改进A*算法的水面舰艇静态航路规划

武善平,黄炎焱,陈天德
南京理工大学 自动化学院,南京 210094

摘 要:针对复杂海洋环境下水面舰艇航路规划时出现的大地图寻路速度慢、航路安全性差、航路不平滑等难题,结合电子海图提出了一种改进A*算法的航路规划方法。提出一种自适应的改进启发函数,在搜索节点时加入目标节点的方位信息,加快了A*算法搜索路径的速度;加入迫使航路远离障碍物的安全距离,解决了传统A*算法沿障碍物边缘寻路导致航路安全性差的问题;对原始航路进行二次优化,在对原始路径提取转折点后,通过判断任意两个转折节点的直线可达性,将转折节点之间的实际距离转化为距离矩阵,使用Dijkstra算法优选出航路长度更短的关键转折点,最终使用二阶贝塞尔曲线对航路转折处进行平滑处理,以满足航路平滑且易跟随的要求。仿真实验表明,相对于传统A*算法,改进算法规划的路径具有寻路速度更快、航路距离更短、航路安全性更高的特点。

关键词:A*算法;自适应启发函数;Dijkstra算法;贝塞尔曲线;电子海图
文献标志码:A **中图分类号:**TP391 doi:10.3778/j.issn.1002-8331.2106-0025

Static Route Planning of Surface Ships Based on Improved A* Algorithm

WU Shanping, HUANG Yanyan, CHEN Tiande
School of Automation, Nanjing University of Science and Technology, Nanjing 210094, China

Abstract: Aiming at the problems of slow path-finding on large maps, poor route safety, and unsmooth route during route planning of surface ships in complex ocean environments, this paper proposes a route planning method with improved A* algorithm combined with electronic charts. First, an adaptive and improved heuristic function is proposed. When searching for nodes, the location information of the target node is added to speed up the search path of the A* algorithm. Second, the safe distance forcing the route to stay away from obstacles is added to solve the traditional A*. The algorithm finds the route along the edge of obstacles, which leads to the problem of poor air route safety. Finally, the original route is optimized twice. After the turning point is extracted from the original path, the straight line reachability of any two turning nodes is judged, and the turning nodes are separated. The actual distance is converted into a distance matrix. The Dijkstra algorithm is used to optimize the key turning points with a shorter route length, and finally the second-order Bezier curve is used to smooth the route turning points to meet the requirements of smooth and easy-to-follow routes. Simulation experiments show that, compared with the traditional A* algorithm, the path planned by the improved algorithm has the characteristics of faster path finding, shorter route distance, and higher route safety.

Key words: A* algorithm; adaptive heuristic function; Dijkstra algorithm; Bezier curve; electronic chart

由于海洋环境的复杂性,航路规划是船舶智能引导的重要组成部分,主要功能是在已知船舶准确位置以及周围静态障碍物信息的环境中,搜索一条从起点到终点的、满足一定要求的航行路径,使船舶在航行过程中能够安全可靠地避开所有障碍区域并且使航路尽可能短、尽可能平滑。

路径规划问题较早应用于自主移动机器人,通过自动推理、全局规划、移动控制等过程,实现机器人的自主导航移动。A*算法是一种控制性能好且发展成熟的方法,是移动机器人全局路径规划的常用算法之一[1]。在静态场景中,A*算法能更快更有效地求解出最优路径,因此被广泛用于未知环境的全局路径规划,但也存在计算量大、效率低、路径拐点不平滑等缺陷[1]。

在静态环境下,A*算法具有最优性、完备性,但在

基金项目:共用信息系统预研项目(315055106);重点实验室基金(JZX7Y202001SY000901)。
作者简介:武善平(1997—),男,硕士研究生,研究领域为两栖兵力登陆规划建模与仿真,E-mail:wu_shanping@foxmail.com;黄炎焱(1973—),通信作者,男,博士,教授,研究领域为系统/体系建模与仿真、效能及体系贡献率分析、指挥控制及优化;陈天德(1994—),男,博士研究生,研究领域为有人无人协同作战规划。
收稿日期:2021-06-02 **修回日期:**2021-07-19 **文章编号:**1002-8331(2022)23-0307-09

随着地图数据变大搜索速度变慢。很多学者对A*算法进行许多改进,探索更高效的算法。文献[2]通过增加$h(n)$的权重来提高A*算法的搜索性能;文献[3]将栅格模型优化为无障碍、静态障碍以及动态障碍三种移动环境下进行路径规划;文献[4]提出一种动态改变步长的快速A*算法;文献[5]提出一种基于关键点选取的策略来代替传统A*算法中的Openlist和Closelist两个列表;文献[6]通过加入引导量对启发函数进行改进,减少具有相同估价值的网格数量来优化算法搜索效率;文献[7]提出了一种对启发函数$h(n)$结合了分区距离信息和角度变量信息加权的改进启发函数;文献[8]在改进启发函数中加入了n个父辈的信息,并根据待扩展节点和目标点相对位置选择扩展象限;文献[9]在启发函数中引入当前节点的父节点启发函数;文献[10-11]引入了双向搜索机制,以原始起点、终点和对向搜索所处的当前节点作为目标点进行搜索操作。

针对A*算法在大地图中复杂障碍物环境下存在数据量大、寻路效率低、航路不平滑等问题,本文提出一种高效的改进A*算法。首先,改进A*算法的启发函数,加入自适应的启发信息,以提高寻路效率;然后,加入安全距离,保证了航路的安全性;最后,对原始路径进行二次优化,进一步缩短了航路的长度并提高了航路的平滑性。

1 传统A*算法

1.1 地图建模

地图建模是路径规划的重要环节,目的是将实际物理环境抽象成便于计算机存储和处理的地图模型,实现从物理环境到虚拟地图的映射[12-13]。在常用的地图建模方法中,栅格地图具有简单有效、易于实现、便于更新的特点,是目前研究和应用最广泛的方法。

传统A*算法一般采用正方形栅格来进行地图建模[14],在正方形栅格地图中,当前节点的邻居节点通常为8邻域,节点可以沿水平、竖直或对角线方向移动。如图1所示,地图中包含障碍物的栅格标记为障碍栅格(黑色栅格),表示不可通行的区域;地图中不包含障碍物的栅格标记为自由栅格(白色栅格),表示可以通行的区域。

为利用真实的海洋环境实现全局航路规划,通过解析电子海图提取的海洋环境信息通常由复杂几何图形组成,多数路径规划算法不能直接使用[15-16]。因此,需要采用栅格法建立基于电子海图的海洋环境模型。首先解析电子海图,提取其中的海域地理信息。然后利用正方形栅格网格划分进行网格化,建立由可航行网格和不可航行网格组成的海洋环境模型。对电子海图的海洋环境信息进行栅格化可以提高路径规划算法的效率[4,6-7]。

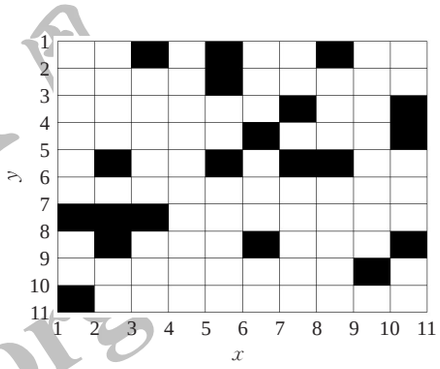


图1 栅格地图

Fig.1 Raster map

1.2 传统A*算法

A*算法是全局路径规划中一种启发式搜索算法,在Dijkstra算法的基础上引入了启发式函数$h(n)$[3],$h(n)$表示了当前节点到目标节点的估计代价。在保证了路径最优性的同时,加入了目标节点的距离信息,提升了搜索效率。其估价函数表示为:

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

式中,$g(n)$表示累积代价,即对从初始节点到节点n的累计真实代价;$h(n)$表示目标代价:即节点n到目标节点的估计代价;$f(n)$表示估价函数,即从初始节点经过当前节点n,再到目标节点的估计代价。

1.3 算法流程

A*算法通过代价估计函数$f(n)$进行路径规划,从初始节点向附近邻居节点进行扩展,将不超过地图的非障碍物邻居节点加入OpenList。然后选取OpenList中$f(n)$值最小的节点选为下一父节点,并将搜索过的节点加入CloseList。重复此过程直到搜索到目标节点,然后沿着父节点的方向进行回溯生成最终路径,完成路径规划[2-4]。传统A*算法的伪代码如下所示:

算法1 A*

Input:node *star*,node *goal*,map[]

Output:path []

1. Begin:
2. *OpenList* =[], *CloseList* =[], *path* =[];
3. add *star* to *OpenList*;
4. While *OpenList* is NOT NULL do
5. get node n having $\min(f(n))$ in *OpenList*;
6. if n is *goal* then break;
7. if $map[neighbor_i]$ is NOT Obstacle then
8. $f(neighbor_i) \leftarrow g(neighbor_i)+h(neighbor_i)$;
9. $parent(neighbor_i) \leftarrow n$;
10. add $neighbor_i$ to *OpenList*;

```
    end
    delete n from OpenList;
    add n to CloseList;
end
if n is goal then
    while n in CloseList and n is NOT star do
        add n to path;
        n ← parent(n);
    end
end
print path;
End
```

A*算法在扩展邻居节点时会把当前节点所有邻居节点都考虑进去，该过程中会生成大量与最终路径无关的节点，或者导致能够通过但是花费的代价较高的情况，从而消耗大量时间，所以提高路径规划效率是本文的研究点之一。

2　改进A*算法

传统A*算法能够对目标点进行有效的全局路径规划，但所规划出的路径存在冗余节点、转折点多、距离障碍物过近且转折角度过大等问题。针对这些问题，从3个方面对A*算法进行改进。一是改进估价函数加快航路搜索速度；二是加入安全距离来保证航路的安全性；三是删除冗余节点然后使用贝塞尔曲线平滑转折节点来减少航路转折次数和航路长度，增加航路的平滑度。

2.1　改进启发函数

在A*算法中，用于估计节点 n 的启发代价值 $h(n)$ 决定着A*算法对其他节点的扩展速度及所扩展节点是否合理。$h^*(n)$ 是指节点 n 到目标节点的真实代价，$h(n)$ 与 $h^*(n)$ 的大小影响算法的精度和速度。A*算法能够找到最短路径的原则为 $h(n)$ 的值不大于 $h^*(n)$ [1]。

当 $h(n)=h^*(n)$ 时，A*算法可以兼顾精度和速度，是A*算法的最优状态，但这个 $h(n)$ 一般不容易找到；当 $h(n)<h^*(n)$ 时，A*算法搜索效率略低，$h(n)$ 越小，意味着需要扩展的节点就越多，效率上越低，但是精度上越准确。当 $h(n)=0$ 时，A*算法退化为Dijkstra算法[7]。

2.1.1　节点距离

传统A*算法一般采用欧氏距离或曼哈顿距离来计算 $h(n)$ [9]，但是由于欧氏距离计算的 $h(n)$ 很多情况下都小于栅格地图的真实目标代价 $h^*(n)$，势必会扩展很多不必要的节点，且往往造成所计算的距离过大或过小。为此，本文采用一种更加适合栅格地图的距离计算方法。

对于地图上任意两个不同坐标节点 $P_1(x_1,y_1)$ 和 $P_2(x_2,y_2)$，定义 x 轴和 y 轴的绝对距离分别为：

$$\begin{cases} \mathrm{d}x=|x_1-x_2| \\ \mathrm{d}y=|y_1-y_2| \end{cases} \tag{2}$$

在无障碍物环境中，栅格地图中的两个节点之间的移动距离都可以用图2中两条折线的距离相加求得。

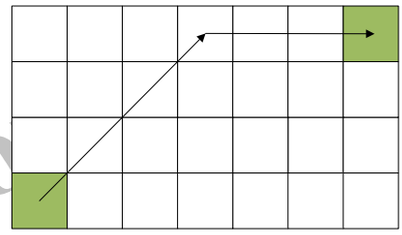


图2　栅格地图节点间实际距离

Fig.2　Actual distance between grid map nodes

在栅格地图中，一般设置横向和纵向移动一个单元的距离为1，斜向移动一个单元的距离为 $\sqrt{2}$。很容易得到两个节点坐标之间的距离计算公式为：

$$dis=(\sqrt{2}-1)\min(\mathrm{d}x,\mathrm{d}y)+\max(\mathrm{d}x,\mathrm{d}y) \tag{3}$$

这种距离也叫切比雪夫距离或棋盘距离。

2.1.2　目标方位信息

由A*算法具体搜索过程可知，导致其效率低的一个重要原因如图3所示：对于无障碍物情况下，真实目标代价 $h^*(n)$ 确定的路径是闭氏解，往往存在多条等价路径，而实际上从起点到终点只需要取其中一条路径即可。选择 $f(n)$ 值最小的点作为下一个扩展节点时，总是会出现往返搜索的情况。

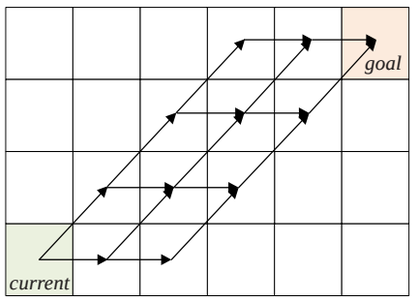


图3　闭式解路径

Fig.3　Closed-form solution path

对于这种情形，可以考虑加入目标节点的方位代价来打破对称性，让A*算法具有保持当前方向的倾向性，可以减少扩展没必要的节点并平滑路径。其次，为提高运动平稳性，在路径规划过程中，不仅要减小转角，也要尽可能减少机器人转向次数。

传统A*算法中启发信息 $h(n)$ 只包含了当前点到目标节点的距离信息，这也导致出现多条冗余路径，是A*算法在大地图数据量爆炸时寻路速度变慢的原因之一。在任意节点扩展邻居节点的时候，不仅希望路径的长度最短，还总是希望路径能够倾向朝着目标节点的前进。基于此本文在启发信息中加入当前节点到目标节

点的方位信息,来引导A*算法偏向拓展分布在当前节点到目标点连线附近的节点。

取当前节点 (x_{node},y_{node}) 指向目标节点 (x_{goal},y_{goal}) 的引导向量 a 为:

$$a=\left(x_{goal}-x_{node},y_{goal}-y_{node}\right) \tag{4}$$

如图4所示,当前节点指向邻居节点的向量为可行向量 t_i。引导向量 a 与可行向量 t_i 的夹角 θ 可以用来表示目标节点的方位信息。夹角 θ 越小,说明该邻居节点位于更靠近目标节点的方位;反之,则说明该邻居节点位于更远离目标节点的方位。

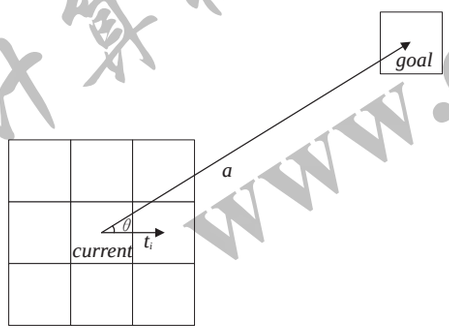


图4　引导向量与可行向量的夹角

Fig.4　Angle between guiding vector and feasible vector

考虑到计算方便以及cos函数在夹角取值范围内的单调性,本文采用引导向量 a 与可行向量 t_i 的夹角 θ 余弦值来计算方位信息,其计算公式为:

$$\cos\theta=\frac{a\times t_i}{|a|\cdot|t_i|} \tag{5}$$

对于栅格地图中搜索路径的任意节点都可以分为两种情况,前方有障碍物和前方没有障碍物,如图5所示。

本文根据寻路过程将节点的状态分为前方存在障碍物和无障碍物两个状态。区分方法就是,从当前节点沿着引导向量的方向前方距离为 d_{min}(一般是给定的安全距离)范围内的节点 (x_i,y_i),判断这些节点中是否存在障碍物:

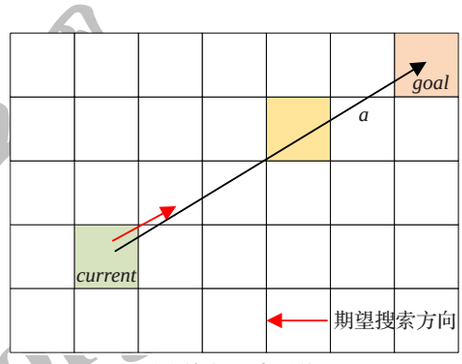
$$\begin{cases}x_i=x_{node}+round\left(\dfrac{i\times(x_{goal}-x_{node})}{\max(|x_{goal}-x_{node}|,|y_{goal}-y_{node}|)}\right)\\ y_i=y_{node}+round\left(\dfrac{i\times(y_{goal}-y_{node})}{\max(|x_{goal}-x_{node}|,|y_{goal}-y_{node}|)}\right)\end{cases},i=1,2,\cdots,d_{min} \tag{6}$$

式中,*round* 表示四舍五入函数。

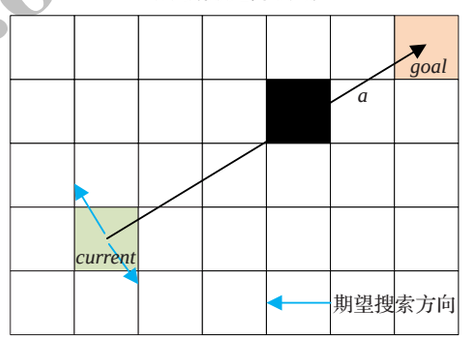
如图5(a)所示,在无障碍环境中(预测节点 (x_i,y_i) 中不存在障碍物),为了加快搜索航路的速度,搜索能力偏向深度优先搜索,搜索节点的期望方向总是向着目标点的方向(红色箭头方向)进行,这时可以设置启发信息为:

$$p(n)=-\cos\theta \tag{7}$$

如图5(b)所示,在遇到障碍物时(预测节点 (x_i,y_i) 中存在障碍物),为了加快绕开障碍物,搜索能力偏向广度优先搜索,搜索节点时的期望方向总是向着引导向量的两侧方向(蓝色箭头方向)进行,这时可以设置启发信息为:



(a)前方无障碍物



(b)前方有障碍物

图5　栅格地图中的两种情况

Fig.5　Two situations in grid map

$$p(n)=-|\cos\theta| \tag{8}$$

可以得到栅格地图中的自适应启发函数为:

$$p(n)=\begin{cases}-\cos\theta, & \text{前方无障碍物}\\ -|\cos\theta|, & \text{前方有障碍物}\end{cases} \tag{9}$$

综上,改进的A*算法估价函数为:

$$f(n)=g(n)+dis(n)+p(n) \tag{10}$$

为了验证改进启发函数搜索路径的快速性,使用传统A*算法和改进A*算法进行了50次随机实验,在相同地图中的起点和终点下,仿真时间对比如图6所示。

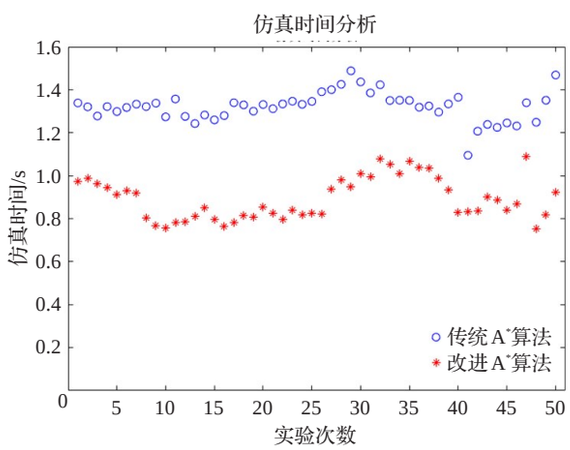


图6　算法时间对比

Fig.6　Comparison of algorithm pathfinding time

从图6可以看出，改进启发函数后的算法都比传统A*算法更快，具有更加高效的寻路速度。

2.2　安全距离

在传统A*算法中，航路存在大量紧贴着障碍物的危险节点。若在实际航路跟踪时，舰艇沿着障碍物边缘行进时，安全性较差。针对这个问题，在搜索航路时需要设置一定的安全距离，迫使航路远离障碍物。

本文提出一种保持航路与障碍物安全距离的方法。在对某个节点搜索邻居节点时，若已经判断该邻居节点可以扩展，则沿着当前方向 t_i 继续搜索 $d_{\min}$（最短安全距离）步，判断到达的节点是否为障碍物。若该节点是障碍物，则直接放弃这个邻居节点；若不是，则将该邻居节点加入OpenLsit。这样可以使得航路跳过那些距离障碍物小于安全距离的节点，保证了航路的安全性。

2.3　二次航路优化

在传统的A*算法中，路径只进行一次规划，得到航路是一条在A*算法模型约束条件下的最优路径。由于在搜寻节点的过程中限制了节点的移动方向，只能向周边八个点进行扩展搜索，使得路径中依然存在着冗余路径点，并且转折次数多且很多拐点处转向难，使舰艇行驶转弯效率低下，不能很好地跟踪所规划的路径。因此对原始航路进行二次优化，对冗余节点进行删除，对转折处进行平滑，获得由起点、终点和关键转折点组成的平滑航路。

2.3.1　提取路径转折点

A*算法得到的原始路径存在大量的冗余节点，占据了大部分的非必要内存。因此，需要对路径转折点进行提取，只保留起点、终点和路径转折点。若原始航路在某一段路径上保持同一个方向趋势不变时，可以认为除了两端之外的节点都是冗余节点。遍历每一个航路节点，计算父节点进入它的相对方向（子节点在父节点邻域中编号），起点的方向默认为0。

若航路是一条线段时，即在一段路径上航路的方向保持不变并且航路长度大于阈值时，则保留这段航路的起点和终点；若航路是一条折线段时，即在一段路径上节点的方向出现反复震荡，则判断震荡的起点和终点是否能直线到达，若可以，则保留震荡航路的起点和终点。最终获得只包括起点、终点和转折点的航路。

2.3.2　优选关键转折点

由起点、终点和转折点表示的航路可以大幅度减少航路的冗余节点，但还在冗余转折点会导致航路不必要转弯，航路的实际距离并不是最优。如图7所示，黑色航路是在栅格地图中规划的最优航路；而在实际应用中，若两节点直线可达，舰艇只需沿着红色航线的路线便可进一步减少转向次数和航程。因此，需要采用Dijkstra算法来删除冗余转折点，优选关键转折点，减少航路长度和航路转折次数。

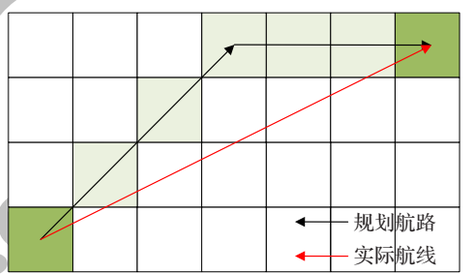


图7　栅格地图航路与实际航线对比

Fig.7　Comparison of grid map route and actual route

Dijkstra（迪杰斯特拉）算法是典型的单源最短路径算法，用于计算一个节点到其他所有节点的最短路径。主要特点是以起始点为中心向外层层扩展，直到扩展到终点为止。

为了进一步优化航路，对提取的转折点之间进行直线可达性判断。遍历每一个转折节点，若两个转折节点之间可以直线到达，则记录它们之间的实际距离（欧式距离）；若不是直线可达，则记录它们的距离为无穷大。将所有可以直线互通的节点距离转换为一个无向图（距离矩阵）。然后使用Dijkstra算法删除航路中不必要的转折点，优取从起点到终点的最短路径的关键转折点。这样进一步缩短航路长度，并减少航路的转折次数。

2.3.3　贝塞尔曲线平滑路径

由关键转折点组成的路径转折次数大幅度减小，当由于转折处都是折线，还存在且拐点处转向难、转弯效率低下的问题。因此，本文采用贝塞尔曲线对由关键转折点进行平滑处理。

贝塞尔曲线由于操作简单且有极强的图像平滑能力，因此在图形设计和路径规划中的应用都非常广泛。贝塞尔曲线完全由其控制点（端点）决定其形状，n 个控制点对应着 $n-1$ 阶的贝塞尔曲线。通用的 n 阶贝塞尔曲线的公式为：

$$\begin{aligned} B(t)=&\sum_{i=0}^{n}\binom{n}{i}P_i(1-t)^{n-i}t^i=\\ &\binom{n}{0}P_0(1-t)^n t^0+\binom{n}{1}P_1(1-t)^{n-1}t^1+\cdots+\\ &\binom{n}{n}P_1(1-t)^0 t^n,\ t\in[0,1] \end{aligned} \tag{11}$$

式中，n 是阶数，t 是参数，取值范围是[0，1]。

为了保证航路在拐点处能够平滑地转向并且不失真，本文使用二阶贝塞尔曲线对关键转折点的每一个转折处进行平滑处理。

如图8所示，二阶贝塞尔曲线需要三个端点来控

制,对应航路上构成一个转折处的三个节点 P_0 、P_1 和 P_2 。在线段 P_0P_1 和 P_1P_2 分别取两个动点 P_0^1 和 P_1^1 ,并使它们满足如下比例关系:

$$\frac{P_0P_0^1}{P_0^1P_1}=\frac{P_1P_1^1}{P_1^1P_2} \tag{12}$$

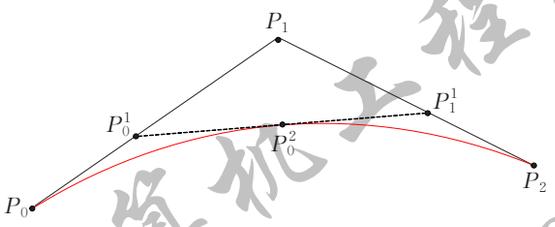


图8　二阶贝塞尔曲线

Fig.8　Second-order Bezier curve

然后过点 P_0 和 P_2 做一条与 $P_0^1P_1^1$ 相切的抛物线,相切点为 P_0^2 ,并满足如下比例等式:

$$\frac{P_0P_0^1}{P_0^1P_1}=\frac{P_1P_1^1}{P_1^1P_2}=\frac{P_0^1P_1^1}{P_0^2P_1^1} \tag{13}$$

引入参数 t ,令式(13)的比值为 t:$(1-t)$,则:

$$\begin{cases}P_0^1=(1-t)P_0+tP_1\\P_1^1=(1-t)P_1+tP_2,\ t\in[0,1]\\P_0^2=(1-t)P_0^1+tP_1^1\end{cases} \tag{14}$$

当 t 从0变到1,即点 P_0^1 向点 P_1 移动、点 P_1^1 向点 P_2 移动时,点 P_0^2 的移动轨迹就是由三节点 P_0 、P_1 和 P_2 三点确定的一条二阶Bezier曲线,这就是式(11)的二阶形式:

$$B(t)=P_0^2=P_0(1-t)^2+2P_1t(1-t)^2+P_2t^2,\ t\in[0,1] \tag{15}$$

二次航路优化的总体效果如图9所示,绿色节点是原始航路,蓝色节点是提取的路径转折点,红色节点是由Dijkstra算法筛选的关键转折点,红色路径是使用贝塞尔曲线平滑的最终路径。可以看到经过二次优化后的航路转折次数大幅减少,缩短了路径长度,提高了轨迹的平滑度,且增加了航路与障碍物之间的距离。

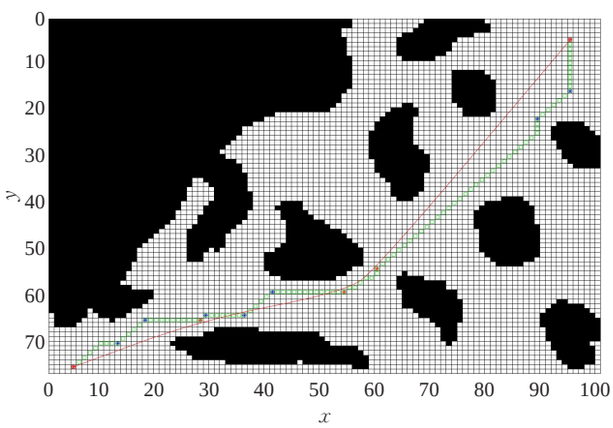


图9　二次航路优化效果

Fig.9　Optimization of secondary route

2.4　算法流程

改进算法在传统A*算法的基础首先增加的非障碍邻居节点是否满足安全距离判断,筛选出大于安全距离的非障碍邻居节点;其次增加了搜索路径时当前节点前方有无障碍物判断,并根据判断结果采用对应的公式计算非障碍邻居节点的 $f(n)$,并记录父节点进入当前节点的相对方向;最后,对原始航路进行二次优化,包括根据节点的父节点进入方向提取航路转折点、判断任意两个航路转折点的直线可达性生成距离矩阵,根据距离矩阵采用Dijkstra算法删除冗余航路转折点、对优选后的航路转折点采用二阶贝塞尔曲线进行平滑处理等操作。改进算法伪代码如下所示:

算法2 Improved A*

Input:node *star* ,node *goal* ,map[], $d_{\min}$

Output:path []

Begin:
OpenList =[], *CloseList* =[], *path* =[], *distance* =[];
add *star* to *OpenList* ;
While *OpenList* is NOT NULL do
　get node n having $\min(f(n))$ in *OpenList* ;
　if n is *goal* then break;
　if $map[neighbor_i]$ and $map[neighbor_i+d_{\min}\times\boldsymbol{t}_i]$ are NOT Obstacles then
　　if $map[(x_i,y_i)]$ is Obstacle then
　　$f(neighbor_i)\leftarrow g(neighbor_i)+dis(neighbor_i)-|\cos\theta|$;
　　else
　　$f(neighbor_i)\leftarrow g(neighbor_i)+dis(neighbor_i)-\cos\theta$;
　　end
　　$parent(neighbor_i)\leftarrow n$;
　　$direction(n)\leftarrow \text{angle}(parent(n),n)$;
　　add $neighbor_i$ to *OpenList* ;
　end
　delete n from *OpenList* ;
　add n to *CloseList* ;
end
if n is *goal* then
　while n is not *star* do
　　add n to *path* ;
　　$n\leftarrow parent(n)$;
　end
end
for $n, n_{i+1}, \cdots, n_{i+k}$ in *path* do
　if $direction(n_i)=direction(n_{i+1})=\cdots=direction(n_j)$ then

28. delete $direction(n_{i+1}), \cdots, direction(n_{i+k-1})$ from $path$;

29. end

30. if $direction(n_i) \neq direction(n_{i+1}) \neq \cdots \neq direction(n_{i+k})$ and (n_i and n_{i+k} are reachable) then

31. delete $direction(n_{i+1}), \cdots, direction(n_{i+k-1})$ from $path$;

32. end

33. end

34. for n_i and n_j in $path$ do

35. if n_i and n_j are reachable

36. $distance(n_i, n_j) \leftarrow \sqrt{n_i^2 + n_j^2}$;

37. else

38. $distance(n_i, n_j) \leftarrow \infty$;

39. end

40. Call Dijkstar Algorithm to optimize $path$ by $distance[]$;

41. for n_i、n_{i+1} and n_{i+2} in $path$ do

42. Call Second-order Bezier curve to optimize n_i、n_{i+1} and n_{i+2};

43. end

44. print $path$;

45. End;

改进算法的具体算法流程图如图10所示。

3　仿真实验

为了验证本文提出的改进A*算法的航路搜索效率和航路优化效果,结合电子海图进行仿真实验。在硬件平台为i5-7500,主频3.4 GHz,运行内存8 GB的台式机,软件平台为Matlab R2018b的实验平台上进行实验,并与传统A*算法进行比较。

首先,从电子海图中获取海洋环境信息,使用正方形栅格建立的环境模型如图11所示,白色区域为可航行区域,黑色区域为不可航行区域。建立以水平向右为正方向的 x 轴,竖直向下为正方向的 y 轴的笛卡尔坐标系,网格数为200×87,一个网格即为一个节点。

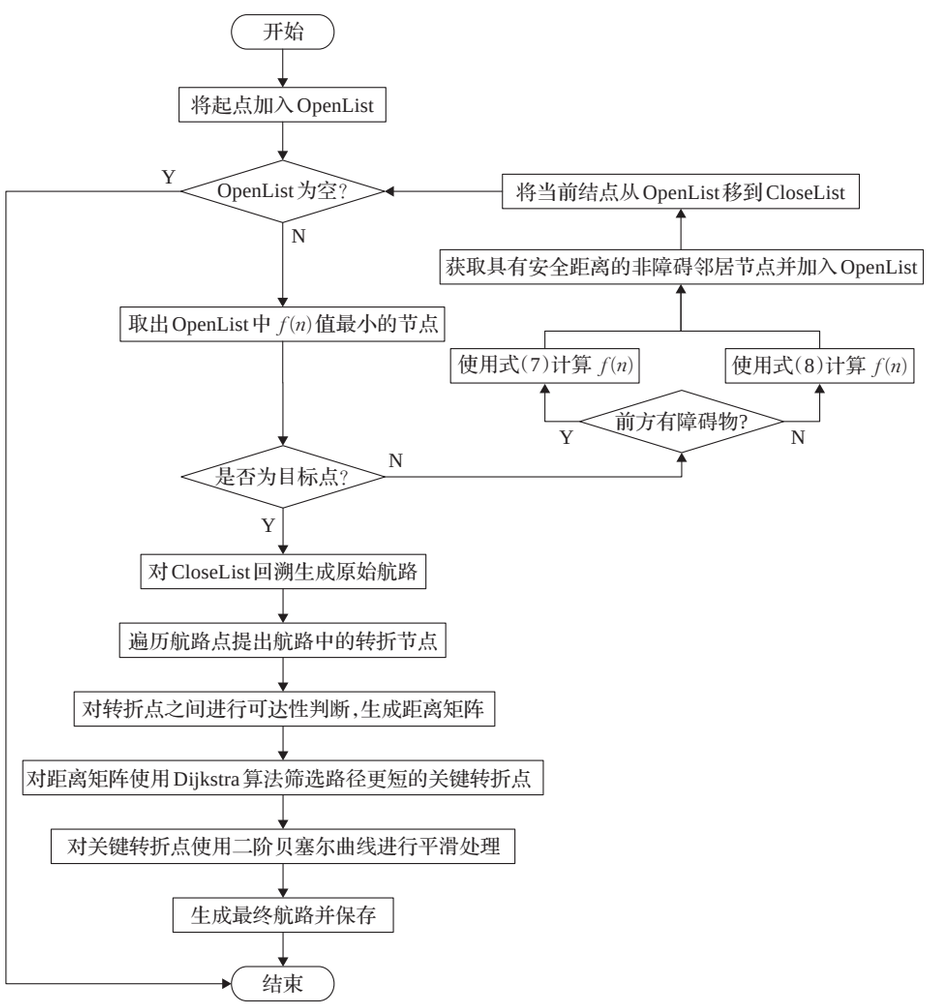


图10　改进A*算法流程图

Fig.10　Chart of improved A* algorithm process

表1　改进A*算法和传统A*算法航路规划结果

Table 1　Comparison of traditional and improved A* algorithm

地图	算法	地图规模	是否平滑	路径长度	OpenList规模	转折次数	安全距离	寻路时间/s
地图1	传统算法	200×87	否	237.45	6 746	24	0	1.452
	改进算法	200×87	是	214.00	4 579	6	2	1.046
地图2	传统算法	200×118	否	187.84	3 436	16	0	1.584
	改进算法	200×118	是	195.52	4 018	6	2	1.374

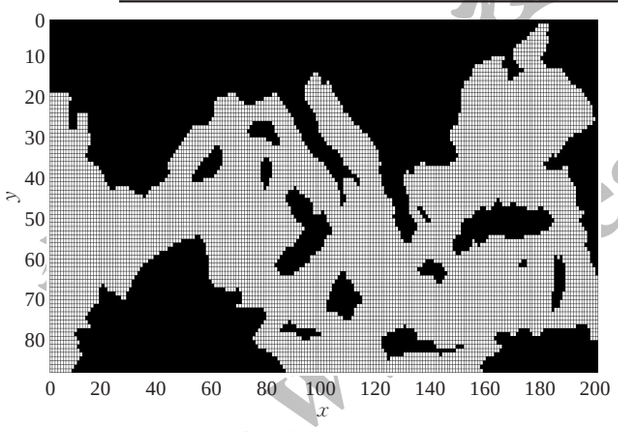
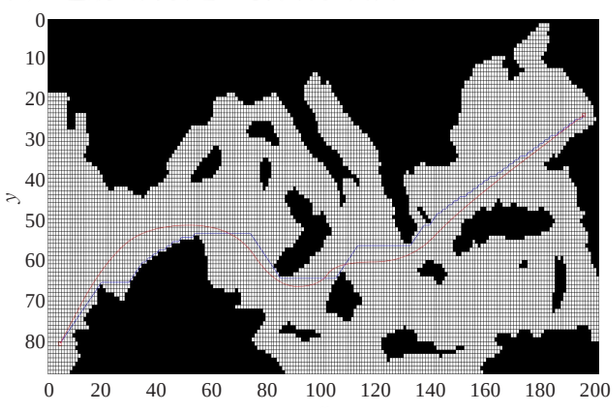


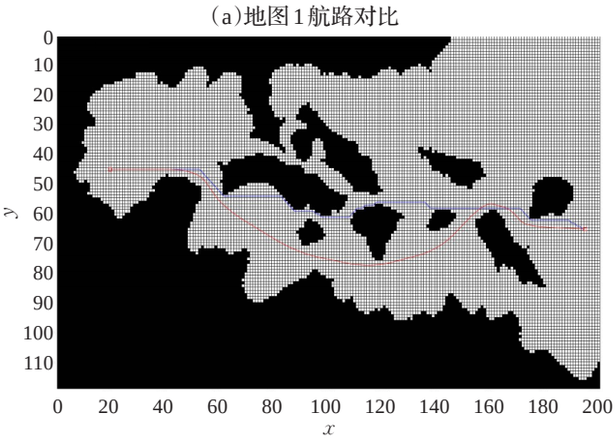
图11　电子海图栅格建模图

Fig.11　Grid modeling diagram of electronic chart

如图12所示,是在两组栅格化电子海图环境模型下的航路结果对比。蓝色航路为传统A*算法规划结果,红色航路为改进A*算法规划结果。



(a)地图1航路对比



(b)地图2航路对比

图12　算法航路对比图

Fig.12　Comparison of route of two algorithms

从图12(a)和表1的统计数据可以明显看出,改进算法的航路比传统算法的航路转折点更少,转弯处更加平滑,距离障碍物更远,使得舰艇更加容易进行跟踪。

从图12(b)和表1的统计数据可以看出,改进算法的航路比传统算法的航路长度略长,但避开了狭窄通道,安全性更高,转折点更少,航路更加平滑。

4　总结

针对复杂的海洋工作环境下,传统A*算法规划航路时出现的规划速度慢、沿障碍物边缘走安全性较差以及航路转折过多且不平滑等问题,本文结合栅格化后的电子海图环境模型,提出了一种加入目标方位信息的自适应启发函数改进A*算法,然后删除冗余节点后使用Dijkstra算法进一步优选航路的关键转折点,最后再使用贝塞尔曲线平滑法对航路转折处进一步优化。仿真实验结果表明,相较于传统A*算法,改进算法能准确地、迅速地在给定地图中内任意两个节点之间生成安全无碰撞且平滑全局航路,较传统A*算法优势明显。

参考文献:

[1] 冉东可,彭富伦,李红光.基于A*算法的路径规划研究综述[J].电子技术与软件工程,2020(24):11-12.
RAN D K,PENG F L,LI H G.A review of path planning research based on A* algorithm[J].Electronic Technology and Software Engineering,2020(24):11-12.

[2] 周克帅,范平清.改进A*算法与人工势场算法移动机器人路径规划[J].电子器件,2021,44(2):368-374.
ZHOU K S,FAN P Q.Improved A* algorithm and artificial potential field algorithm for mobile robot path planning[J].Chinese Journal of Electron Devices,2021,44(2):368-374.

[3] 韩学行,顿向明,林子洋.基于A*改进算法的机器人移动路径优化仿真[J].计算机仿真,2021,38(2):313-317.
HAN X X,DUN X M,LIN Z Y.Simulation of robot mobile path optimization based on A* improved algorithms[J].Computer Simulation,2021,38(2):313-317.

[4] 王子静,陈熙源.基于改进A*和DWA的无人艇路径规划算法[J].传感技术学报,2021,34(2):249-254.
WANG Z J,CHEN X Y.Path planning algorithm based

on improved A* and DWA for unmanned surface vehicle[J].Chinese Journal of Sensors and Actuators, 2021,34(2):249-254.

[5] 刘子豪,赵津,刘畅,等.基于改进A*算法室内移动机器人路径规划[J].计算机工程与应用,2021,57(2):186-190.
LIU Z H,ZHAO J,LIU C,et al.Path planning of indoor mobile robot based on improved A* algorithm[J].Computer Engineering and Applications,2021,57(2):186-190.

[6] 程杰,陈姚节.基于正六边形建模的无人水面艇路径规划[J].计算机技术与发展,2020,30(11):37-41.
CHENG J,CHEN Y J.Path planning of unmanned surface craft based on regular hexagon modeling[J].Computer Technology and Development,2020,30(11):37-41.

[7] 陈新,袁宇浩,饶丹.一种改进A*算法在无人船路径规划中的应用[J].计算机仿真,2021,38(3):277-281.
CHEN X,YUAN Y H,RAO D.Improved A* algorithm and its application in path planning of unmanned surface vehicle[J].Computer Simulation,2021,38(3):277-281.

[8] 祁玄玄,黄家骏,曹建安.基于改进A*算法的无人车路径规划[J].计算机应用,2020,40(7):2021-2027.
QI X X,HUANG J J,CAO J A.Path planning for unmanned vehicles based on improved A* algorithm[J].Computer Applications,2020,40(7):2021-2027.

[9] 刘生伟,马钺,孟树峰,等.改进A*算法的AGV路径规划[J].计算机应用,2019,39(S2):41-44.
LIU S W,MA Y,MENG S F,et al.AGV path planning with improved A* algorithm[J].Computer Applications, 2019,39(S2):41-44.

[10] 孔继利,张鹏坤,刘晓平.双向搜索机制的改进A*算法研究[J].计算机工程与应用,2021,57(8):231-237.
KONG J L,ZHANG P K,LIU X P.Research on improved A* algorithm of two-way search mechanism[J].Computer Engineering and Applications,2021,57(8):231-237.

[11] 吴鹏,桑成军,陆忠华,等.基于改进A*算法的移动机器人路径规划研究[J].计算机工程与应用,2019,55(21):227-233.
WU P,SANG C J,LU Z H,et al.Research on mobile robot path planning based on improved A* algorithm[J].Computer Engineering and Applications,2019,55(21):227-233.

[12] LIMA J,COSTA P,ECKERT L,et al.A* search algorithm optimization path planning in mobile robots scenarios[J].AIP Conference Proceedings,2019,2116(1):220005.

[13] SANG H Q,YOU Y,SUN X,et al.The hybrid path planning algorithm based on improved A* and artificial potential field for unmanned surface vehicle formations[J].Ocean Engineering,2021,223:108709.

[14] MIN H T.Autonomous driving path planning algorithm based on improved A* algorithm in unstructured environment[J].Proceedings of the Institution of Mechanical Engineers,2021,235(2/3):513-526.

[15] XUAN S L.Unmanned vessels path planning based on A* algorithm[J].International Core Journal of Engineering, 2021,7(2).

[16] XIONG X Y,MIN H,YU Y,et al.Application improvement of A* algorithm in intelligent vehicle trajectory planning[J].Mathematical Biosciences and Engineering, 2020,18(1):21.