Journal of Computer Applications
计算机应用, 2020, 40(7): 2021 - 2027
ISSN 1001-9081
CODEN JYIIDU
2020-07-10
http://www.joca.cn

文章编号:1001-9081(2020)07-2021-07
DOI:10.11772/j.issn.1001-9081.2019112016

基于改进A*算法的无人车路径规划

祁玄玄,黄家骏,曹建安*
(西安交通大学 电气工程学院,西安 710049)
(*通信作者电子邮箱2787477370@qq.com)

摘　要:传统的A*算法在无人车路径规划中存在规划时间较长和搜索范围较大的缺点。综合分析A*算法的计算流程后,从四个方面对A*算法进行改进:1)目标性拓展,即根据待扩展节点和目标节点的相对位置来有目标性地选择不同的象限进行节点拓展;2)目标可见性判断,即判断待扩展节点与目标点之间有无障碍物,若无障碍物则跳出A*算法的探索过程,以此减少多余的搜索;3)改变A*算法的启发函数,即增加待扩展节点的n辈父节点到目标点的代价估计,以此减少到目标点的代价估计的局部最优情况;4)改变扩展节点的选取方略,即改变传统的最小化启发函数来选择扩展节点的方式,通过引入模拟退火法来优化扩展节点的选择方式,使得搜索过程尽可能向靠近目标点的方向进行。最后通过Matlab仿真实验结果表明,在模拟的地图环境下,提出的改进A*算法在运行时间上减少67.06%,经历的栅格数减少73.53%,优化路径长度浮动范围在±0.6%。

关键词:路径规划;A*算法;目标性拓展;启发函数;模拟退火
中图分类号:TP242.6　　**文献标志码:**A

Path planning for unmanned vehicle based on improved A* algorithm

QI Xuanxuan, HUANG Jiajun, CAO Jian'an*
(*School of Electrical Engineering, Xi'an Jiaotong University, Xi'an Shaanxi 710049, China*)

Abstract: The traditional A* algorithm has the disadvantages of long planning time and large search range in unmanned vehicle path planning. After comprehensively analyzing the calculation process of the A* algorithm, the A* algorithm was improved from four aspects. Firstly, targeted expansion, that is, different quadrants were selected with target for node expansion according to the relative position of the node to be expanded and the target node. Secondly, target visibility judgment, that is, whether there were obstacles between the node to be expanded and the target point was determined, if there were no obstacles, A* algorithm jumped out of the exploration process to reduce redundant searches. Thirdly, the heuristic function of the A* algorithm was changed, that is, the cost estimation of the n-th generation parent node of the node to be expanded to the target point was increased, thereby reducing the local optimal situation of the cost estimation to the target point. Fourthly, the selection strategy of the expanded nodes was changed, that is, the traditional method of minimizing the heuristic function to select the expanded nodes was changed, and the simulated annealing method was introduced to optimize the selection method of the expanded nodes, so that the search process was performed as close to the target point as possible. Finally, the Matlab simulation experimental results show that, under the simulated map environment, the improved A* algorithm has the running time reduced by 67.06%, the number of experienced grids decreased by 73.53%, and the fluctuation range of the optimized path length is ± 0.6%.

Key words: path planning; A* algorithm; targeted expansion; heuristic function; simulated annealing

0　引言

路径规划在智能车运动控制中占有核心地位,路径规划算法的效率将直接影响无人车的寻路效率及实施规划能力。目前路径规划算法基本分为两种类型:基于图搜索算法的传统算法和智能算法。图搜索算法主要指的是Floyd算法[1]和Dijkstra算法[2];智能算法包括蚁群算法[3]、粒子群算法[4]、遗传算法[5]、神经网络[6]、模拟退火[7]等。传统的图搜索算法存在随着环境信息的增加计算复杂性呈现指数式增加的缺点,智能算法作为一种路径规划的新思路其对计算机性能要求过高且存在计算时间较长的缺点。A*(A-Star)算法[8-9]是基于传统图搜索的思维的智能启发式算法,相比传统图搜索算法,它具有计算量小、规划路径相对最优等突出特点。但是A*算法的启发函数考虑维度较为简单,导致该算法在寻路过程中会出现很多冗余的扩展栅格。目前已经提出了很多改进A*算法,如:文献[10]中通过估价函数进行指数衰减的方式加权减少了冗余的扩展;文献[11]中通过建立禁忌表来改进A*算法的估价函数,能快速、有效地实现越野路径规划。与文献[10-11]类似地通过对A*算法估价函数进行改进以达到减少计算时间的算法还有很多,但是它们的普适性并不好。文献[12]中通过起点和终点同时运行时效A*算法寻找路径,文献[13]

收稿日期:2019-11-28;**修回日期:**2020-01-13;**录用日期:**2020-01-15。
作者简介:祁玄玄(1994—),男,安徽濉溪人,硕士研究生,主要研究方向:智能机器人、嵌入式计算机;　黄家骏(1996—),男,陕西咸阳人,硕士研究生,主要研究方向:嵌入式计算机;　曹建安(1971—),男,陕西西安人,副教授,博士,主要研究方向:电力电子技术、嵌入式计算机。

中通过并行算法改进A*寻路时间，这两者在一定程度上可减少计算时间，但是都过于依赖计算机的性能。

从上述文献中可以看出，过去A*算法的改进方法主要是通过优化代价函数或提高计算空间来减少计算时间；但这些方法都不具有很高的普适性，并且部分改进方法十分依赖于计算机的性能。本文从四个方面对A*算法进行改进，提出一种兼顾普适性和计算效率的A*算法：首先是优化算法的拓展方向，根据目标和扩展节点的相对位置选择双象限方向进行节点拓展；二是目标可见性判断，目标可见时跳出A*算法的探索过程；其三，通过加入k个父辈估计信息改变算法的启发函数；最后，通过引入模拟退火法改变待扩展节点的选取方略。通过Matlab仿真实验，在栅格地图环境下对比A*算法与改进A*算法的计算结果，验证了本文算法计算时间更短，扩展节点更少，寻路的目标性更强。

1 全局路径规划算法

1.1 A*算法

作为一种图搜索算法，A*算法适合大面积的地图中路径搜索。这是一种以Dijkstra算法和广度优先搜索(Breadth First Search，BFS)算法为基础的启发式搜索算法。因为其具有Dijkstra算法的特点，所以，A*算法可以用于搜索最短路径。与此同时由于A*算法包含BFS的特点，因此它在搜索路径的判断依据中包含启发函数，该函数就是BFS算法之中的得分函数。在路径搜索过程中，A*算法使用代价函数来评估节点的质量。算法将选择的代价函数数值最小的节点作为下一步扩展的节点，然后它将继续从下一个节点搜索，直到到达目标点。像BFS一样，A*可以使用启发式函数来引导自己。A*算法的代价函数如下：

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

其中：n代表路径搜索过程之中最近的一个节点，$g(n)$代表从起始点到n节点的最短路径，$h(n)$代表从n节点到目标节点的启发函数的预测值。$g(n)$经常都是固定的数值，这个数值就是Dijkstra算法中的到起始节点的最短路径；然而$h(n)$却是不固定的，它的改变将会改变优化出来的路径，因此选择合适的$h(n)$可以得出最优的路径。

$h(n)$代表从n节点到目标节点的启发函数的预测值，因此最能表达其含义的就是两点之间的距离，本文采用曼哈顿距离作为$h(n)$函数的估计。其计算表达式如式(2)所示：

$$h(n) = D(\text{abs}(n.x - goal.x) + \text{abs}(n.y - goal.y)) \tag{2}$$

从式(2)可以看出，$h(n)$函数表示n节点到目标节点的x轴和y轴相对距离的绝对值的和。

和Dijkstra算法一样，A*算法在计算过程中也存在两个核心集合。定义已经走过的点的集合为*Close*，待选集合为*Open*，集合里面的每个元素为*node*，该元素的属性是该节点代价函数的估计值$f(n)$。每次从*Open*选出延伸节点，然后往四个方向或者八个方向进行扩展。图1为A*演示模型[14]，其中绿色点是起始点设为A，红色点是目标点设为B，3个蓝色点是障碍物分别设为C、D、E。方格的左上角、左下角、右下角分别代表代价函数的估计值$f(n)$，从起始点到n节点的最短路径$g(n)$，从n节点到目标节点的启发函数的预测值$h(n)$。具体演示流程如下。

1) 初始化集合。

首先把A点加入到*Open*集合，初始化A点代价函数的估计值$f(n)$数值为0，把障碍物加入到*Close*集合。

2) 扩展节点。

从*Open*集合中选取$f(n)$数值最小的点，将其放入*Close*集合。然后往该节点相邻的八个方向扩展方格，并计算不在*Close*集合中的扩展方格的$f(n)$数值。如果扩展方格已经在*Open*集合，那么根据此时计算的扩展方格的$f(n)$数值更新该节点，如果较小则更新，否则不更新。如果扩展方格不在*Open*集合，那么根据计算的结果往*Open*集合中添加该扩展节点。

3) 循环判断。

重复步骤2)，如果扩展到目标节点或者*Open*集合为空则退出循环，根据回溯算法倒推出从起始点到目标点的最短路径。

通过A*算法可以从某一点到另外一点的最短路径，图1中红色圆点连接就是扩展得出的路径。

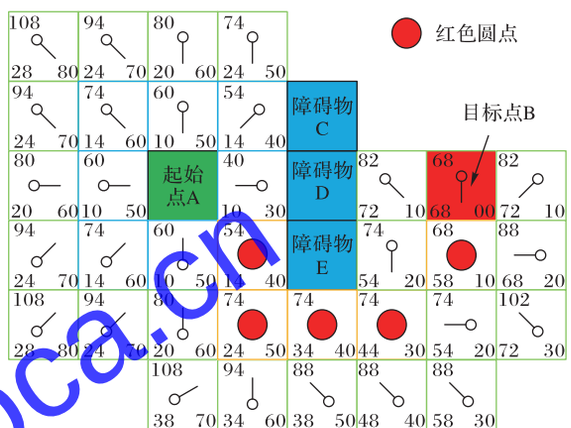


图1 A*演示模型

Fig. 1 A* demonstration model

1.2 A*算法改进

A*算法无疑是一种高效的算法，但传统的A*仍存在一些不足，例如扩展方向盲目性、启发函数过于简单等，导致A*算法计算效率较低。因此，本文从以下四个方面对传统的A*算法进行改进。

1.2.1 自适应扩展方向

在固定的环境中A*在扩展时能够在任何一个方向上扩展，即分别进行上下左右的扩展。而改进的A*算法提出的是确定目标所在的象限，一旦确定目标象限，就会向该象限扩展从而忽略其他没必要的象限。因此改进的A*算法在扩展时只能往上下左右的某一个方向扩展。确定好象限之后就可以确定扩展的相邻节点。由于扩展方向缩减为原先的四分之一，因此在一定程度上可以减少算法的计算时间。

定义当前扩展节点的坐标为$(n.x, n.y)$，目标节点的坐标为$(goal.x, goal.y)$。通过目标节点和扩展节点的做差得到Dx，Dy。即式(3)所示：

$$\begin{cases} Dx = goal.x - n.x \\ Dy = goal.y - n.y \end{cases} \tag{3}$$

根据Dx，Dy的数值本文可以确定扩展方向所在的象限以及所扩展的点。具体判断依据如式(4)所示：

$$\begin{cases} Dx \geqslant 0, Dy \geqslant 0 \Rightarrow \text{目标在第一象限} \\ Dx \leqslant 0, Dy \geqslant 0 \Rightarrow \text{目标在第二象限} \\ Dx \leqslant 0, Dy \leqslant 0 \Rightarrow \text{目标在第三象限} \\ Dx \geqslant 0, Dy \leqslant 0 \Rightarrow \text{目标在第四象限} \end{cases} \tag{4}$$

如图2象限分布所示,当目标在第一象限时,扩展节点的扩展邻居为{1,2,3,4,5};当目标在第二象限时,扩展节点的扩展邻居为{1,2,3,7,8};当目标在第三象限时,扩展节点的扩展邻居为{1,5,6,7,8};当目标在第四象限时,扩展节点的扩展邻居为{3,4,5,6,7}。

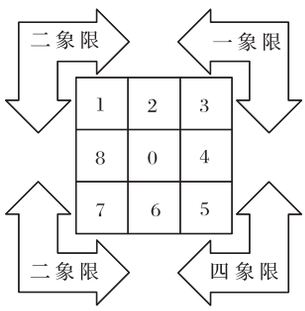


图2　象限分布

Fig. 2　Quadrant distribution

1.2.2　判断目标可见

传统的A*在循环扩展的时候,循环截止条件是靠近目标点或者是*Open*集合为空,这将导致在靠近目标点的时候没有任何障碍物依然会有额外的扩展节点的开销,这显然是不合理的。基于此,本文提出在扩展节点和目标节点之间连线没有任何障碍物的时候,可以跳出A*的启发式探索过程。判断的依据如式(5)所示:

$$Newpos = (n.x, n.y) + r(\cos\varphi, \sin\varphi) \tag{5}$$

其中:r代表更迭步长,φ代表扩展节点和目标节点之间的夹角。*Newpos*代表在栅格地图中的坐标,因此只需要判断*Newpos*位置处是不是存在障碍物就可以判断扩展节点和目标节点之间连线有无任何障碍物。

1.2.3　改变启发函数

代价函数的估计值$f(n)$是从起始点到n节点的最短路径$g(n)$和从n节点到目标节点的启发函数的预测值$h(n)$的和。起始点到n节点的最短路径是在探索过程中从父节点累加的,因此$g(n)$是和前面的探索节点是有关联的。而$h(n)$只是求n节点到目标节点的距离估计值,因此和前面的探索节点是没有任何关联的。基于此本文在代价函数的估计值$f(n)$中添加当前n节点父节点及其祖辈节点的启发函数的预测值$h(n)$。因此代价函数的估计值$f(n)$如式(6)所示:

$$f(n) = g(n) + D \times (h(n) + h(p) + h(p^2) + \ldots + h(p^k)) \tag{6}$$

其中:p是n节点父节点,p^k代表n节点的k父辈,$h(p)$是n节点父节点的启发函数的预测值,D是启发函数的系数。D和k这两个数都会影响A*算法的计算效率,因此选择合适的D和k是十分重要的。

1.2.4　改变扩展节点选取方略

传统的A*算法在从*Open*集合选取待扩展节点的原则是最小的代价函数的估计值的节点。从起始节点到目标节点搜索最短路径时,$g(n)$的数值是从父节点累加过来的,因此它可以尽可能地保证前面走过的路径是最短的。而$h(n)$代表从n节点到目标节点的启发函数的预测值,如果可以尽可能地保证$h(n)$是逐渐减小的,那么在一定程度上就是保证n节点后面的路径是在接近目的地。

为了实现n节点后面的路径是在接近目的地,如果每次都以贪婪策略,即$h(n)$最小值节点来作为扩展节点是最合适的,但是这样就忽略了$g(n)$的影响,该算法也就变成了传统的深度优先搜索了,很有可能优化出来的路径只是局部最优。为了摆脱局部异常的束缚,本文引入模拟退火算法来选择扩展节点。

模拟退火[15]是一种解决无约束和有边界约束的优化问题的方法。该方法模拟了加热材料然后缓慢降低温度以减少缺陷的物理过程,从而最大限度地降低了系统能耗。模拟退火算法包含两个部分,即Metropolis算法和退火过程。

1) Metropolis算法过程。

图3是模拟退火的过程的示意图,A点是迭代起始点,随着迭代次数增加,到达局部最优解B点,此时若根据梯度下降准则再更新的话是不被允许的,而模拟退火算法会在此时以一定的概率跳出这个局限,这个概率和物质能量以及迭代次数息息相关,类似情形通过C点,最终到达全局最优解D点。

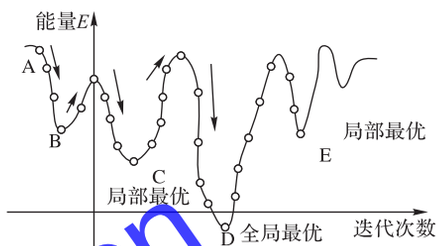


图3　模拟退火的过程

Fig. 3　Simulated annealing process

概率准则如式(7)所示。其中n代表迭代次数,$E(n)$代表第n次迭代的值。

$$P = \begin{cases} 1, & E(n+1) < E(n) \\ -\dfrac{E(n+1) - E(n)}{T}, & E(n+1) \geqslant E(n) \end{cases} \tag{7}$$

从式(7)可以看到:如果能量减小了,那么这种转移就被接受(概率为1);如果能量增大了,就说明系统偏离全局最优值位置更远了,此时算法不会立刻将其抛弃,而是进行概率操作:首先在区间[0,1]产生一个均匀分布的随机数ε,如果$\varepsilon < P$,则此种转移被接受,否则拒绝转移,进入下一步,往复循环。其中P以能量的变化量和T进行决定概率P的大小,所以这个值是动态的,而且随着迭代次数的推移往往P逐渐变小。

2) 退火过程。

退火意思就是温度降低的过程,主要包含初始温度、退火速率和终止温度三个内容。初始温度如果给得太高则获得高质量的解的概率越大,耗费的时间越长。退火速率的设置形式较多,常使用指数式下降型,具体如式(8)所示。其中退火速率r是小于1的数,一般取值[0.8,0.99]。如果在若干次迭代后没有可以更新的新状态或者达到用户设定的阈值,则退火完成,此时的温度称为终止温度。

$$T(n+1) = T(n) \times r \tag{8}$$

根据模拟退火算法的基本特点本文设计选择扩展节点的流程如下。首先把从n节点到目标节点的启发函数$h(n)$视作为模拟退火的能量函数。

1) 初始化各个参数。

初始化初始温度$T = 3$,退火速率$r = 0.98$,能量函数初始数值为起始点到目标节点的启发函数的预测值。扩展初始节点,更新*Close*集合和*Open*集合,即把起始节点放入*Close*集合,根据自适应扩展方向算法把起始点相邻节点放入*Open*

集合。

2) 产生新解 n'。

从 *Open* 集合之中根据最小代价函数的估计值 $f(n)$ 选取新的节点 n'，计算新的节点目标节点的启发函数的预测值 $h(n')$。

3) 计算增量。

因为目标函数差仅由变换部分产生，所以目标函数差的计算最好按增量计算。事实表明，对大多数应用而言，这是计算目标函数差的最快方法。根据式(9)计算能量函数的变化量：

$$\Delta T = h(n') - h(n) \tag{9}$$

4) 判断是否接受当前解 n'。

判断的依据是一个接受准则，最常用的接受准则是 Metropolis 准则：若 $\Delta T < 0$ 则接受 n' 作为新的当前解 n；当 $\Delta T > 0$ 时，产生一个均匀分布的随机数 ε，若 $\exp(-\Delta T/T) > \varepsilon$ 接受 n' 作为新的当前解 n，若 $\exp(-\Delta T/T) \leqslant \varepsilon$ 不接受 n' 作为新的当前解 n，根据最小到目标节点的启发函数的预测值 $h(n)$ 选取新的节点 n'。

5) 更新各个参数。

改变退火温度 $T = T * r$，退火速率 $r = 0.98$，扩展新的节点 n'，更新 *Close* 集合和 *Open* 集合，即把新的节点 n' 放入 *Close* 集合，根据自适应扩展方向算法把新的节点 n' 相邻节点放入 *Open* 集合。

本节从四个方面具体介绍了改进 A*算法的细节。改进后的 A*算法伪代码如下所示：

```
初始化Open集合，Close集合，以及模拟退火相关参数；
while Open集合非空 m
    根据最小f(n)原则从Open集合选取节点n′；
    计算能量函数增量ΔT；
    if ΔT < 0
      接受n作为新的当前解，将n′节点放入Close集合；
    else
      if exp(-ΔT/T) > ε
        接受n′作为新的当前解，将n′节点放入Close集合；
      else
        不接受n′作为新的当前解；
        根据最小化h(n)选取新的节点n′；
      endif
    endif
    if 目标可见
      回溯得到最优路径；
    Endif
    根据式(3)计算Dx, Dy；
    根据式(4)计算n′节点周围的需要计算的节点列表L；
    for each in L
      if each是障碍物
        pass；
      else
        计算each点的f(each), g(each), h(each)
        if each在Open集合
          更新each在Open集合的属性；
        else
          添加each在Open集合；
        endif
      endif
    endfor
endwhile
```

改进后的 A*算法在搜寻从一个源点到另一个源点的最短路径的过程中更加有目的性，可以在一定程度上减少遍历的栅格点的数目，选择更加优化的路径。尤其要注意的是式(6)中 p^k 代表 n 节点的 k 父辈。虽然越多的父辈被加入到代价函数之中可以更加优化寻找的最短路径，但是这样也会增加计算负担和内存，因此需要综合考虑 k 的取值。

2 仿真和分析

为了测试改进的 A*算法，本文采用 26 × 27 的栅格地图作为模拟对象，如图 4 所示。图中黑色部分为障碍物，白色空白部分为可行走地段。在仿真过程中，起始点和目标点都是固定的，障碍物位置也是固定的。在栅格地图坐标系(横轴为从左到右，纵轴为从下到上)下，设置规划起始点为(24，3)。一旦确定目标点将会开始路径规划。本文从三个方面进行评价：一是计算时间；二是优化路径长度；三是经历的栅格数目。

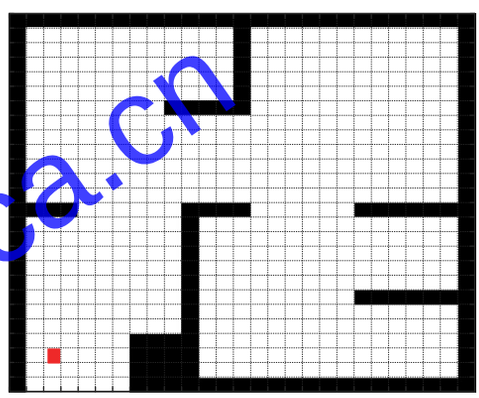


图 4 测试栅格地图

Fig. 4 Grid map for testing

2.1 确定启发函数 D 和 k 的数值

为了确定 D 和 k 的最优数值，本文用传统的 A*算法测试不同 D 和 k 取值时系统规划路径时经历的栅格数目。由于测试算法都是 A*，因此经历越多的栅格那么算法的执行时间也就越长。

图 5 是 D 和 k 取不同数值时经历栅格数目测试图。

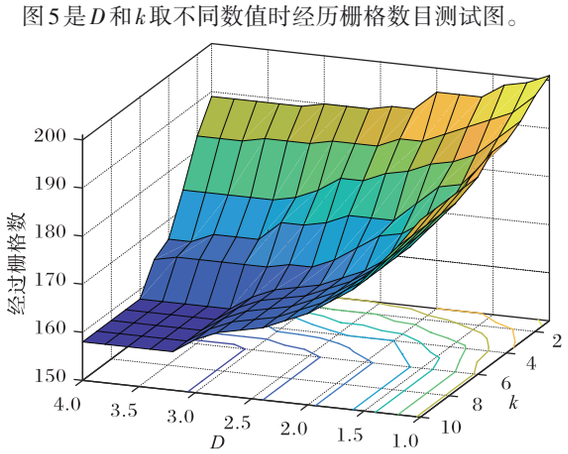


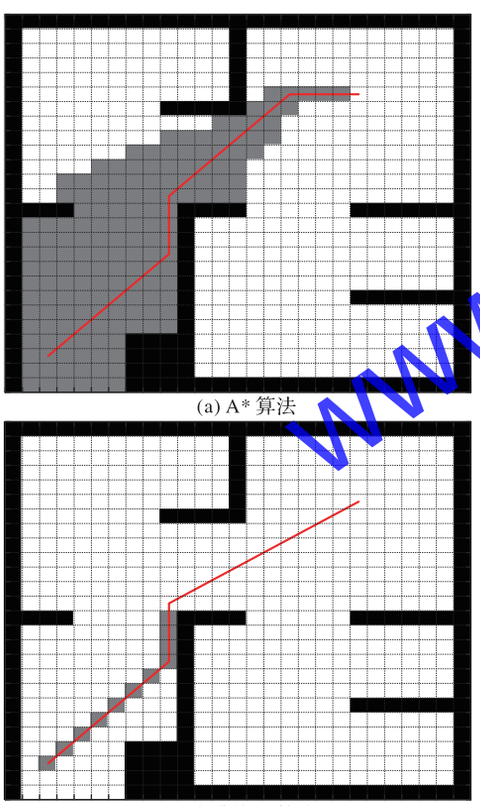
图 5 D 和 k 取不同数值时经历栅格数目测试图

Fig. 5 Test chart of experienced grid number with different D and k values

从图5中可以看出当k数值越大那么算法的执行效率基本也是越高,但是当k增加到第6代的时候算法效率的提升效果就不是很明显了。而更多的父代数据加入进来会增加计算的复杂性,因此本文选定k的取值范围是6。同样的,从图5中可以看出D的数值越大算法的执行效率基本也是越高,但是其在等于3的时候效率提升的效果遇到了瓶颈。所以综合考虑本文选取的D和k的数值分别是3和6。

2.2　改进的A*和A*对比仿真

图6是改进的A*和A*路径规划过程对比图,图中灰色部分代表的是路径规划过程中经历的栅格,红色线条是规划的路径。起点和终点都是一样的分别为(24,3),(6,21)。图6(a)中灰色栅格个数为161,程序运行时间为5.58 s,计算出的最优化路径长度为27.799 0。图6(b)中灰色栅格个数为12,程序运行时间为0.66 s,计算出的最优化路径长度为26.937 9。通过数据可以很清晰地看出改进的A*算法在保证最短路径的前提下,灰色栅格个数和程序计算时间上都有了明显下降,同时优化而出的路径长度也较小于传统A*算法。因此在计算效率上本文提出的算法还是得到很好的印证。



(a) A* 算法

(b) 改进的A*算法

图6　改进的A*算法和A*算法路径规划过程对比

Fig. 6　Comparison of improved A* algorithm and A* algorithm in path planning process

从图6中还能看得出两点:一是改进的A*算法在前一扩展节点指向后一扩展方向的向量与前一节点指向目标点的向量夹角大多数满足锐角关系,这个特点就是自适应扩展方向导致的结果;二是在扩展后半段当目标可见时改进的A*算法可以直接跨过搜索过程。以上两个特点可以一定程度提高A*算法的计算效率。

图7是改进的A*和A*迭代过程对比,从图中可以看出传统的A*算法迭代过程中整体上到目标点的距离是逐渐减小的,但是在局部过程存在震荡,也就是说其局部并没有优化。而改进的A*算法到达至目标点附近所需要的迭代次数大幅减少,且虽然前期仍存在局部震荡,但后期就不存在震荡现象了,这一点正好印证了本文改进的A*措施中的第4)条,即模拟退火过程中,前期温度较高,可以在一定概率上接受局部违背梯度下降的原则,随着温度逐渐降低,这种情况被接受的概率降低到几乎为零的地步。

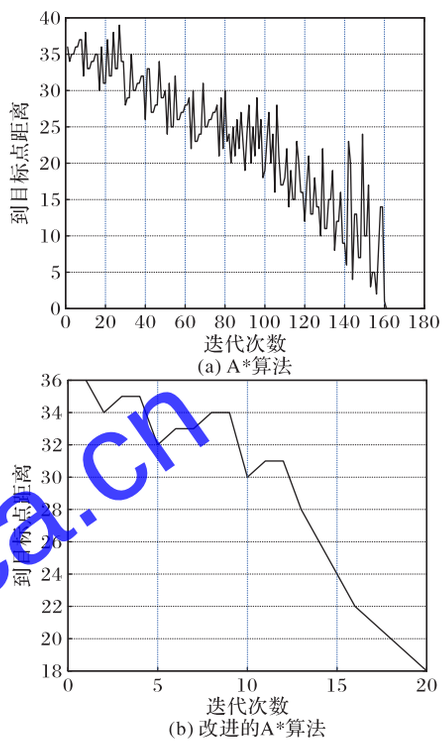


图7　改进的A*算法和A*算法迭代过程对比

Fig. 7　Comparison of improved A* algorithm and A* algorithm in iterative process

为了验证本文算法的普适性,以起始点的三个不同方位作为目标点进行传统A*和改进A*算法的路径规划,表2是不同目标方位统计结果。从表中可以看出,改进的A*算法在运行时间、经历栅格数都少于传统的A*算法,且优化路径长度几乎是一样的。

表2　不同目标方位下的路径规划统计

Tab. 2　Statistics of path planning with different target directions

目标方位	运行时间/s		经历栅格数		优化路径长度/m	
	A*算法	改进A*算法	A*算法	改进A*算法	A*算法	改进A*算法
(6,21)右上	5.58	0.66	61	12	27.799 0	26.937 9
(24,21)右下	8.70	4.25	232	80	32.384 8	34.358 4
(6,3)左上	3.65	1.01	115	17	20.899 5	20.280 1
均值	5.98	1.97	136	36	27.027 8	27.192 1

从平均值的角度分析,本文提出的改进的A*算法在运行时间上减少67.06%,经历的栅格数减少73.53%,优化路径长度浮动范围在±0.6%。

2.3 复合地图下仿真

在传统栅格地图上，通过仿真可以看出，应用本文的改进的A*算法是非常高效的，但是当障碍物较多或者地图面积较大的时候即使是改进的A*算法也会产生很多无用的扩展，这是十分消耗时间的。其次栅格地图的缺点是如果分辨率太高，那么寻路过程会被拖慢，栅格地图如果分辨率太低，或丧失很多关键障碍物信息，那么寻路结果会变得不准确。

基于此问题本文引入多层地图的概念，上层地图是节点和连线组成的拓扑地图，下层是栅格大小相同的栅格地图。图8是复合地图的一个示意图。上层地图由黑色的点和黑色连线组成，分别代表采样路标，以及路标是否连通。下层地图由大小相等的黑白栅格组成，黑色为障碍物，白色为可行走区域。

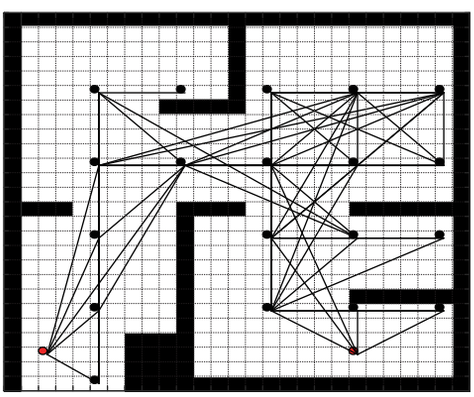


图8 复合地图

Fig. 8 Composite map

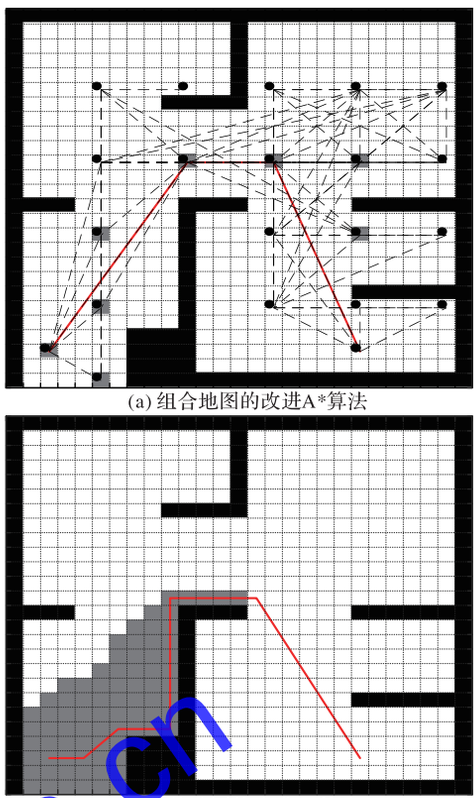
基于复合地图的概念，A*改进算法的扩展规则从原来的栅格的相邻栅格变为拓扑地图下路标的毗邻路标，其他过程和栅格地图是一致的。

图9是在起始点为(24,3)，目标点为(24,21)时，复合地图和栅格地图下使用本文提出的改进的A*算法得到的路径规划图。选择图形右下角作为目标点的原因是其寻路过程较为复杂，因此可以更加清晰测试出复合地图下改进的A*算法的优势。

从图9中可以清晰看到复合地图下经历栅格数只有9，而栅格地图下经历的栅格数达到了80，这将大幅度缩短程序的计算时间。而优化路径的长度几乎是没有差别，复合地图下是34.192 7，栅格地图下是34.385 4。而且组合地图计算的路径更加远离障碍物，一定程度上较为合理。

以起始点的三个方位作为目标点进行不同地图模式下改进的A*算法的路径规划，表3是不同目标方位统计结果。从表3中可以看出在复合地图下改进的A*算法在运行时间、经历栅格数都少于栅格地图下A*算法，且优化路径长度几乎是一样的。从平均值的角度分析，本文提出的在复合地图下运用改进的A*算法在运行时间上减少90%以上，经历的栅格数减少86%左右，优化路径长度浮动范围在±0.1%。

分析其如此高效的原因是复合地图下，改进的A*算法拥有了较大的步长，因此可以更快地寻找到目标点。但是其步长还不能自适应地去改变，因此这也是本课题下一步需要改进的地方。



(a) 组合地图的改进A*算法

(b) 栅格地图的改进A*算法

图9 不同地图模式下改进的A*算法结果

Fig. 9 Results of improved A* algorithm in different map modes

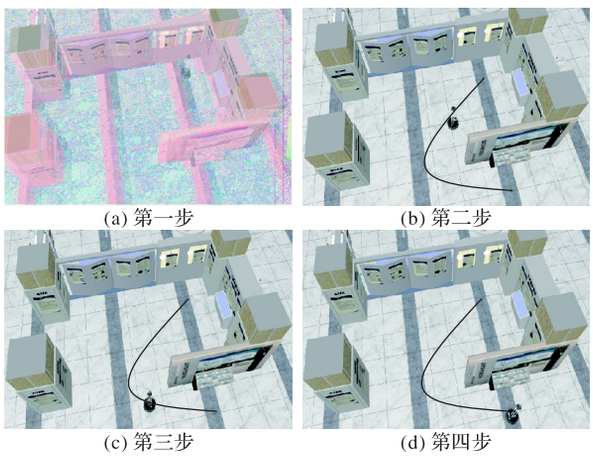
表3 不同目标方位在不同地图模式下的路径规划统计

Tab. 3 Statistical table of path planning with different target directions

目标方位	运行时间/s		经历栅格数		优化路径长度	
	复合地图	栅格地图	复合地图	栅格地图	复合地图	栅格地图
(6,21)右上	0.021	0.66	3	12	27.799 0	26.937 9
(24,21)右下	0.095	4.25	9	80	34.192 7	34.358 4
(6,3)左上	0.020	1.01	3	17	19.172 6	20.280 1
均值	0.045	1.97	5	36	27.054 7	27.192 1

2.4 实际场景验证

为验证本文改进的A*算法的有效性，本文在开源的硬件平台Turtlebot3上进行测试，Turtlebot3是一个小型、低成本、完全可编程的移动机器人。其大小是138 mm×178 mm×192 mm，CPU是32位ARM Cortex-M7，具备里程计、360°激光距离传感器LDS-01和IMU等基本导航设备。测试环境是实验室环境，其范围大概在5 m×4 m的范围，路上没有障碍物。测试的过程是从室内某一点运行到走廊上某一点。图10是实际场景下改进的A*算法结果，图中黑色曲线是全局规划的路线，路径的曲率较小的原因是在A*算法规划出路径后进行了路径平滑处理。通过4步，Turtlebot3可以自主行进到指定位置。实验说明本文提出的改进的A*算法具备实用价值，可以在规避障碍物的条件下高效地进行全局路径规划。



(a) 第一步　(b) 第二步　(c) 第三步　(d) 第四步

图10　实际场景下改进的A*算法结果

Fig. 10　Results of improved A* algorithm in real scene

3　结语

在进行无人车路径规划设计时,传统A*算法存在一些不足,比如扩展方向盲目性、启发函数只考虑当前节点的信息等,导致A*算法寻路过程包含很多冗余的节点,计算效率较低。本文从四个方面改进传统A*算法:首先越靠近目标节点越以目标节点的估计距离作为选取待扩展节点的原则,为了实现这个目的,本文引入根据模拟退火从*Open*列表中选取待扩展节点以避免陷入局部最优解;其次改进的A*算法的启发函数,加入了n个父辈的信息,以此作为启发寻路的"历史经验";然后根据待扩展节点和目标点相对位置选择扩展象限,以此来避免待扩展节点盲目往四面八方扩展;最后判断目标可见时跳出寻路过程,以此避免接近目标点时还存在盲目的冗余扩展。

在Matlab环境下,对不同目标方位下改进的A*算法和A*算法的路径规划算法进行仿真,结果显示,本文提出的改进的A*算法在运行时间上减少67.06%,经历的栅格数减少73.53%,优化路径长度浮动范围在±0.6%。可以看出本文改进的A*算法在不同情况下都能够有效地缩短寻路时间,减少冗余扩展,算法的计算效率较高。

参考文献(References)

[1] TSITSIASHVILI G S, LOSEV A S. Application of the Floyd algorithm to the asymptotic analysis of networks with unreliable ribs[J]. Automation and Remote Control, 2008, 69(7):1262-1265.

[2] 李泽文,唐平,曾祥君,等. 基于Dijkstra算法的电网故障行波定位方法[J]. 电力系统自动化, 2018, 42(18):162-168.(LI Z W, TANG P, ZENG X J, et al. Method of traveling wave fault location based on Dijkstra algorithm in power grid[J]. Automation of Electric Power Systems, 2018, 42(18): 162-168.)

[3] CHEN R M, HSIEH F R, WU D S. Heuristics based ant colony optimization for vehicle routing problem[C]// Proceedings of the 7th IEEE Conference on Industrial Electronics and Applications. Piscataway: IEEE, 2012: 1039-1043.

[4] 于立婷,谭小波,吴艳梅. SDN网络中基于改进粒子群的最优路径规划算法[J]. 沈阳理工大学学报, 2019, 38(2):20-25.(YU L T, TAN X B, WU Y M. Optimal path planning algorithm based on improved particle swarm optimization under the SDN architecture [J]. Journal of Shenyang Ligong University, 2019, 38(2):20-25.)

[5] 李俊,舒志兵. 基于改进D* Lite遗传算法路径规划研究[J]. 机床与液压, 2019, 47(11):39-42.(LI J, SHU Z B. Research on path planning based on improved D* Lite genetic algorithm[J]. Machine Tool and Hydraulics, 2019, 47(11):39-42.)

[6] 禹建丽,李晓燕,王跃明,等. 一种基于神经网络的机器人路径规划算法[J]. 洛阳工学院学报, 2001, 22(1):31-34.(YU J L, LI X Y, WANG Y M, et al. An algorithm of path planning for car-like robots based on neural network[J]. Journal of Luoyang Institute of Technology, 2001, 22(1): 31-34.)

[7] 巩敦卫,曾现峰,张勇. 基于改进模拟退火算法的机器人全局路径规划[J]. 系统仿真学报, 2013, 25(3):480-483, 488.(GONG D W, ZENG X F, ZHANG Y. Global path planning method of robot based on modified simulated annealing algorithm [J]. Journal of System Simulation, 2013, 25(3):480-483, 488.)

[8] 贾庆轩,陈钢,孙汉旭,等. 基于A*算法的空间机械臂避障路径规划[J]. 机械工程学报, 2010, 46(13):109-115.(JIA Q X, CHEN G, SUN H X. Path planning for space manipulator to avoid obstacle based on A* algorithm[J]. Journal of Mechanical Engineering, 2010, 46(13):109-115.)

[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.(WANG D J. Indoor mobile-robot path planning based on an improved A* algorithm [J]. Journal of Tsinghua University (Science and Technology), 2012, 52(8): 1085-1089.)

[10] 王维,裴东,冯境,等. 改进A*算法的移动机器人最短路径规划[J]. 计算机应用, 2018, 38(5): 1523-1526.(WANG W, PEI D, FENG Z. The shortest path planning for mobile robots using improved A* algorithm [J]. Journal of Computer Applications, 2018, 38(5): 1523-1526.)

[11] 吴天羿,许继恒,刘建永,等. 基于改进A*算法的越野路径规划研究[J]. 计算机应用研究, 2013, 30(6): 1724-1726.(WU T Y, XU J H, LIU J Y, et al. Research of cross-country path planning based on improved A* algorithm[J]. Application Research of Computers, 2013, 30(6): 1724-1726.)

[12] 高民东,张雅妮,朱凌云. 应用于机器人路径规划的双向时效A*算法[J]. 计算机应用研究, 2019, 36(3): 792-795, 800.(GAO M D, ZHANG Y N, ZHU L Y. Bidirectional time-efficient A* algorithm for robot path planning[J]. Application Research of Computers, 2019, 36(3): 792-795, 800.)

[13] 熊壬浩,刘羽. A*算法的改进及并行化[J]. 计算机应用, 2015, 35(7):1843-1848.(XIONG R H, LIU Y. Improvement and parallelization of A* algorithm[J]. Journal of Computer Applications, 2015, 35(7):1843-1848.)

[14] PATRICK. A* pathfinding for beginners[EB/OL]. [2019-03-15] https://www.jianshu.com/p/e52d856e7d48.

[15] 冯玉蓉. 模拟退火算法的研究及其应用[D]. 昆明:昆明理工大学, 2005.(FENG Y R. Study and application of simulated annealing algorithm[D]. Kunming: Kunming University of Science and Technology, 2005.)

QI Xuanxuan, born in 1994, M. S. candidate. His research interests include intelligent robot, embedded computer.

HUANG Jiajun, born in 1996, M. S. candidate. His research interests include embedded computer.

CAO Jianan, born in 1971, Ph. D., associate professor. His research interests include power electronic technology, embedded computer.