An Improved A* Algorithm Based on Divide-and-Conquer Method for Golf Unmanned Cart Path Planning

Yi Chen(✉), Liangbo Xie, Wei He, Qing Jiang, and Junxing Xu

School of Communications and Information Engineering,
Chongqing University of Posts and Telecommunications, Chongqing, China
751796746@qq.com

Abstract. Path planning based on A* algorithm has been widely used in various engineering projects, but the time cost of A* algorithm for large-scale road networks is expensive, which is proportional to the square of node N. This paper proposes an improved A* algorithm based on the divide-and-conquer method for the golf unmanned cart path planning requirements, which splits the global optimal path into several local optimal paths and greatly reduces the time cost of the traditional A* algorithm with the more data space needs. Experiment results show that the proposed algorithm can decrease the time cost by at least 46% compared with the traditional A* algorithm. The real-time performance is stronger and the global optimizing path is smoother.

Keywords: Path planning · A* algorithm · Golf unmanned cart · Divide-and-conquer method

1 Introduction

With the fast advancement of science and technology, the human desire for the intelligentization of all things is becoming stronger. Unmanned robots are the direct products of such demand, such as driverless cars, unmanned factory and so on. For mobile robots, path planning is one of the most important requirements [1]. Path planning is planning an optimal path in time or distance in a certain environment model. When gives the source node and destination node information of the robot, it can guide the robot to complete the planning mission. Common path planning algorithms include A* algorithm [2], Dijkstra algorithm [3], Lazy Theta* algorithm [4], RRT algorithm [5] and some intelligent optimization algorithms such as genetic algorithm [6] and ant colony algorithm [7]. Among them, A* algorithm is an optimal path planning algorithm based on map search, which uses an inspiring idea to search along the direction of the target, avoiding blind, time-consuming and useless search effectively. This heuristic search algorithm is now widely used in path planning [8].

As one of the most classical path planning algorithms, A* algorithm has attracted great attractions of a large number of scholars and has achieved many research achievements. Chen et al. [9] use the direction vector as the heuristic search function of A* algorithm and introduce the parallel search method to search both the source node

S. Han et al. (Eds.): AICON 2019, LNICST 286, pp. 497–505, 2019.
https://doi.org/10.1007/978-3-030-22968-9_46

and the destination node simultaneously, which greatly improves the efficiency of the algorithm. In [10], the method of A* algorithm with time cost is proposed to achieve efficient and collision-free path planning of multiple AGVs for the parking scenario. In [11], a smooth A* algorithm is proposed. When the front and back nodes of the intermediate node meet the obstacle-free condition, the intermediate node on the extension line can be deleted, which can get better path planning at the expense of part of the time efficiency. In [12], the optimal path is obtained by simplifying the A* algorithm in the indoor positioning environment, and the rotation direction and the minimum rotation angle at the inflection node are obtained from the vector theory. Cui et al. [13] extend the A* algorithm from the traditional 8 domain to the 24 domain, increasing the search directionality, and selecting the optimal fscore and the sub-optimal fscore when expanding the node, thereby increasing the search flexibility. The number of turning nodes of planning path is reduced and the planning path becomes smoother. Huang et al. [1] combine distance transform and line-of-sight algorithms to make the planning path smoother and farther away from obstacles.

The above algorithms optimize the A* algorithm from a certain aspects. In order to solve certain environments, the time efficiency of some algorithms has to be sacrificed. For complex scenarios or large-scale road networks, the real-time performance is insufficient and the above algorithms are only theoretically simulated, which lacks certain engineering practice. Therefore, this paper proposes an improved A* algorithm with divide-and-conquer method that aims to improve the time cost of optimal path for the golf unmanned cart. First, establish a precise two-dimensional grid map of the golf course environment. Second, based on the A* algorithm, the divide-and-conquer method which splits the global optimal path into several local optimal paths is used to optimize the algorithm which reduces the time cost of the traditional A* algorithm and improves the real-time performance.

The rest of the paper is organized as follows. In Sect. 2, a precise two-dimension grid map model of golf environment is introduced. In Sect. 3, we describe the detailed theories of improved A* algorithm. The experiment results show the algorithm efficiency in Sect. 3. And Sect. 4 concludes this paper.

2 Environment Map Model

The environment map extracts the position, state and other information of objects in the real environment into a map model, which is the first step of all map-based search algorithms. The common map model representation methods mainly include grid maps and topological maps. The former uses the way of the coordinate system to be occupied by obstacles or not to describe the environmental features. The method is simple and the environment description ability is strong; the second is the use of topology nodes to establish map information which is mainly for environmental maps with large-scale road networks [14]. In this paper, based on the characteristics of the golf environment, a grid map is used to represent the golf environment characteristics.

A grid map breaks down the real environment into a series of discrete grid nodes and all grid nodes are uniformly and evenly distributed which are described according to whether they are occupied by obstacles. If the grid cell is occupied by an obstacle, it is a barrier grid, and vice versa is a free grid.

The construction process of the golf environmental model is mainly as follows. Firstly, the anchor nodes are identified on the four corners of the golf course, the real position information of the golf course is collected by RTK equipment, and then the aerial image technology is used to obtain an environmental picture with high resolution; The node can use the interpolation algorithm to obtain the real position information of each node in the environment, and then match the high-resolution environment picture one by one, and finally establish a two-dimensional grid map, the map not only has environment specific information, and each grid has real location information, which greatly facilitates the path planning algorithm and the automatic control of golf cart. Figure 1 shows an aerial view of a 75×120 m^2 golf course. Figure 2 shows a two-dimensional grid map with an interval of 1 m, white is a feasible area and red is an obstacle.

Fig. 1. Golf aerial image (Color figure online)

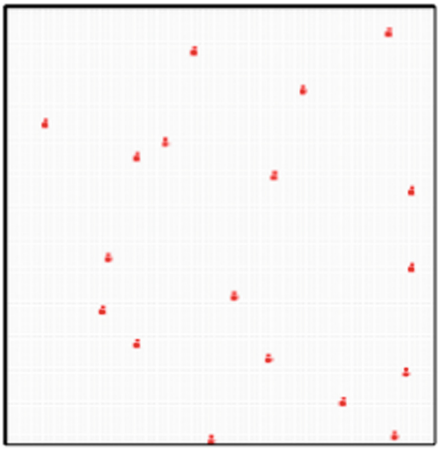

Fig. 2. Grid map (Color figure online)

3 Algorithm Description

3.1 A* Algorithm

The A* algorithm is a heuristic search algorithm which is similar to breadth-first search (BFS) and depth-first search (DFS) algorithm. They are based on certain principles to determine how to expand the node tree structure that needs to be searched [15], A* algorithm can be considered as a search algorithm based on “advantage nodes”.

The core part is to design an estimation function for each road node, as shown in Eq. (1):

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

where $\mathrm{f}(s)$ represents the estimated length from the starting node through the node s to the goal node; $\mathrm{g}(s)$ represents the known path length from the starting node to the current node, which is calculated by Eq. (2).

$$\sum_{\mathrm{i}=start}^{k-1} \cos t(s_i, s_{i+1})(k \leq goal) \tag{2}$$

In the Eq. (1), the heuristic function $\mathrm{h}(s)$ is the estimated value of the current node to the goal node, and the precondition that A* algorithm can get the optimal path must be able to satisfied Eq. (3).

$$\mathrm{h}(s) \leq \cos t^*(s, s_{goal}) \tag{3}$$

where $\cos t^*(s, s_{goal})$ is the optimal distance from current node to goal node. The larger the value $\mathrm{h}(s)$ satisfying the Eq. (3), the fewer the extension nodes. In order to get the optimizing path planning, the heuristic function is usually chosen as the Manhattan distance, the diagonal distance or the Euclidean distance [16]. The process of A* algorithm is as followed.

- Step 1: given the source node and the goal node, and establish a node set OPENSET for expanding search and a node set CLOSEDSET for completed search node;
- Step 2: select node cur with the smallest cost from OPENSET, that is $\min\{\mathrm{f}(cur)\}$ and perform 8 domain to expand the search. If the extended node u is not in the OPENSET and not in the CLOSEDSET, then add it to the OPENSET.
- Step 3: using the current node cur for cost update, the update criterion is when the cost of satisfying the source node src to the extended node u is greater than the sum of the cost of the source node src to the current node cur and the cost of the current node cur to the extended node u;
- Step 4: after the current node cur is expanded, put it into CLOSEDSET, indicating that the node has found the shortest distance and does not need to join OPENSET for search and update again;
- Step 5: loop through step 2, step 3 and step 4 until the goal node is found.

3.2 Improved Algorithm

The time cost of traditional A* algorithm is $O(N^2)$ [17], which means its computational complexity is proportional to the square of the grid number in the two-dimensional grid map, so its cost is very large for some scenes having many grids. For example, in a 2D game, the game map is at least 88×88. Using A* algorithm to find the optimal path, it is necessary to traverse all the nodes in the worst case, and the time cost is $\mathrm{n} = 88 \times 88 = 7744$. Therefore, this paper uses the divide-and-conquer method to

improve the A* algorithm, which can increase the efficiency of A* algorithm and reduce its time complexity.

Classic physics theories prove that the straight line between point A and point B is the shortest. Therefore, the optimal path searched by A* algorithm is a straight line when there is no obstacle. Their middle point C obtained by using the line drawing algorithm must be in the optimal path. If the optimal path from A to B is divided into two segments AC and BC, respectively, we can get their local optimal path planning with traditional A* algorithm. Finally, according to the dynamic programming idea, the global optimal path planning is achieved by put their local optimal path together with more space cost. For the previous example, the algorithm complexity at this time can be calculated as $n = (44 * 44) + (44 * 44) = 3872$, so the time cost decrease 50% at least.

Figure 3 is the logic block diagram of the improved A* algorithm. First given the segment number N, source node SRC, destination node DST and grid map MAP, and then we use the line drawing algorithm to get the $N - 1$ middle point. If there is an obstacle, take the neighboring point with the shortest heuristic distance as an alternative. Secondly, use the A* algorithm to find the local optimal path of each segment, and then connect them to obtain the global optimal path. The improved A* algorithm pseudo code is as follows.

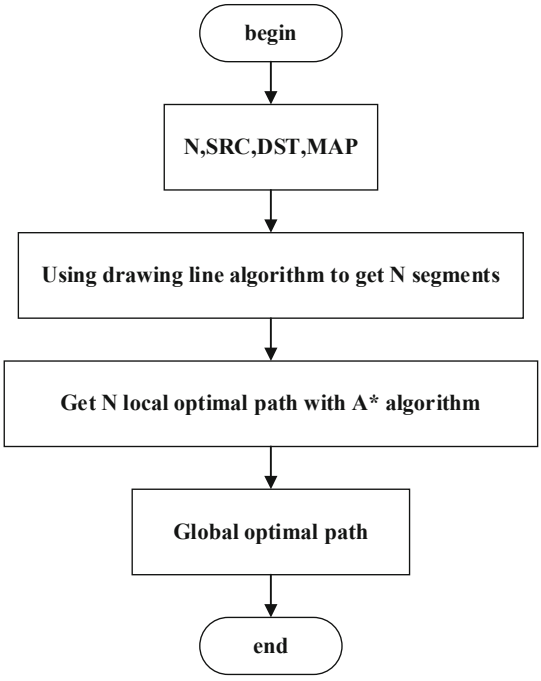

Fig. 3. Algorithm block diagram

```
function globalOptimizingPath = ImprovedAstar(MAP, SRC, DST, N)
create globalOptimizingPath
create POINT_SET
create stepX, stepY, index
stepX = (DST.x – SRC.x) / N
stepY = (DST.y – SRC.y) / N
index = 1
// find N – 1 middle points without obstacles
POINT_SET = getMiddlePoint(SRC, DST, stepX, stepY, N)
while all points from POINT_SET
  curSrc = POINT_SET(index)
  curDst = POINT_SET(index+1)
  localBestPath = AStar_Routing(MAP, curSrc, curDst)
  add localBestPath into globalOptimizingPath
end
end fucntion

function best_path = AStar_Routing(MAP, curSrc, curDst)
create vertex set CLOSEDSET  // set of already visited nodes
create vertex set OPENSET   // set of nodes to be expanded
create gScore, fScore with default value inf
insert curSrc into OPENSET
gScore[src] = 0, fScore[src] = h(src, dst)
create pre_path with default value nullptr
while OPENSET is not empty:
    current = the node v in OPENSET s.t. fScore[v] is minimum in OPENSET
    if current == curDst
      return reconstruction_best_path(pre_path, current)
    remove current from OPENSET
    insert current into CLOSEDSET
    for each neighbor u of current:
      if u is in closeSet:
        continue;          // ignore the neighbor who has already been evaluated
      candidate_score = gScore[current] + h(current, u)
      if u not in OPENSET:     // discovered a new node
        insert u into OPENSET
      else if candidate_score >= gScore[u]:
        continue;          // this is not a better path
      pre_path[u] = current
      gScore[u] = candidate_score
      fScore[u] = gScore[u] + h(u, dst)
end fuction
```

4 Experiment Results

In order to verify the efficiency of the improved algorithm, WINDOWS10 education version is used as the platform and the algorithm is simulated and verified by MATLAB software. The experimental hardware platform is Intel Core i5-3230M processor, 2.6 GHz and 8 GB memory. Under the same platform conditions, we have the comparison experiment between the proposed algorithm and the traditional A* algorithm. For the two-dimensional grid map of the golf course, different source nodes and destination nodes are given respectively to verify their algorithm time cost.

As shown in Fig. 4, three different source nodes and destination nodes are selected respectively. The optimal path obtained by the traditional A* algorithm and the proposed algorithm is compared, the green path and the black path are the result of traditional A* algorithm and the improved A* algorithm, respectively. We can find that the proposed algorithm is smoother and the path length is shorter than the traditional A* algorithm.

Table 1 is the cost time of the traditional A* algorithm and the proposed algorithm at different source and destination nodes when N is 2. The time cost of three group data using the traditional A* algorithm is 0.1583 s, 0.1246 s, 0.1736 s, respectively, and the time cost of the proposed algorithm is 0.0800 s, 0.0633 s, 0.0928 s, respectively. The increase efficiency is 49%, 49%, and 46%, respectively, so the proposed algorithm is at least 46% better in time efficiency than the traditional A* algorithm. In addition, it is known from Table 2 that under different N values (N = 2, 4, 6), we can find that the time cost of the three group data decreases as the value of N increases, as shown in Fig. 5.

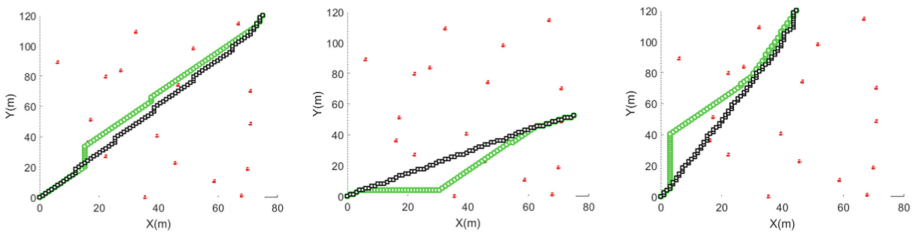

Fig. 4. Algorithm comparison (black is improved A*, green is traditional A*) (Color figure online)

Table 1. Algorithm comparison

Group	Source	Destination	A* cost (s)	Improved A* cost (s)	Efficiency
1	(1, 1)	(75, 90)	0.1583	0.0800	49%
2	(1, 1)	(75, 40)	0.1246	0.0633	49%
3	(1, 1)	(45, 90)	0.1736	0.0928	46%

Table 2. N efficiency comparison

Group	Source	Destination	N = 2	N = 4	N = 6
1	(1, 1)	(75, 90)	49%	55%	64%
2	(1, 1)	(75, 40)	49%	53%	62%
3	(1, 1)	(45, 90)	46%	62%	71%

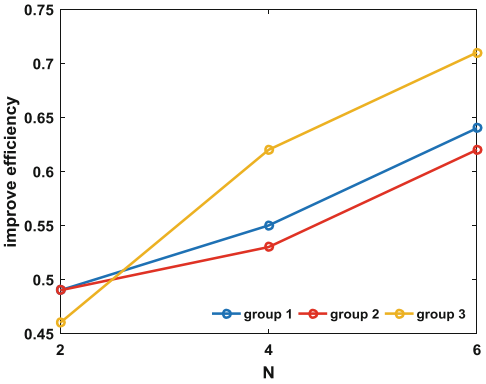

Fig. 5. Different N increase efficiency

5 Conclusions

This paper proposes a method to reduce the time cost of the traditional A* algorithm. Using the divide-and-conquer method, the optimal path between two nodes is divided into N segments, and then using the traditional A* algorithm to get N local optimal path planning. Finally, the N local optimal path is integrated to obtain the global optimal path. The experimental results show that the proposed algorithm improves the time efficiency by at least 46% compared with the traditional A* algorithm, and its efficiency increases with the increase of the N value and certain space sacrifice. The obtained optimal path is also smoother and shorter, which has a strong engineering practice.

References

1. Huang, L., Zhou, F.: Path planning of moving robot based on path optimization of D* lite algorithm. Control Decis. J. (2018)
2. Bundy, A., Wallen, L.: A* algorithm. In: Bundy, A., Wallen, L. (eds.) Catalogue of Artificial Intelligence Tools. SYMBOLIC, p. 1. Springer, Heidelberg (1984). https://doi.org/10.1007/978-3-642-96868-6_2
3. Johnson, D.B.: A note on Dijkstra's shortest path algorithm. J. ACM **20**(3), 385–388 (1973)

4. Firmansyah, E.R., Masruroh, S.U., Fahrianto, F.: Comparative analysis of A* and basic Theta* algorithm in android-based pathfinding games. In: International Conference on Information & Communication Technology for the Muslim World. IEEE (2017)
5. Song, J.Z., Dai, B., Shan, E.Z., et al.: An improved RRT path planning algorithm. Acta Electron. Sin. (2010)
6. Tsai, C.C., Huang, H.C., Chan, C.K.: Parallel elite genetic algorithm and its application to global path planning for autonomous robot navigation. IEEE Trans. Ind. Electron. **58**(10), 4813–4821 (2011)
7. Fan, X., Xiong, L., Sheng, Y., et al.: Optimal path planning for mobile robots based on intensified ant colony optimization algorithm. In: IEEE International Conference on Robotics (2003)
8. Chen, H., Xiong, G.: Theory and Design of Unmanned Ground Vehicle. Beijing Institute of Technology Press, Beijing (2018)
9. Chen, H., Li, Y.: Research of mobile robot path planning based on improved A* algorithm optimization. Autom. Instrum. 12 (2018)
10. Zhang, Y., Chen, Y., Wei, L.: AGV intelligent parking algorithm based on improved A* algorithm. Comput. Syst. Appl. **28**(1) (2019)
11. Wang, H., Ma, Y.: Path planning for mobile robots based on smoothing A* algorithm. J. Tongji Univ. (Nat. Sci. Ed.) **38**(11), 1647–1650 (2010)
12. Wang, D.: Indoor mobile robot path planning based on improved A* algorithm. J. Tsinghua Univ. (Nat. Sci. Ed.) **8**, 1085–1089 (2012)
13. Cui, B., Wang, M., Duan, Y.: Algorithm a path planning based on searchable 24 neighborhoods. J. Shenyang Univ. Technol. **40**(2), 180–184 (2018)
14. Liu, S.: First Book on Driverless Technology. Electronic Industry Publisher (2017)
15. Hart, P.E., Nilsson, N.J.: A formal basis for the heuristic determination of minimum cost paths in graphs. IEEE Trans. Syst. Sci. Cybern. (SSC) **4**(2), 100–107 (1968)
16. Bander, J.L., White, C.C.: A heuristic search algorithm for path determination with learning. Int. J. Imaging Syst. Technol. **16**(5), 154–161 (1998)
17. Feng, Q., Gao, J., Deng, X.: Path planner for UAVs navigation based on A* algorithm incorporating intersection. In: 2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC). IEEE (2016)