杂．因此，对于这些节点，若不经过 x 到达该点的路径优于经过 x 到达的路径，则认为该节点为扩展过程中的不必要节点，即图中的灰色部分．因此，在节点周围不存在障碍物的情况下，对于水平和竖直方向上的扩展，有以下筛选条件：

$$L(\langle x_{\text{pred}},\cdots,n|x\rangle) \leqslant L(\langle x_{\text{pred}},x,n\rangle) \tag{3}$$

其中函数 $L(\,)$ 表示某一路径段的长度，n 为 x 的相邻节点，$\langle x_{\text{pred}},\cdots,n|x\rangle$ 表示一条以 x_{pred} 为起始点、n 为目标点且不经过 x 的路径，$\langle x_{\text{pred}},x,n\rangle$ 表示路径 $x_{\text{pred}} \to x \to n$，该路径经过节点 x；同理推出节点周围无障碍物情况下，对角线方向扩展时跳点的筛选条件：

$$L(\langle x_{\text{pred}},\cdots,n|x\rangle) < L(\langle x_{\text{pred}},x,n\rangle) \tag{4}$$

图 5(b) 所示为对角线方向上的扩展，根据式 (4)，满足筛选条件的节点为除 a_2、a_3、b_3 之外 x 的其他相邻节点．由以上结论可知，满足筛选条件的节点都是不必要节点，即图中的灰色节点，再对不必要节点进行删除操作，得到具有代表性的跳点．

当扩展到 x 节点时，根据筛选条件筛选出不必要的节点（灰色节点），这时定义 x 的其他相邻节点为节点 x 的自然邻节点．

case 2 节点周围存在障碍物

如图 6 所示，当节点 x 的周围存在障碍物时，不能直接按照式 (3) 和 (4) 进行筛选，对于这种情况，给出筛选条件：

(1) n 不是节点 x 的自然邻节点；

(2) $L(\langle x_{\text{pred}},\cdots,n|x\rangle) > L(\langle x_{\text{pred}},x,n\rangle)$.

所有满足筛选条件的 x 的邻节点 n 定义为强制邻节点，根据筛选条件，图中的 a_3、a_1 节点都是强制邻节点，在扩展过程中它们都将作为跳点进行操作．

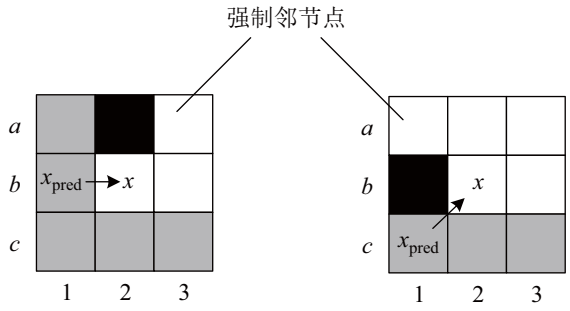


图 6 节点周围存在障碍物时跳点的筛选条件

Fig.6 Filtering criteria of jump points with adjacent obstacles

3.2.2 改进 A* 算法寻路过程

通过以上操作获得满足筛选条件的节点，在节点周围存在障碍物的情况下，提取得到直接作为跳点的强制邻节点，剩下的节点都是寻路过程中的不必要节点．在改进 A* 算法中，删除这些不必要节点得到有代表性的跳点，再执行 A* 算法的一般操作，得到最优路径．改进后的 A* 算法寻路过程如图 7 所示．

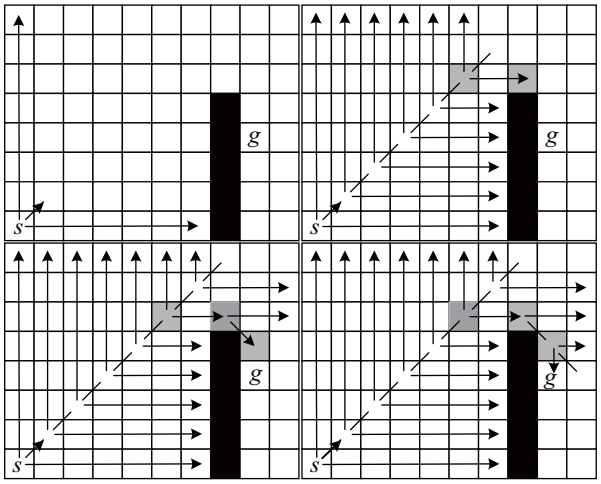


图 7　改进 A* 算法寻路过程

Fig.7　Path-finding process of the improved A* algorithm

在地图栅格中，s 表示起始节点，g 表示目标节点，首先将起始节点 s 添加到 OpenList，并沿着水平和竖直方向进行扩展（扩展方向取决于父节点到当前节点的方向，当父节点不存在时，先向水平和竖直方向扩展），直到遇到障碍物或到达地图边缘，这时候开始沿着对角线扩展．当沿着对角线扩展到一个新的节点时，继续沿水平和竖直方向扩展，重复此过程，直到竖直方向的扩展到达地图的边缘，水平方向的扩展发现一个强制邻节点，这时候将强制邻节点添加到 OpenList，强制邻节点作为一个跳点继续向右扩展，直到到达地图边缘，此时沿着对角线继续扩展，在扩展过程中发现下一个强制邻节点，将其加入 OpenList 并作为新的跳点继续沿水平和竖直方向扩展，直到发现目标点．这时停止迭代，将目标点加入 OpenList，得到最终路径．改进 A* 算法的流程图如图 8 所示．

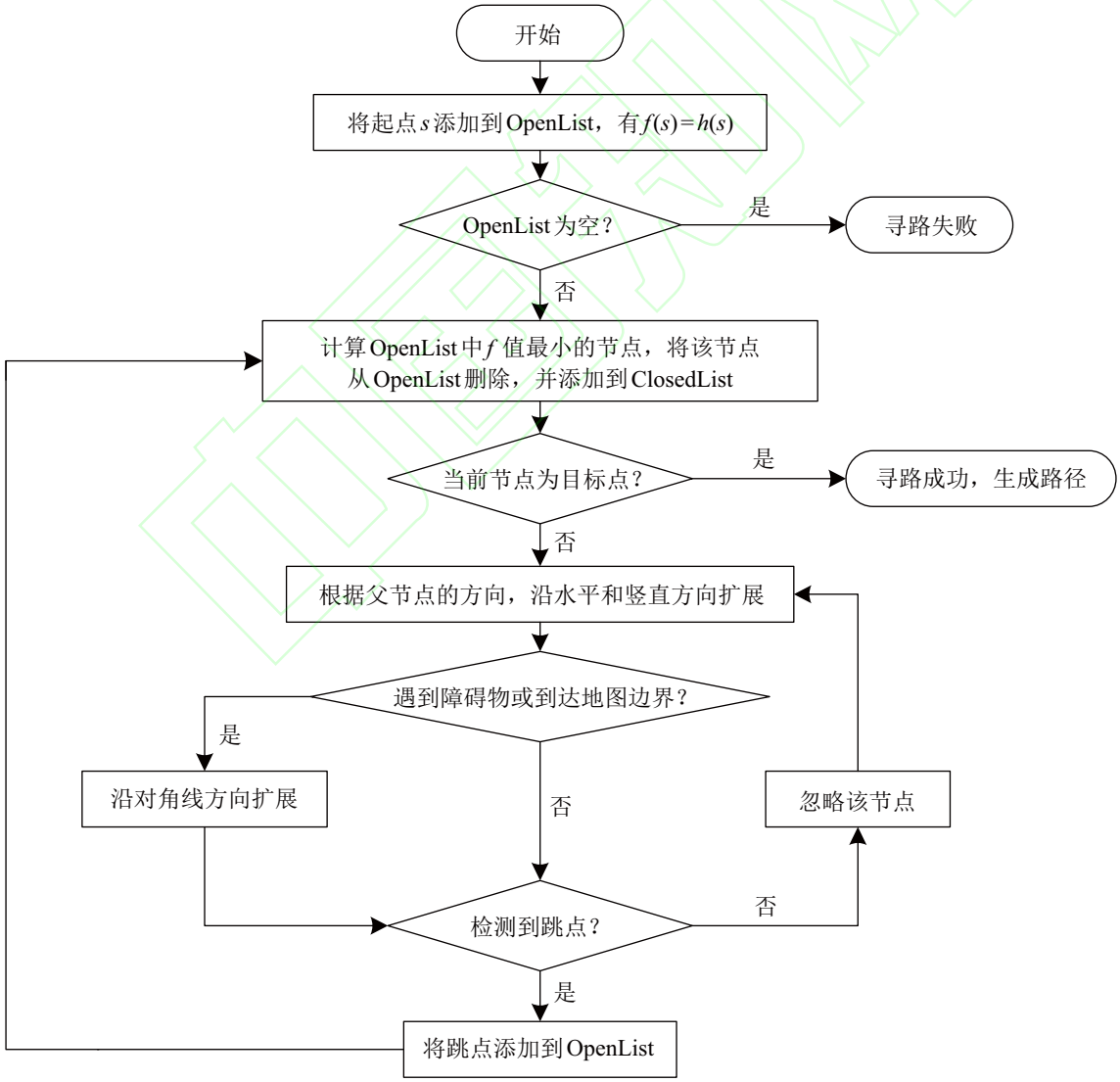


图 8　改进 A* 算法流程图

Fig.8　The flowchart of the improved A* algorithm

4 仿真及实验验证（Simulation and experimental verification）

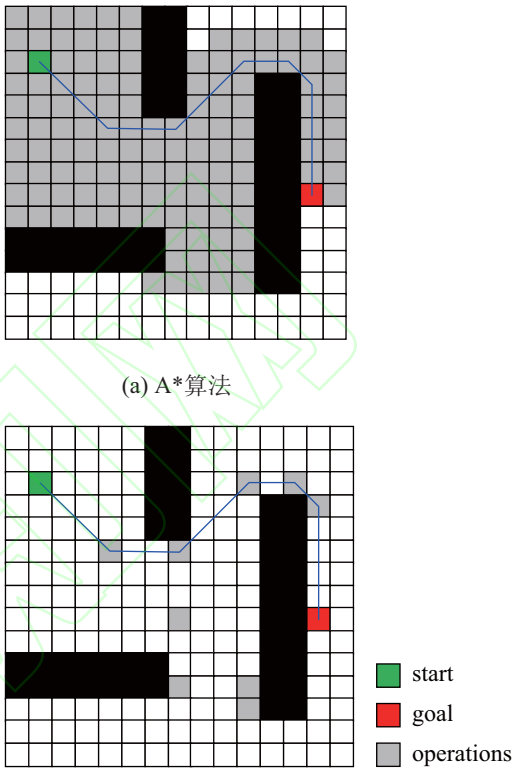
4.1 仿真分析

为了验证改进 A* 算法的有效性，本文分别对 A* 算法和融合了跳点搜索算法的改进 A* 算法进行仿真，在 5 个不同尺寸的栅格地图中进行实验，实验采用的栅格地图障碍物密度均为 1/5，计算机配置为：Windows 10 操作系统，处理器为 i5-7200U，主频 2.5 GHz，运行内存为 8 G.

图 9 所示为 15×15 栅格地图下的仿真实验，其中绿色栅格表示起始节点，红色栅格为目标节点，灰色栅格表示寻路算法在搜索过程中访问过（曾被添加到操作列表）的节点，蓝色折线表示生成的最终路径. 可以看出，相同环境下 2 种寻路算法能够生成同样的路径，且改进 A* 算法在寻路过程中操作的节点数量远少于 A* 算法. 其他地图环境下的仿真结果如表 1 所示.

表 1 给出了 A* 算法与改进 A* 算法在不同栅格地图中的搜索时间和扩展节点数量的对比，可以看出，改进 A* 算法在大部分情况下能生成和 A* 算法相同的路径，甚至有所优化. 在较小栅格地图进行实验时，改进 A* 算法的加速效果不是特别明显，当栅格地图为 50×50 以及更大尺寸的时候，改进 A* 算法在搜索速度上有了明显提升，这种加速效果随着环境地图的增大变得更加突出.

根据表 1 可知，在保证得到同样路径的基础上，改进 A* 算法在寻路过程中将扩展节点的数量减少了一个数量级，从 A* 算法寻路过程中的访问每个邻节点并估价排序转变为对数量更少的跳点进行操作，从而大大降低了计算的复杂度，减少了寻路过程中对内存的消耗. 实验结果表明，改进 A* 算法不仅有效提高了 A* 算法的搜索速度，还能大大减少搜索过程中扩展节点的数量，无论是寻路效率还是对算法的优化，改进 A* 算法都明显优于 A* 算法.



(a) A*算法

(b) 改进A*算法

图 9 A*和改进 A* 算法的仿真结果

Fig.9 Simulation results of the A* algorithm and the improved A* algorithm

表 1 A* 算法和改进 A* 算法的搜索时间与扩展节点数量对比

Tab.1 The search time and the number of extended nodes of the A* algorithm and the improved A* algorithm

地图大小	搜索时间 /ms		扩展节点数量 / 个		路径长度		搜索时间之比	扩展节点数量之比
	A* 算法	改进 A* 算法	A* 算法	改进 A* 算法	A* 算法	改进 A* 算法		
15×15	0.3000	0.2050	128	21	19.90	19.90	1.46	6.10
30×30	0.4800	0.2600	311	29	30.49	30.49	1.85	10.72
50×50	1.9000	0.3400	1406	35	59.70	59.70	5.59	40.17
50×100	14.7500	2.3550	2202	47	112.60	106.50	6.26	46.85
100×100	19.3200	2.6700	2991	67	143.90	143.90	7.24	44.64

4.2 实验验证

为了验证改进 A* 算法在移动机器人路径规划中的可行性，将改进 A* 算法应用到基于 ROS（机器人操作系统）的移动机器人 Turtlebot2 上，如图 10 所示. Turtlebot2 采用体感相机 Kinect 获取外界环境信息，经计算生成 3 维点云数据，并转换成为模拟激光数据后利用 amcl 和 gmapping 模块完成定位和 2 维地图的构建，最后通过 move_base 模块中的 global planner 和 local planner 分别实现全局和局部路径规划[19].

图 10　Turtlebot2 移动机器人

Fig.10　The mobile robot Turtlebot2

global_planner 模块的实现框架如图 11 所示，在 Turtlebot2 中，默认使用 Dijkstra 算法实现全局路径规划．为了方便开展实验，本文直接采用改进后的 A* 算法代替原来的 Dijkstra 算法．

本文的实验场景为实验室中的一段楼道，如图 12(a) 所示．图 12(b) 为楼道的整体环境图，实验过程中分别设置 p_1, p_3 为起始点，p_2, p_4 作为目标点．首先移动 Turtlebot2 构建出该段楼道的 2 维地图，在建好的地图上进行路径规划实验，通过对比同一路径下不同寻路算法的计算效率来验证改进 A* 算法的有效性．

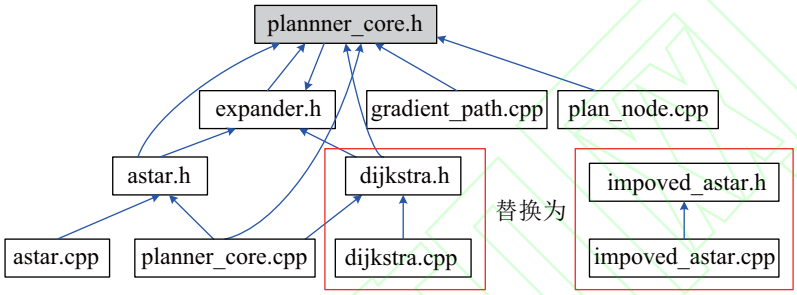
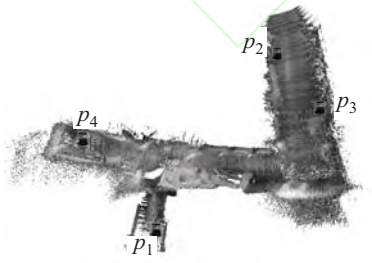


图 11　global_planner 实现框架

Fig.11　Implementation framework of global_planner

(a) 实验场景

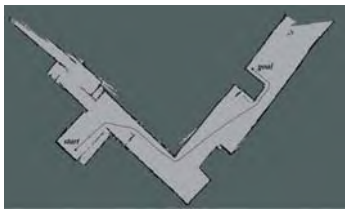


(b) 场地整体环境

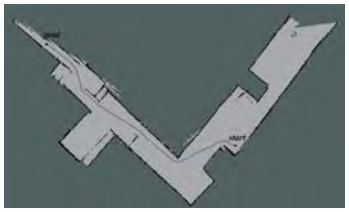
图 12　改进 A* 算法的场地实验

Fig.12　Field experiment of the improved A* algorithm

改进 A* 算法的寻路结果如图 13 所示．该图为 ROS 的可视化工具 RViz 中显示的 2 维地图和路径规划结果．图中的浅灰色区域为地图中实验室的 2 维形态，2 组实验分别设置不同的起点和终点，蓝色线段为改进 A* 算法生成的路径．



(a) 路径1



(b) 路径2

图 13　改进 A* 算法的 Turtlebot2 路径规划结果

Fig.13　Path planning result of the improved A* algorithm on Turtlebot2

表 2 和表 3 分别列出了 2 种算法在不同路径下的寻路时间，考虑到测量误差的影响，本文认为改进 A* 算法基本能够生成最优路径．同时，在搜索速度上，改进 A* 算法明显优于传统 A* 算法．

实验结果证明，结合了跳点搜索的改进 A* 算法，能够有效完成移动机器人的路径规划．且相比传统 A* 寻路算法，改进 A* 算法优化了搜索策略，

因而计算速度更快，寻路效率更高.

表 2 路径 1 下 2 种算法寻路时间对比

Tab.2 Comparison of search time between two algorithms on path 1

	A* 算法	改进 A* 算法
路径长度 /m	15.1	15.0
寻路时间 /ms	241.90	73.51

表 3 路径 2 下 2 种算法寻路时间对比

Tab.3 Comparison of search time between two algorithms on path 2

	A* 算法	改进 A* 算法
路径长度 /m	13.8	14.0
寻路时间 /ms	196.34	67.10

5 结论（Conclusion）

A* 算法在寻路过程中存在内存开销大、计算时间长等缺点，不能满足移动机器人在较大场景路径规划中的实时性要求. 为了提高路径规划的速度，本文结合跳点搜索算法，对 A* 算法加以改进，在搜索过程中优化搜索策略，通过筛选有代表性的跳点进行扩展，取代了原算法中对每个邻节点的操作，从而提高寻路效率. 不同规格栅格环境下的仿真实验证明，改进 A* 算法在保证生成最优路径的同时，能够显著提高寻路速度，减少内存开销，特别是在较大场景下效果更加明显. 将改进 A* 算法应用在 Turtlebot2 机器人上的实验充分表明，改进 A* 算法可以满足实际要求，且优化效果明显.

参考文献（References）

[1] 曲道奎，杜振军，徐殿国，等. 移动机器人路径规划方法研究 [J]. 机器人，2008，30(2)：97-101,106.
Qu D K, Du Z J, Xu D G, et al. Research on path planning for a mobile robot[J]. Robot, 2008, 30(2): 97-101,106.

[2] 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.
Wang D J. Indoor mobile-robot path planning based on an improved A* algorithm [J]. Journal of Tsinghua University: Science and Technology, 2012, 52(08): 1085-1089.

[3] Jeddisaravi K, Alitappeh R J, Pimenta L C A, et al. Multi-objective approach for robot motion planning in search tasks[J]. Applied Intelligence, 2016, 45(2): 305-321.

[4] 刘一松，魏宁，孙亚民. 基于栅格法的虚拟人快速路径规划 [J]. 计算机工程与设计，2008(5)：1229-1230,1267.
Liu Y S, Wei N, Sun Y M. Path planning algorithm based on grid method for virtual human[J]. Computer Engineering and Design, 2008(5): 1229-1230,1267.

[5] Bakdi A, Hentout A, Boutami H, et al. Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control[J]. Robotics and Autonomous Systems, 2017, 89(1): 95-109.

[6] 刘建华，杨建国，刘华平，等. 基于势场蚁群算法的移动机器人全局路径规划方法 [J]. 农业机械学报，2015，46(9)：18-27.
Liu J H, Yang J G, Liu H P, et al. Robot global path planning based on ant colony optimization with artificial potential field[J]. Transactions of the Chinese Society of Agricultural Machinery, 2015, 46(9): 18-27.

[7] 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.
Zhu D Q, Yan M Z. Survey on technology of mobile robot path planning[J]. Control and Decision, 2010, 25(7): 961-967.

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

[9] 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.

[10] Stentz A. Optimal and efficient path planning for partially-known environments[C]//IEEE International Conference on Robotics and Automation. Piscataway, USA: IEEE, 2002: 3310-3317.

[11] Koenig S, Likhachev M. Fast replanning for navigation in unknown terrain[J]. IEEE Transactions on Robotics, 2005, 21(3): 354-363.

[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.
Xin Y, Liang H W, Du M B, et al. An improved A* algorithm for searching infinite neighbourhoods[J]. Robot, 2014, 36(5): 627-633.

[13] Korf R E. Depth-first iterative-deepening: An optimal admissible tree search[J]. Artificial Intelligence, 1985, 27(1): 97-109.

[14] Botea A, Müller M, Schaeffer J. Near optimal hierarchical path-finding[J]. Journal of Game Development, 2004, 1: 7-28.

[15] Goldberg A V, Harrelson C. Computing the shortest path: A* search meets graph theory[C]//Annual ACM-SIAM Symposium on Discrete Algorithms. New York, USA: ACM, 2005: 156-165.

[16] Sturtevant N R, Felner A, Barrer M, et al. Memory-based heuristics for explicit state spaces[C]//International Joint Conference on Artificial Intelligence. USA: International Joint Conferences on Artificial Intelligence, 2009: 609-614.

[17] Harabor D, Grastien A. The JPS pathfinding system[C]//5th Annual Symposium on Combinatorial Search. Menlo Park, USA: AAAI, 2012: 207-208

[18] Korf R E. Real-time heuristic search[J]. Artificial Intelligence, 1990, 42(2/3): 189-211.

[19] Harabor D, Grastien A. Online graph pruning for pathfinding on grid maps[C]//25th AAAI Conference on Artificial Intelligence. Menlo Park, USA: AAAI, 2011: 1114-1119.

作者简介：

赵 晓（1993 –），男，硕士生. 研究领域：移动机器人路径规划，视觉 SLAM.

王 铮（1986 –），男，博士. 研究领域：智能系统与智能自动化技术.

黄程侃（1994 –），男，硕士生. 研究领域：移动机器人路径规划.