2010 Seventh International Conference on Information Technology

Path Planning for Virtual Human Motion Using Improved A* Algorithm

Junfeng Yao, Chao Lin , Xiaobiao Xie
School of Software,
Xiamen University,
Xiamen, Fujian Province, China, 361005

Andy JuAn Wang, Chih-Cheng Hung,
School of Computing and Software Engineering,
Southern Polytechnic State University,
Marietta, GA, 30060, U.S.A

Abstract—Calculating and generating optimal motion path automatically is one of the key issues in virtual human motion path planning. To solve the point, the improved A* algorithm has been analyzed and realized in this paper, we modified the traditional A* algorithm by weighted processing of evaluation function, which made the searching steps reduced from 200 to 80 and searching time reduced from 4.359s to 2.823s in the feasible path planning. The artificial searching marker, which can escape from the barrier trap effectively and quickly, is also introduced to avoid searching the invalid region repeatedly, making the algorithm more effective and accurate in finding the feasible path in unknown environments. We solve the issue of virtual human's obstacle avoidance and navigation through optimizing the feasible path to get the shortest path.

Keywords—Improved A* Algorithm;Evaluation Function; Path Optimization;Manual Searching Marker.

I. Introduction

In the scene map in which there are obstacles, it is a basic issue to plan an optimal path from the starting point to the target point to commit a mission in the virtual reality systems, robotics, geographic information systems, game development and other areas. Researchers have developed a variety of sophisticated virtual human motion models using kinematic and dynamic technology for years; however these models primarily reflected the virtual humans' motion in the current field. Not much studies in long-distance navigation of virtual human have been done [1]. With the rapid development of computer simulation and virtual reality technology, the demand for calculating automatically and generating optimal motion path of virtual human is becoming more and more pressing.

Search strategies are divided into non-information-search (also known as blind search) ones and information search (heuristic) ones. While the shortest path search strategy is to find the most direct and shortest path from the starting point to the target point according to the terrain and obstacles in the map. The A* algorithm, a combination of heuristic methods such as BFS and methods like Dijkstra algorithm [2], have been put forward in 1968. However, the traditional A* algorithm has characteristic of slow search speed and can easily fall into the failed search state when trap obstacles are met in the unknown environments [3]. In this paper, the improved A* algorithm for target movement navigation is put forward focusing on the virtual human in a provided virtual environment, the evaluation function is weighted, the manual searching marker is also introduced and the search speed is enhanced and the dead search state is avoided.

II. The A* algorithm

In the field of heuristic searching algorithm, A* algorithm which is widely applied is a graph searching algorithm applying evaluation function to sort the nodes[4]. This algorithm uses mainly an evaluation function: $f(n) = g(n) + h(n)$, to provide guidance and selection to the expanded nodes of the list *OPEN*, where $g(n)$ is the actual cost from the initial node to the node n (i.e. the cost finding the optimal path), $h(n)$ is the estimate of the cost of optimal path from node n to the target node, which depends on the heuristic information of the problem areas. If there is $h(n) \leqslant h^*(n)$ (being inclined to a conservative estimate) for all the nodes n in the path searching problem, then the best path to reach the target node will be found in the A algorithm[5]. The algorithm is known as the A* algorithm if the lower bound $h(n)$ of $h^*(n)$ is looked as a heuristic function in the A algorithm.

Applying the A * algorithm in path searching problems is mainly because of the understanding of the problem, solving process as well as the obtained solution, to search for some information related to and conducive to problem-solving, which is regarded as the heuristic information of the solution. Different from many other path searching algorithms, it does not traverse the entire map and all the nodes, the searching process of the A* algorithm is as follows.

1) To mark the initial node and expand the unmarked subsequent nodes or the child nodes.

978-0-7695-3984-3/10 $26.00 © 2010 IEEE
DOI 10.1109/ITNG.2010.53

2) To calculate the evaluation function value for each subsequent node, to sort them according to the value of the evaluation function and to identify and mark the node of the minimum evaluation function value. Searching can not be stopped until the current node is the target node.

3) It is necessary to employ the above processing steps for it and to record the shortest path if the current node is not the target node.

The most important step in the A* algorithm is to choose the evaluation function. It will guide the searching towards the most promising direction to seek the shortest and optimal path if the evaluation function is chosen properly.

III. Evaluation function

Now the evaluation function is set as *f*, to the value *f (n)* of *f* can estimate the smallest path cost from the initial node to node *n* at any node and the minimum sum path cost from node *n* to a certain destination node, that is, *f (n)* is an estimated value of the minimum cost path through the node *n* with restraint. Therefore, the node with the smallest *f* value in list *OPEN* is the estimated node of the least stringent constraint. To expand this node will be reasonable in the next step.

The function *g*(n)* is defined as the cost of one optimal path from the known start node to any node *n*, the function *h*(n)* is defined as the cost of the minimum cost path from node *n* to the destination node within the entire set of all the target nodes. A random path of *h*(n)* can be obtained from node *n* to the destination node, that is, the best path from node *n* to a destination node. The path length of the node *n*, which totally can not reach the target node, is defined as infinite.

Next step, the function *f*(n)* for any node *n* is defined. Its function value is the actual cost of the optimal path from the initial node to node *n* plus the cost of the optimal path from node *n* to the destination node, the formula is as follows:

$$f^*(n) = g^*(n) + h^*(n)$$

To the above formula, the function *f *(n)* is the cost of an optimal path, starting from the initial node and passing the node *n* with constraint. When the initial node and node *n* are combined into one node, the above formula expression becomes:

$$f^*(n)=h^*(n)$$

The formula represents the cost of one optimal path without constraint from the initial node to a destination node. Therefore, the above-mentioned evaluation function *f (n)=g(n)+h(n)* is an estimate of *f*(n)*, that is, where *g(n)* is an evaluation function of *g*(n)* and satisfies $g(n) \geqslant g^*(n)$. Similarly, the *h(n)* is an estimate of *h*(n)* and an evaluation function of *h*(n)* as well.

The definition of the evaluation function is the key point to the A* algorithm. Starting from the idea to find a minimum cost path, we divide the evaluation function into two parts, one is from the initial node to node *n*, the other is from the node *n* to the target node and calculation and analysis to their cost are carried out [6].

IV. The improvement and implementation of the A* algorithm

A. Implementation Of A* algorithm

In the implementation process of the algorithm, we need a structure *G*, in which the current generated search graph is displayed. The table *OPEN* is used to store the generated and to be extended nodes. The table *CLOSED* is applied to store the generated and processed nodes.

Concrete simulation steps are as follows:

1) To establish a search graph G formed by the initial node s. To push the initial node s into table OPEN, OPEN =(s). CLOSED=(), at this time, table CLOSED is an empty table, so f(s)=0+h(s) can be expressed.

2) To repeat the following process until the destination node is searched out. If the table OPEN is empty, that is, OPEN=(), then quit, that is, failed to find out the destination.

3) To select and mark the node of the smallest f value without previous setting from the table OPEN, remove it from the table OPEN and put it into the table CLOSED.

4) If the best node is the destination node, that is, the target destination has been reached, the search has successfully obtained a solution and the algorithm is over. The best path from the initial node to node n is gotten by tracking the pointer of the main chain.

5) The node n will be expanded if the best node n is not the destination node. The node set M=(m) is generated which consisted of all the successor nodes of node n that are not its ancestors. Node m is added as a successor to the node n into the search graph G.

6) Node m is added to the table OPEN if m appeared neither in the list OPEN nor in the table CLOSED.

7) Node k is removed from the table OPEN and the node m is added to the table OPEN if m has a duplicate node k in the table OPEN and g (m) <g (k).

8) To see whether the successor node is in the table CLOSED. If it is not in table OPEN, the operations as follows are carried out if the successor node has duplicate node in the table CLOSED.

a) To change node k into node m in the table CLOSED;

b) To modify the successor g, f value in the table OPEN and CLOSED which contains node k in the sequence of the successor elements.

9) To resort the nodes from small to large in the OPEN table according to the value of f.

10) To jump to the step(2) and to cycle.

In fact, a valuation function was just introduced into the A* algorithm on the basis of width preferred searching method. Not all the scalable nodes are expanded each time, however the valuation function is applied to evaluate all the non-expanded nodes to identify the nodes which should be expanded mostly and they are expanded until the destination node is found. In this paper, the A* algorithm is applied to search the optimal path and the result is shown in Figure 1. It is showed obviously that the optimal path has not be gotten as the traditional A* algorithm has two defects: slow Searching speed and falling easily into the failure of "dead" state in the obstructions trap of an unknown environment. Improvements in two aspects were made in this paper to solve the two points:

11) To weighted process the valuation function to ensure its reliability.

12) To introduce the manual searching mark to make the virtual man pre-judge the obstacles in front, so as to avoid falling into the dead state. The improved A Algorithm is shown in Figure 2.*

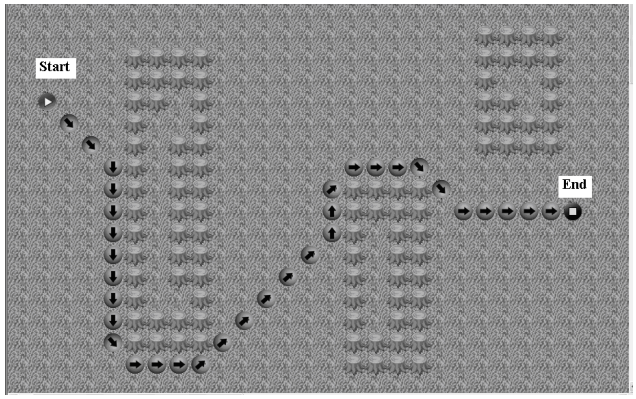


Figure 1. The searing results of the optimal path applying A* algorithm

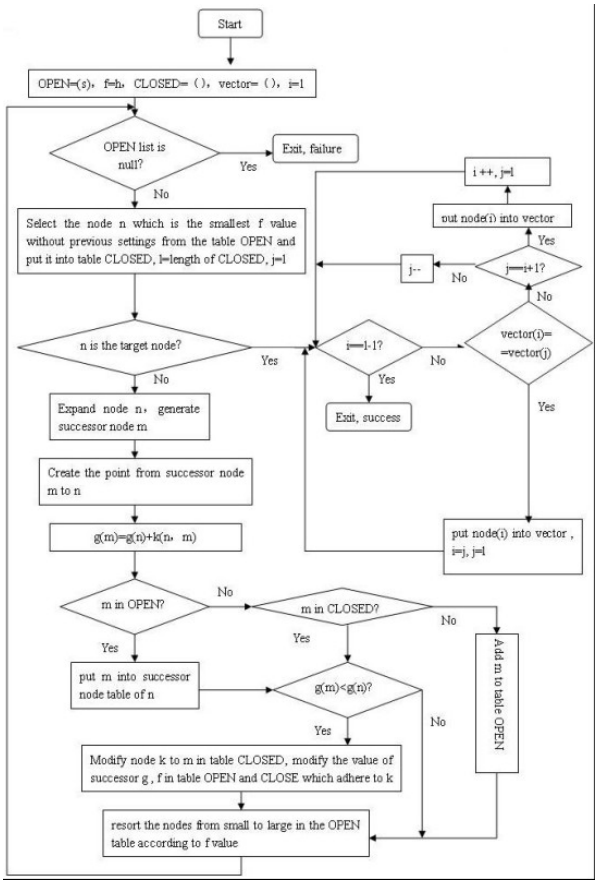


Figure 2. The searching process of the improved A* algorithm

B. Weighted processing

Generally, the estimated cost function *h (n)* is set as the Euclidean distance between the two nodes, which is often less than the actual shortest distance. It affects the searching efficiency of the algorithm although the admissibility of the algorithm can be guaranteed.

Therefore weighted processing is done to the valuation function according to the following acceptable proof lemma[7].

Lemma: there is always a node *n** in each step of the A* before termination, which has the following characteristics on table *OPEN*:

1) n is on the best path of achieving the objectives.*

2) A has found the best paths to reach n*.*

3) f (n) ≦f(n0).*

Then the algorithm is acceptable. Being further extended, any function, which distance is not higher than *h*, is acceptable, so these improvements are still acceptable.

Using a weighted cost function, as is shown below based on the above lemma:

$f_w(n) = (1-w)g(n) + wh(n)$

In the experiment, $w \in (0.5,0.9)$, the improved cost function does not affect the admissibility of the algorithm, which ensures to find solutions on the condition that some feasible solutions existed. Therefore the improved algorithm can be accepted and make searching efficiency being enhanced.

C. Introduction of manual searching marker

The traditional A* algorithm may be trapped in a narrow-based and arc-type trap barrier, so the trap should be prejudged and be escaped from. Unbelievable points should be defined firstly: in the Grid-based environment, the point set $P(A)= ((x, y)|x= Ax+ i, y=Ay+j, (x,y) \neq A, i, j \in (-1,0,1))$ should be constructed, which is adjacent to point $A (x, y)$. The obtained point $B(x, y)$ satisfies $h (B)<h (A)$ and $B (x, y)$ belongs to the obstruction set when $i, j \in \{0,1\}$. Thus $A (x, y)$ is called as an unbelievable point.

In order to avoid the virtual human being trapped in a narrow-type and arc-type trap, at first the motion of virtual human beings is allowed in the narrow and deep direction. When the virtual human reached an unbelievable point, it is defaulted as potential obstacle in advance and added to the collection of manual search markers. Virtual human beings should move along the sequence of the recorded parent nodes when the virtual human traced back. All the traced points are defined as unbelievable points, added to the Collection of manual searching markers. All its child nodes are removed from the table *OPEN*. The collection is merged into the barriers and will no longer be searched.

The searching efficiency of the algorithm can be improved by introducing inner searching markers and the algorithm will not fall into the trap. However the algorithm is still powerless when meeting the edge obstacles. By artificially increasing the outermost barrier, the obtained condition will met the requirements of artificially increasing obstacles in the environment, thus the edge obstacle situations can be classified into four categories: above, underneath, left and right[8,9].

D. Optimization of feasible path

It is necessary to determine the surrounding direction when an obstacle is encountered in the searching process of A* algorithm. The node $n+1$ will be randomly selected when its expanded nodes of the current node n are of the same minimum evaluation value if the evaluation function is not properly selected. But perhaps the next expanded node $n+2$ is not on the optimal path, the obtained path is not the optimal path. The path should be optimized furthermore. The artificial searching markers are continually added into the environment obstacles and a better path is searched repeatedly on the precondition of not affecting the searching efficiency and the closed nature of obstacles. The feasible path is compared with the previous feasible path after each search is completed. If it is completely identical, searching can be stopped, otherwise it should continue.

The algorithm to extract a feasible path from table *CLOSED* is described as follows steps:

1) To set end node label i=1, the length n of table CLOSED is solved, to make the node label j=n, to make vector empty;

2) If i=n-1, to go to Step 6). Otherwise, to compare the node i with the node j. If i and j are different, to go to Step 3). If i and j are the same, to go to Step 4);

3) If j=i+1, then to extract the node i, merge into vector and turn to Step5), else j--, transfer to Step2);

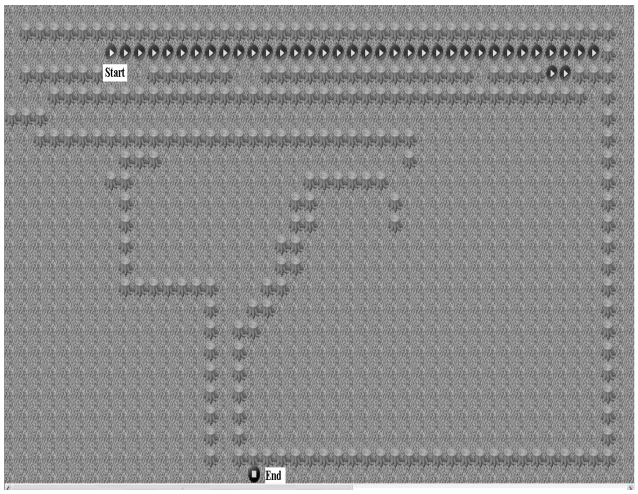
4) To extract the nodes i and merge into vector, i= j, j= n, to turn to Step2);

5) i++, j = n, to turn to Step2);

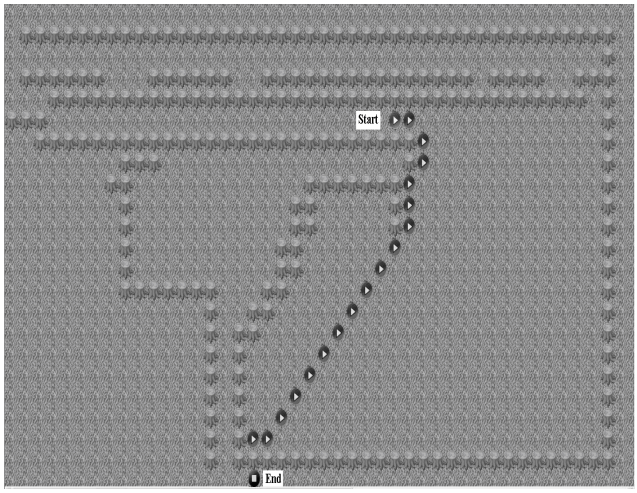
6) When node extraction is completed, the node sequence stored in the vector shall be the feasible path.

E. Experiment results and analysis

The searching efficiency is poor and searching is often caught in traps using traditional A * algorithm to solve the strip-type and arc-type trap cases. When a complex environment combined with many obstacles is met, it's more powerless. The improved A* algorithm can effectively compensate for the shortcomings of traditional A* algorithm, such as boosting the search speed and pre-judging or escaping from the trap. The experiment results are shown in Figure 3 and Figure 4 below:



(a) Caught in strip-type traps using traditional A * algorithm



(b) Caught in arc-type traps using traditional A * algorithm

Figure 3. Searching results with traditional A * algorithm

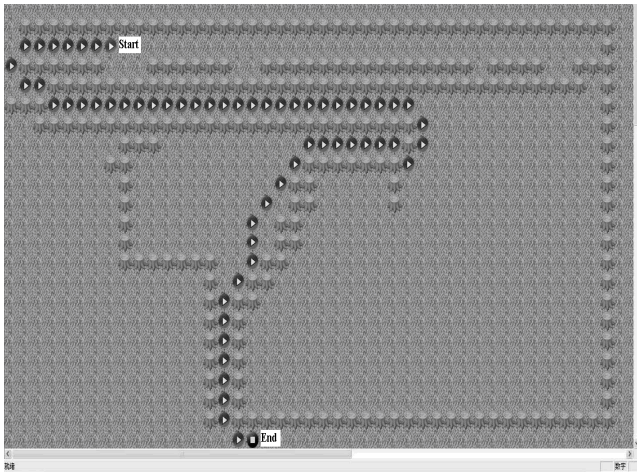


Figure 4. The searching results with improved A * algorithm

In Figure 3(a), the traditional A* algorithm can not escape from the trap when encountering the first narrow-type trap, the search frequency is 200 times. It can not escape from the arc-type trap either, the search frequency is 103. While in the same environment, the improved A* algorithm can escape from the trap and correctly move to the target node, as is shown in Figure 4. The search times is only 80. Obviously it has improved the search efficiency and the comparison of results is shown in Table 1:

TABLE I. SEARCH EFFICIENCY COMPARISON TABLE

Type	Search times	Search cost (S)	Process time (S)
Traditional A*	200	4.359	7.879
Improved A*	80	2.823	3.601

Search times: times finding the optimal path

Search cost: time spent in finding the optimal path

Process time: the total time spent in finding the optimal path and marking points in the optimal path

From table 1, it is showed by comparison figure 3(a) and figure 4, which is the same scene map, that the search cost is decreased form 4.359 s to 2.823 s, the process time is reduced form 7.879 s to 3.601s.

V. Conclusion

A* algorithm is a classic algorithms in artificial intelligence, which can be not only used in the shortest path searching in game maps, but commonly in the optimal state deduction in chess class game with complex state-space. In this paper it is applied in path planning of virtual human being motion, and has solved the shortest path searching problem in target navigation motion in a already known virtual environment.

At the same time, the traditional A* algorithm has been improved and achieved through the grid of obstacles, classification, simplification, weighted processing the evaluation function, adding manual search marker and optimized the found feasible path. Thereby the efficiency of A* algorithm is improved and the effectiveness of the A* algorithm to the unknown environment with obstacles is also enhanced.

References

[1] Bandi S, Thalmann D: "The use of space discretization for autonomous virtual humans" [C]//*Proceedings of the second International Conference on Autonomous Agents* (AGENTS - 98) , Minneapolis,1998: 336- 337.

[2] Srikanth Bandi, Daniel Thalmann : "Path finding for human motion in virtual environments" [Z]. *Electronic Imaging and Media Communications*, University of Bradford, Bradford, West Yorkshire, BD7 1DP .

[3] Anthony Stentz : "Optimal and Efficient Path Planning for Partially-Known Environments" [A]. The Robotics Institute; Cmegie Mellon University; Pittsburgh, PA 15213.

[4] SUN Shudong, LIN Mao, "The coordination path planning of multiple moving robots based on GA"[J] .*Automation Journal*, 2000,26(5) : 672-676.

[5] Mahmoud Tarokh : "Hybrid Intelligent Path Planning for Articulated Rovers in Rough Terrain" [Z]. *Fuzzy Sets and Systems*. 2008,159(21):2927-2937.

[6] YE Tao, CHEN Haikui, YANG Guosheng, "a new method of Global robot navigation and obstacle avoidance in Unknown environment" [J],*robot*, 2003,25(6) : 516-520.

[7] GAO Qingji, YU Yongsheng, HU Dandan, "feasible path search and optimization Based on an improved A * algorithm", *China Civil Aviation College Journal*, 2005,23 (4): 42-44.

[8] WEI Ning, "Virtual Global Path Planning Technology [D]. Jiangsu University", Computer Science and Communication Engineering College, 2007.

[9] WANG Li, "The Study of Mobile Robot Path Planning" [M]. Northwestern University. 2007.