AN IMPROVED REAL 3D A* ALGORITHM FOR DIFFICULT PATH FINDING SITUATION

Lei Niu, Guobin Zhuo

State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing,
Wuhan University, Wuhan, China, 430072
- niuneilneo@vip.163.com

Commission Ⅳ, WG-IV-6

KEYWORD: GIS, Three-dimensional, A* Algorithms, Urban, Space

ABSTRACT:

Path finding requirement for people under restricted situation is difficult to meet. Traditional routing solution such as A* algorithm is not suitable for the restricted area. For routing requirement under three dimensions is much more complex than under two dimensions, the traditional A* algorithm should be improved to meet the routing requirement. The new path finding solution concentrates on the experiment area's composition and each movement's moving style. The test result has shown that the improved A star algorithm is superior to traditional A star algorithm in both storage consuming and time efficiency.

1. INTRODUCTION

Accompanying with the increasing activity of people among cities and countries in modern society, the requirement for navigating people's movement grows with unexpected speed. Researchers have paid more efforts on the developing of path finding technology(Jones, 2001). It is necessary to have a look on its history to understand path finding. From the nineteen seventies, some scientists started research on the routing solution for moving chess in the chessboard or moving fragment in the puzzle map(Eklund et al., 1996). The reason of research first starting on these subjects is that the problems involved in these situations are easily abstracted to set up a request for the path finding algorithm. And with the development of path finding, several new classical routing algorithms have been introduced to generate better routing solution. Dijkstra algorithm is the most famous one, which evaluates the moving cost from one node to any other node and sets the shortest moving cost as the connecting cost of two nodes(Eklund et al., 1996).

Nearly at the same period Best-First-Search algorithm is also introduced in the community. Quite different from Dijkstra algorithm, Best-First-Search algorithm estimates the distance from current position to target, and chooses the step more approaching target(Amit). With the path finding situation's difficulty growing, the classical path finding algorithms need to be improved to meet the new requirement. Thus a new path finding algorithm named A* algorithm is introduced. A* algorithm combines the advantages of Dijkstra algorithm and Best-First-Search algorithm, for the A* algorithm not only intends to take shortest step among each movement, but also cares about the choosing step whether on the direction which is just from source to target(Jones, 2001).

With the development of A* algorithm, improving A* algorithm's efficiency has become the key point of research. For A* algorithm is a breadth first algorithm, it consumes huge memory to keep the data of current proceeding nodes(Nelson and Toptsis, 1992). During the traversing of all grids which are possible to be placed on the optimized path, a huge size of stack is needed to contain the considering grids. Beside developing A* algorithm's own efficiency, new methods of using A* algorithm are also considered by the researchers. For example bidirectional A* algorithm searching method has been used to reduce the time cost of A* algorithm(Nelson and Toptsis, 1992). Compared to classical A* algorithm's searching from source to target, bidirectional A* algorithm searches nodes not only from source to target, but also from target to source. The searching stops immediately when the two direction's searching progresses meet each other in bidirectional A* algorithm(Nelson and Toptsis, 1992).

Accompanying the three dimensional trend in computer society, three dimensional A* algorithm's development has caught more attentions. To solve three dimensional path finding problem, some path finding solution maps three dimensional problem area to two dimensional expression in order to use traditional A* algorithm solving the path finding request(Makanae and Takaki, 2004). Although the method of mapping 3D to 2D is working for path finding requirement under some simple 3D situations, the mapping method could not easily be used to finish path finding under complex situations. For example in the restricted spatial situations such as underground and inner building, the overlapping layers may appear frequently, and these situations seems impossible to take traditional A* algorithm solution, for mapping 3D into 2D and deriving the optimum path are nearly impossible under these special circumstances Thus A* algorithm should be improved to meet these routing requirements.

The three dimensional A* algorithm is required to work the routing problem out under restricted situations. Several certain modifications should be taken for standard A* algorithm and a new improved 3D A* algorithm is introduced.

2. ALGORITHM

The improved 3D A* algorithm is based on a classical A* algorithm theory, which is a heuristic method to find optimum path from starting position to destination. A* algorithm's main idea is to treat the testing area as a grid collection and generate the optimized path. In the standard A* algorithm each movement along the optimized path is evaluated by the formula:

$$F = G + H \quad (1)$$

In the formula, F represents the total moving cost from source to target, while G represents the moving cost from the source to current position, and H refers to the estimated moving cost from the current grid to target(Jones, 2001). The open list keeps grids which are still in the evaluating process by the path finding algorithm, and closed list keeps grids which have already been evaluated by the path finding solution. A* algorithm searches the whole area by maintaining an open list and a closed list to find an optimized path(Amit).

In the new improved A* algorithm, the basic unit used is *cell*, and the spatial object which contains specific type of cell is called *region*. Cell is an object which contains a specific quantity of space. As the objects in three dimensions does not only have neighbours on the same height level, but also has neighbouring objects above and below, the consideration of cell's shape in three dimensions is critical. The reason is that the shape of cell will directly define the number of cell's neighbouring units. In this experiment, the cell takes cube shape. Thus every movement of the routing solution is represented by transportation between cube shaped cells. Some constraints are made for the routing algorithm, according to the situation in the real world. For example by the consideration of the gravity on earth, it is not intelligent to allow moving a large distance in vertical direction, so the algorithm rejects any vertical moving request which are not acceptable under normal situations.

Improved A* path finding starts from initializing the testing area with a format which computer could recognize. Firstly testing area is set up by cell which is suitable for the problem representation, and secondly cells of same type compose region. After setting up the regions in the experiment area, the path finding for the whole area can be divided into routing work in each region. So the third step of the new routing algorithm is to determine which region will be used to set up the best path, and a judging algorithm is used to find involved regions. The judging algorithm finds the regions which are more approaching the segment of optimum path between the starting position and end position at the fourth step. After finding candidate regions along the optimized path, a testing algorithm is used to find out whether chosen regions can truly set up a path leading people from source to target. If there are several compositions of regions along the optimized path, which composition is best among these candidate compositions will be chosen by a evaluation algorithm.

The routing solution described above has many key points. The moving style's change in 3D situation is the first key point. 3D A* algorithm is much different from 2D A* algorithm, and the reason is the increasing quantity of styles for every single movement. In traditional 2D routing progress, only horizontal moving condition is considered. Thus there are only four basic moving styles in 2D situation, which are moving forward, moving backward, moving left and moving right; and there are other four complex moving styles which are moving forward plus left, moving forward plus right, moving backward plus left and moving backward plus right. While the traversing styles between cells in 3D contains vertical moving besides horizontal moving. So the moving styles have been increased from eight to twenty six in 3D routing situation, which contains eight styles in horizontal moving and eighteen new moving styles brought up by the introducing of vertical moving. The eighteen new styles are vertical up, vertical down, vertical up plus eight types of horizontal moving introduced by 2D situation and vertical down plus the eight types horizontal moving.

The increasing number of moving styles in 3D situations does not only cause more choices in moving step, but also change the routing algorithm's time efficiency and memory cost. As is known to all, A* algorithm maintains an open list and a closed list. The closed list contains the units which are chosen or rejected to be placed on the result path, while the open list keeps the neighbours of the unit which are being considered[red]. With the extension of dimension, the number of units which are possibly involved in the consideration of each moving step will be increased by 3.125 times maximally. The increasing moving choice will directly affect the open list's size. Under the most pessimistic situation, the open list could grow with the power of 3.125. The same problem arises with the time spent for the routing process. For the options of next moving step are increasing, the algorithm will consume more CPU computing time to find the best choice of next movement. Beside the probably obvious storage and computing time's increasing of the path finding algorithm, there are other problems lying below the dimension increasing. The most serious problem is that the cache hit rate will become much lower in the 3D routing algorithm. The decreasing cache hit rate is caused by the dimming possibility of the next moving step whether along the optimum path. For the moving options have been 3.125 times larger under the worst situation, the chance of choosing the best moving option is decreased by 3.125 times based on the 2D situation. And the total cache hitting possibility would be very pessimistic according to the huge choice increasing.

The second key point is that fortunately the improved 3D routing algorithm will somehow overcome the shortness caused by increasing moving styles under 3D situations. For the testing area will be divided into several regions for path finding algorithm, the detail path finding work is limited in each region and region size is indeed very small compared to the total experiment, so the total path finding process will not suffer a lot on the storage cost,. And the improved 3D routing algorithm also brings other benefits by using region. For example for the path searching in each region is independent, the paralleling computing can also be introduced to the path finding solution,. Thus the time cost of total path finding solution could be minimized.

3. APPLICATIONS AND EXPERIMENTS

The experiment area used to testify the improved real 3D A star algorithm is part of Shibuya station. The station is not only used for underground transportation, but it also contains bus stop, supermarket and railway station. The total area is composed by three main layers which cover thousands square meters area. The experiment area used has two parts, which is part of first layer and second layer of Shibuya station. The nodes

represented in the Figure 1 are used to help people get general view of the experiment area. Compared to the traditional A* algorithm using grid description of the experiment area, the 3D object "cell" is used in improved 3D A star algorithm. The cell used here is a cuboid shape having 100cm width by 100cm's width with 250cm height. The reason that the height of cell is just the half of the layer is that one physical layer will contain two layers of cells, and the cell layer near ceiling has the function of blocking movement which is not reasonable under normal circumstances.

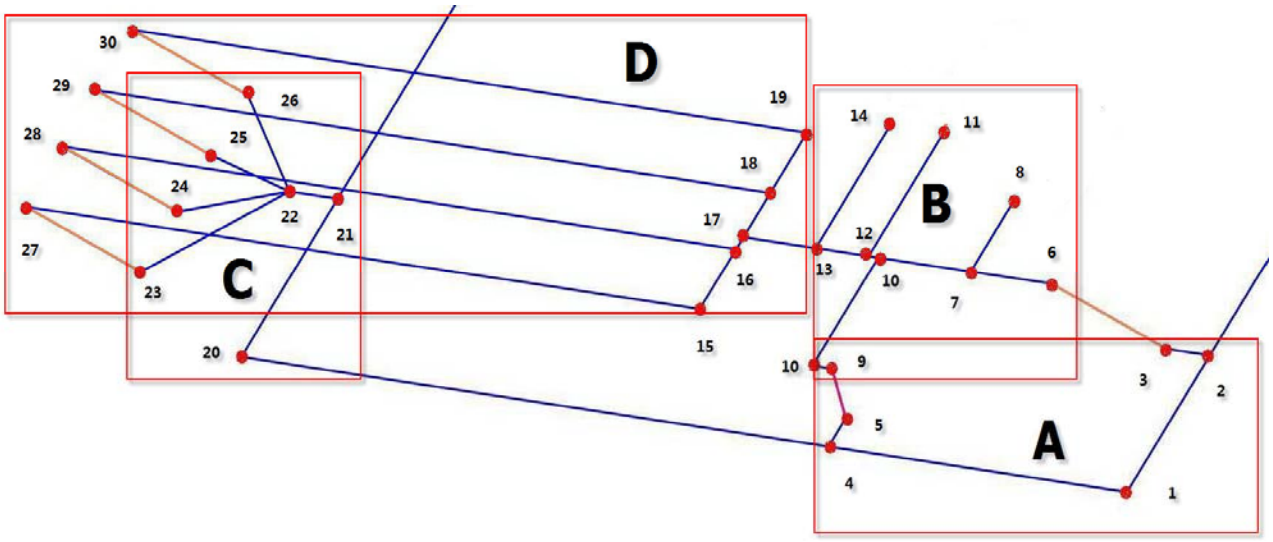

Figure 1. The region division of the testing area

Intending to get good testing result, total experiment area is divided into four regions. Region A contains the area which is selected by the rectangle "A" in Figure 1, and region A doesn't include the area around node 9 and node 10. Region B contains the area which is selected by the rectangle "B" in Figure 1, and region B includes the area around node 9 and node 10. Region C contains the area which is selected by the rectangle "C" in Figure 1. Region D contains the area which is selected by the rectangle named after "D" in Figure 1, and region D doesn't include the area contained in the "C" rectangle below. The improved A* algorithm first decides the regions which shall be involved in the optimized path solution, and then improved A* algorithm finds the joint nodes which shall also be used. In this test, starting position is set at node 2, and the ending point is set at node 3. Thus the region A, region B and region D should be taken to form up the optimum path, for that connecting line of node 2 and node 3 cross these three regions.

To generate a clear view of improved 3D A* algorithm's performance, we introduce the path finding result by traditional A* algorithm as the comparing object. Being different from the improved A* algorithm, the optimized path solution provided by traditional A* algorithm doesn't care about the region questions caused by the "region" idea such as travelling across region boundaries. Although the routing solution generation theory in traditional A* algorithm is simple, the storage space cost and CPU cost are massive. In the experiment, the cache of heap for storing searching node needs to be no less than the maximum number of nodes. So the detail travelling matrix of whole test area needs 30 × 117 × 3 elements by using traditional A* algorithm. While the improved A* algorithm only takes region A, region B and region D into consideration instead of the whole area. Region A travelling matrix contains 17 × 31 × 3 elements; region B travelling matrix contains 14 × 18 × 3 elements, and region D travelling matrix contains 24 × 85 × 3 elements. All these data is contained in Table 1.

The searching time cost is 0.314 milliseconds through the whole area by traditional A* algorithm. While searching path in region A costs 0.066 milliseconds, and searching path in region B and region D cost 0.032 milliseconds and 0.184 milliseconds by improved A* algorithm. The region switching time cost is nearly definite time, and it can be ignored in the routing solution. It is obvious that the sum of time cost in region A, region B and region D by improved A* algorithm is smaller than cost of searching in the total experiment area by traditional A* algorithm. And there is another good news:the paralleling computing ability of the improved A* algorithm makes it possible that the summary time cost only equal the largest time cost among the involved regions, which means the total time cost is equal to region D's searching time 0.184 milliseconds in the experiment.

Not only the computing time has been shortened for the introducing of improved A* algorithm, but also the storage cost is reduced. The static cost of storing elements in traditional A* algorithm takes 84240 bytes. The new improved A* algorithm static storage cost is 6324 bytes for region A, 3024 bytes for region B and 24480 bytes for region D. The total storage cost of these three regions is 33828 bytes. The dynamic cost is represented by the cache size of temporary storage heap in our test, and it means that in the A* algorithm at least three times storage space of static storage space need to be reserved for open list. And that means 126360 bytes for the experiment's total area dynamic using in this article. And under paralleling situation the improved algorithm only needs the dynamic storage space of the biggest region's dynamic storage cost, which means the improved algorithm only takes 73440 bytes space for temporary using. The space dividing of experiment area also enables the routing solution to face the emergency situations. For example, if there is an earthquake happens in the experiment area and disables travelling in region B, then the routing algorithm only changes the travelling cost in region B to

make region B unable to be passed and the improved routing algorithm will take region C to travel to the target instead.

There are still many ways to optimize the improved A* algorithm. For example region used in the improved A* algorithm can be taken place by *layer*. The layer is a concept that owns certain cells which have same vertical attribute and are in one certain horizontal scope. For the normal moving style is horizontal to layer, the path finding algorithm only need to consider moving in x and y axes' directions. And the real path finding process can be optimized to search only 8 neighbouring cells of current position for vertical moving appears nearly impossible in layer. And this more restricted moving condition could minimize the cost of CPU and memory used on many uninvolved nodes in path finding process.

4. CONCLUSION

After the experiment in the test area, advantages of the improved A* algorithm has been demonstrated:

1) As is known to all, A* algorithm's open list consumes huge size of cache. Although effort has been made to reduce the cache cost of A* algorithm in the past, the result is not very significant. The region using idea in the improved algorithm partly overcome the cache consuming shortage of A* algorithm. Because of introducing of "Region", smaller amount of storage is provided for each region's processing.
2) Paralleling computing of the algorithm calculation is introduced by using the improved A* algorithm. For each region is an independent path finding area, it makes program own the ability to use multi-thread to calculate each segment of the optimum path.
3) The "Cell" structure makes the improved A* algorithm works more flexibly than ever. Any change of the corresponding territory will be represented in the cell. Small modification in the experiment area will only affect the related cells in one region instead of reconstructing the total problem area.

REFERENCE

Amit, Amit's Game Programming Information. http://www-cs-students.stanford.edu/~amitp/gameprog.html. (accessed 28 Jan.2008)

Eklund, P.W., Kirkby, S. and Pollitt, S., 1996. A Dynamic Multi-source Dijkstra' Algorithm for Vehicle Routing. In: N.a. Jain (Editor), Conf. on Intelligent Information Systems, Australian New Zealand.

Jones, J.H., A* Tutorial. http://www.geocities.com/jheyesjones/astar.html. (accessed 28 Jan.2008)

Makanae, K. and Takaki, M., 2004. Development of the 3-Dimensional Urban Spatial Data Model and Application to the Pedestrian Navigation System. ITS シンポジウム, 3.

Nelson, P.C. and Toptsis, A.A., 1992. Unidirectional and Bidirectional Search Algorithms. Digital Object Identifier, 9(2): 77-83.

Area	Routing Solution	Matrix Elements Number	Static Storage Cost	Reserved Dynamic Storage Cost	Time Cost
Total area	Standard 3D A* algorithm	30 × 117 × 3	42120 bytes	126360 bytes	0.314 millis econd s
Region A	Improved 3D A* algorithm	17 × 31 × 3	6324 bytes	18972 bytes	0.066 millis econd s
Region B	Improved 3D A* algorithm	14× 18× 3	3024 bytes	9072 bytes	0.032 millis econd s
Region D	Improved 3D A* algorithm	24 × 85 × 3	24480 b1ytes	73440 bytes	0.184 millis econd s

Table 1. Testing result of the different A* algorithm