J. Marine. Sci. Appl. (2010) 9: 27-33
DOI: 10.1007/s11804-010-8011-6

Framed-Quadtree Path Planning for an Underwater Vehicle with the Task of Tracking a Moving Target

Bo Gao*, De-min Xu and Wei-sheng Yan

College of Marine Engineering, Northwestern Polytechnical University, Xi'an 710072, China

Abstract: An autonomous underwater vehicle (AUV) must use an algorithm to plan its path to distant, mobile offshore objects. Because of the uneven distribution of obstacles in the real world, the efficiency of the algorithm decreases if the global environment is represented by regular grids with all of them at the highest resolution. The framed quadtree data structure is able to more efficiently represent the environment. When planning the path, the dynamic object is expressed instead as several static objects which are used by the path planner to update the path. By taking account of the characteristics of the framed quadtree, objects can be projected on the frame nodes to increase the precision of the path. Analysis and simulations showed the proposed planner could increase efficiency while improving the ability of the AUV to follow an object.
Keywords: regular grid; framed quadtree; data structure; algorithm efficiency
Article ID: 1671-9433(2010)01-0027-07

1 Introduction

Planning a collision-free path is one of the fundamental requirements for the work of an underwater vehicle. There have been several approaches to satisfy this aim (Murphy, 2000; Latombe, 1991), which could guide the vehicles to avoid collision with obstacles and reach the goal with global information about the environment. This work can be referred as the work of navigation. Underwater environment pose special challenges to the navigation over the structured world that is often found on land. Not only must the vehicle avoid collision with the obstacles such as the rock, it must also take the efficiency of navigation into consideration for the environment has vast areas. Besides of that, vast areas have their own special issues about the path planning.

When the vehicle travels in such proportion of the space that typically have large areas like in the deep sea, there may probably be few obstacles and the vehicle could travel freely. On the other hand, when traveling offshore, the obstacles may be populated intensively, which interfere with vehicles' motion largely. Besides of that, the resolution of the map of the whole work space that the vehicle could get in advance is generally low and the vehicle should perfect the map when traveling in the real world. Therefore, as the existence of these special issues of the underwater environment, especially for the vast areas and unevenly distribution of obstacles, we should make improvement on the storage of the map of the environment. The challenges here mainly include: how to store the large amount of information of complex model about the environment, how could the vehicle determine the location of the vehicle on the map, in what way constructing the map of environment that requires low usage of memory and performs the work at high resolution when necessary.

Approaches to represent the environment can be classified into two categories: exact representation and approximated representations (Choset *et al*., 2005), which have different characteristics on path planning. The exact representation for the environment could get complete solution of path throughout the planning algorithm such as visibility graph (Rohert, 1986), Voronoi diagram (Choset and Burdick, 2000), and cell decomposition (Lozano, 1983). However, the cost of storage of the map and complex of the planning will increase phenomenally with the complexity of environment. The main advantage of the approximation representation is that the storage usage of map and complexity of algorithm could be controlled through the adjustment of resolution. Here we use the method of discretizing the world.

Regular grids are mostly used in metric path planning as the discretized representations of environment. This kind of representation has the advantage of high resolution of environment. However, because the information about environment is disregarded, there are much space wasted in storing the area where the vehicle seldom travels, which will eventually have bad influence on space and time efficiency. According to the situation, an efficient kind of space representation called quadtree was proposed (Samet, 1982) to make progress in saving the space. However, as the grids in quadtree are different in size, the path created on the foundation of quadtree can not be guaranteed to be the optimal path in real world. Considering that situation, a new kind of data structure called framed quadtree is proposed. It combines the merits of regular grid and quadtree (Chen *et al*.,

Received date: 2008-03-17.
Foundation item: Supported by the National Natural Science Foundation of China under Grant No. 60875071.
***Corresponding author Email:** carrot304@gmail.com

1995; Yahja *et al*., 1998). Here a data structure of multi-level tree is proposed for framed quadtree. Also, the efficiency of regular grid, quadtree and framed quadtree are compared. In this paper, we propose a method of decomposing the environment and increasing the resolution when needed to satisfy the requirement.

In practical implementation, the target chased by the vehicle is always moving dynamically. The path planner should adjust the planning path also keeping the low cost of path at the same time. The traditional metric path planner such as Dijkstra algorithm (Dijkstra, 1959), A* algorithm (Pearl, 1984) and D* algorithm (Stentz, 1995a; Stentz, 1995b) could plan the optimal path for static object perfectly. If we separate the dynamic object into several static objects and combine them, the static planner will recursively plan the optimal path from the goal to the starting position. If the dynamic object is far from the vehicle, the expense of planner is large. Illuminated by Mckeever (2002), here we proposed a layered approach to navigation which includes the global planner for efficiency and local planner for adjustment. The path planner here is extracted from D* algorithm, which constructs pointer variables called backpointer for each grid involved. The vehicle could follow the backpointers up to the goal. Taking account of the particularity of framed quadtree, the path is projected onto backpointers of the framed nodes. When the target is moving, the target is projected on the so called image on the cells for the vehicle to plan the path. The planner only needs to compute the local optimal path and adjust the backpointer locally.

In view of simplicity, the environment here is a 2-D plane and the position of dynamic object can be acquired exactly. Besides of that, the velocities of object and vehicle are constant and known. The map is constructed from the on-line digital map.

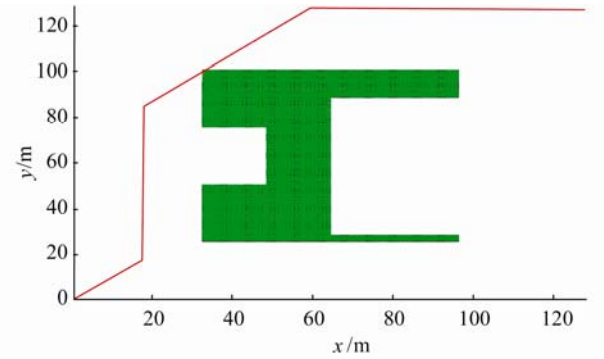
2 Map representation and construction

2.1 Map representation

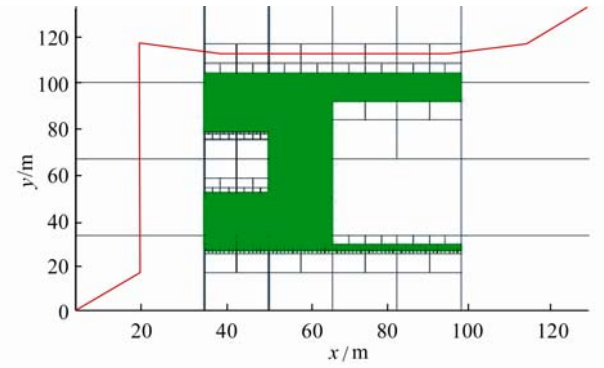
The data structure of map representation is important to the efficiency for metric path planning. For the kind of representation of discretization of space, the complexity and accuracy could be controlled through the resolution of cell. On the other hand, the completeness of algorithm also has correlation with it. Generally speaking, this kind of map is always be called as the roadmap (Choset *et al*., 2005).

The regular grid is the representation frequently used to discretize the environment with the equal sized cells. The space occupied by the method is proportional to the resolution and the data structure of regular grid is represented by adjacency matrix. The complexity of algorithm is proportional to the number of cells involved in the computation. As we can see from Fig.1(a), the planned path from the regular grid is the best of the three kinds of paths with the same starting position and goal. But the complexity of planning is much more than the others.

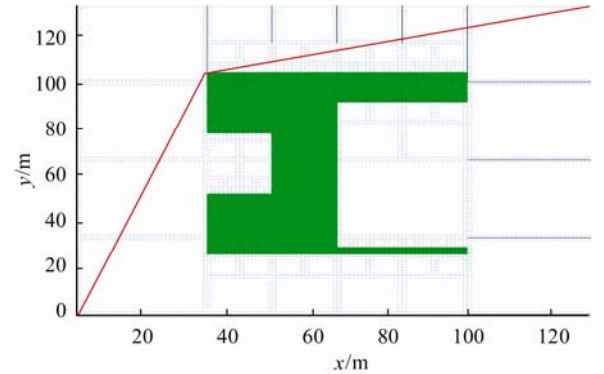
Quadtree is a transformation of regular grid, which could reduce the used space phenomenally. It subdivides the map into four equal sized quadrants, and the procedure will continue until each quadrant is completely filled with obstacle or space. The data structure of quadtree can be represented by a tree structure in which every root has 4 leaves. The complexity may decrease phenomenally compared with others, however, when the size of grid changes, as we can see in Fig.1(b), the shortest path has cost of optimality of the planned path to reduce the complexity.



(a) Regular grids and planned path



(b) Quadtrees and planned path



(c) Framed quadtrees and planned path

Fig.1 Examples of path planning with discretization of space

The framed quadtree which was firstly proposed by Chen *et al*. (1995) and then by Yahja *et al*. (1998) is shown in Fig.2. The

framed quadtree is constructed on the basis of quadtree, and framed cells are built for each quadtree grid. So this method could increase the precision compared with quadtree. The results could be seen in Fig.1(c).

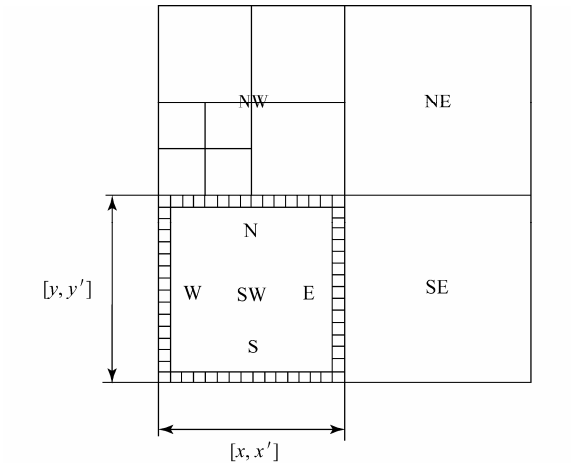
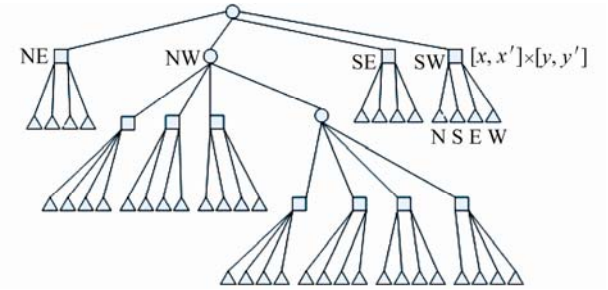
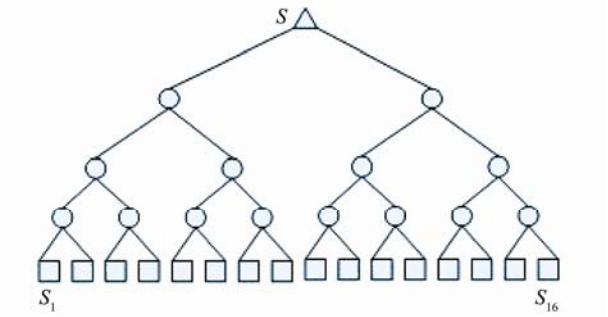


Fig.2 Framed quadtree

Here we propose a data structure to represent the framed quadtree, which is a multi-level tree. The main tree is quadtree, the leaves and knots in the tree correspond to every quadtree grid and the range of Euclid $[x,x']\times[y,y']$ as shown in Fig.2. Besides, there will be four edge nodes for every leaf to link to a balanced structure to represent the framed cells for each direction. The framed cells are arranged and combined with the balanced tree, as represented in Fig.3.



(a) Main quadtree



(b) Balanced tree for edge cells

Fig.3 Tree structure of framed quadtree

2.1.1 Depth of tree structure

For each framed cell on the framed quadtree, the depth in the multi-level tree is $\Theta(\log(s/c))$, where c is the resolution of the configuration space, s signifies the length of edge (for the meaning of $\Theta(n)$ and $O(n)$, refer to Levitin (2003)).

Proof: as every when the depth of quadtree increases by one, the size of edge of the grid will decrease in half. Assume the depth of framed cell is i on quadtree structure, the corresponding edge width of quadtree is $s/2^i$, the edge is constructed by balanced tree with the resolution; assume the depth of balanced tree is j, the size of framed cell is $1/2^j$ of the edge. There will be the equation of $c = s/2^{i+j}$, so the depth of framed cell is $\Theta(\log(s/c))$.

2.1.2 Storage of space used

For the environment with resolution of c and width of s. the storage used by regular grid is $\Theta((s/c)^2)$, storage of quadtree is $O(s/c+1)$, storage of framed quadtree is $O(s/c+2)$.

For regular grid, assume that the data structure is adjacency linked list, as the space needed is proportional to the number of grid, that is $(s/c)^2$, the space needed by regular grid is $\Theta((s/c)^2)$. Assume the depth of quadtree is d, then the sum of knots on quadtree is not more than $\sum_{i=1}^{d} 4^i = 4^{d+1}-1$, as the depth of quadtree is $O(\log(s/c))$, the space for quadtree is $O(s/c+1)$. Assume the depth of framed quadtree is d_1, the balanced tree is d_2, so the knots can not be more than $\sum_{i=1}^{d_1} 4^i \times \sum_{j=1}^{d_2} 2^j < \sum_{i=1}^{d_1} 4^i \times \sum_{j=1}^{d_2} 4^j$. With the depth of framed quadtree of $\Theta(\log(s/c))$, the space needed is $O(s/c+2)$.

2.2 Map construction

Traditionally the construction of discretized representation is processed through the computation of geometry computation of environment (Samet, 1988; Mark *et al*., 2008), it always decomposes the space through recursive iterations when it is constructed or increase the resolution necessarily. This method is straightforward; however, it is complex in computation and hard to analyze the resolution or completeness. That is to say, if we want to adjust the resolution when necessary, we have to process the whole procedure to make the discretization like before, it's time consuming and needed to be improved. Here we propose a way to construct the discretized representation of map especially for the framed quadtree structure and analyze and control the resolution when necessary. The method is generated from the quasi-random sampled way of constructing the roadmap for probabilistic roadmap (PRM) planner (Branicky *et al*., 2001).

For the uniform sampling strategy, it is the easiest theme to sample the environment. It can be proved to be complete. However, the sampling method is random and takes a large amount points which may be unnecessary. There are several ways to improve the performance the sampling strategy. Quasi-random is one of these methods (Lindemann and LaValle, 2005). Because the discretization of space is to find the proper size of squares and the center of squares is the representation of position about the square. We could use the quasi-random sampling method to construct the map.

For the regular grid, we suppose the map to be a unit square and set the discrepancy to be δ, the boundary of grid is a points set P, the close set of P is X, then the resolution should be:

$$\delta(P,\rho)=\sup_{x\in X}\min_{p\in P}\rho(x,p)$$

If we set the number of grids to be N, then according to Sukharev sampling criterion, the discrepancy also could be defined from the unit cube that:

$$\delta(P,\rho)\geq\frac{1}{2N^{\frac{1}{d}}}\rightarrow N\geq(\frac{1}{2\delta})^{d}$$

where the *d* represents the dimensions of configuration space. We can construct the regular grids with the discrepancy δ and the position of grid is set at the corner.

$$\Phi_j(n)=\sum_i a_{ij}p_j^{(i+1)}=a_{0j}+p_j a_{1j}+{p_j}^2 a_{2j}+\cdots,\qquad a_{ij}\in\{0,1\}$$

where the base p_j represents the sampling bases. In our setting the p_j is 0.5. We choose all the different binary value of a_{ij} in the Van der Corput sequences with the binary bits is i of every j; we will get all the position of grids. We then set the grids with the discrepancy $p_j^{(i+1)}$ at the every position of $(\Phi_1(n),\Phi_2(n),\cdots)$, which is called the Halton sequence. If we want to increase the resolution, then we only need to increase the number of i and take the similar procedures to add the new grids with the new discrepancy. The advantage of this way to construct the grids is that the construction is performed between the lower and upper halves and could be extended easily when the resolution is necessary to rise. The other important step is to decide whether the particular gird belongs to obstacle or not, it can be judged from the position of grids on the environment. This procedure doesn't have influence on the constructing of grids.

For the construction of quadtree the method is somewhat like the procedure of constructing the regular grids. The apparent difference lies in that the procedure of judging the obstacle or not will have influence on the constructing of grids. Generally speaking, the procedure is that, as the Van der Corput sequence is constructed from reversing the bits in the binary decimal representation of the naïve increasing sequence. If we regard that at some level of the grid is completely obstacle or free space, then there will be no need of sampling the sequential Halton points that lie between that span of unit discrepancy at that level. This method will decrease the complexity of constructing the map when the resolution increases. Besides of that, the procedure that judge whether the inner of grid is obstacle or not could be processed as that accumulating all of the lattices in the grids. This procedure is sequential in Halton sequence, so it will prevent the iterations.

For the construction of framed quadtree, the construction is that it also judges that whether the inner of the grid is completely obstacle or free space and records the level when the answer is true. Then there will be borderlines of the Halton sequence from that level. When sampling in the sequential Halton sequence, we introduce a procedure that judge whether the position is next to the borderline or not. It is like the sampling near the obstacles. If the occasion occurs, then the position will be kept in the sequence. Otherwise, it will be diminished in the sequence. The advantage of this method is that it also will decrease the complexity when the resolution increases.

After constructing the map of discretization, as the construction here is not like the iteration as before, so there is an additional procedure called finding the neighbors, that is to say, as our algorithm finding the path is through the connection of neighbors, we should define the set of nearby grid points for which an edge may be constructed. The procedure could be defined by the distance between points. Special care should be taken to boundary point. In our environment, there are at most four 1-neighborhoods:

$$N_1(q)=\{q+\triangle q_1,q+\triangle q_2,q-\triangle q_1,q-\triangle q_2\}$$

There are four 2-neighborhoods:

$$N_2(q)=\{q\pm\triangle q_1\pm\triangle q_2\}$$

3 Orientation of the goal on map

The position of goal changes with the chasing of vehicle. Assume that the exact position in real world can be acquired exactly through the sensors on vehicle. The precondition of path planning is that the goal should be projected from real world onto map. Taking the regular grid for example, the center of one regular grid can represent the position of vehicle, as the regular grids have the highest resolution in representation, the precision of goal on map is the highest. But for the quadtree, as the position lies in the center of quadtree grid, when the size of the grid is large somewhere, the precision of orientation is low, which is one of the reasons that the generated path is less optimal.

With respect to framed quadtree, we also represent the goal's position with the center of grid where the goal locates. If the

goal is in framed cells, the precision is high; or otherwise, the precision is low as in the structure of quadtree. To increase the precision of object's position in map, one possible choice is to set the goal as the obstacle included in the environment and decompose the grid and update the framed quadtree to increase the precision. This method is satisfying when the goal is static. However, when the goal is dynamic, it will cost plenty of time to update the data structure. Here we propose a way of projecting the position of goal onto framed cells. The process is shown in Fig.4.

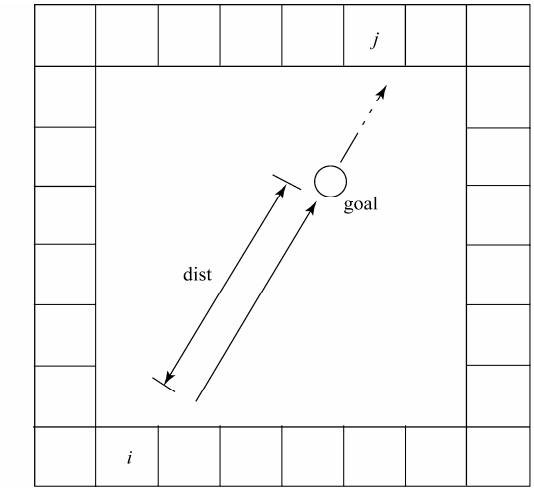


Fig.4 Project the goal onto framed cells

The algorithm is:
OBJECTINQUARD(Objectp)
L1: Find the quad node Objectp belongs
L2: for all frame nodes *i* in quad node
L3: Compute the dis(*i*) from *i* to Objectp
L4: INSERTQUEUE(*i*, dis(*i*))
L5: Compute the diagonal frame node *j*
L6: backpointer(*i*)←*j*

By taking the goal's position in real world as the input, the process initializes the path planner, computes all the distances from framed cells in the quadtree grid to the position of goal and then inserts them in the priority queue (L3~L4) and then sets the backpointer pointing to the diagonal framed cell (L5~L6).

4 Path planner for dynamic object

D* algorithm is a transformation of Dijkstra algorithm proposed by Stentz, in which each grid stores a backpointer to direct the motion of vehicle. Our planner chasing dynamic object is based on the property of that. Every when coming to a new location of goal, it only needs modification by the algorithm computing the path between the positions of goals as shown in Fig.5. If the distance between goal and vehicle is large, then this procedure could reduce the cost phenomenally. The idea was firstly proposed by McKeever (2002). The procedure is used to make modification according to the property of framed quadtree. First we analyze the efficiency of algorithm which is used to find the neighbor nodes and we can find that it has great correlation with data structure.

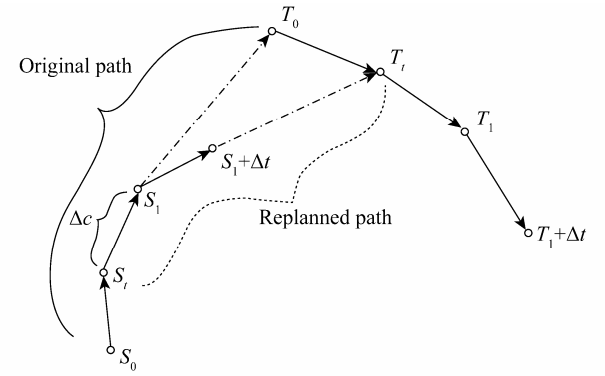


Fig.5 The local planner for moving target

4.1 Time consumption of algorithm

The time used to find the neighbor node for regular grid is $\Theta(s/c)$, the quadtree is $O(\log(s/c)+1)$, and the framed quadtree is $\Theta(\log(s/c)+1)$.

The proof of this lemma can be represented as the same way as in the section 2.1.

4.2 Path planner

The planner can be summarized as follows. There are several variables used in the algorithm, including the priority queue queue used in planner, the backpointer used to direct the vehicle, the current position of goal Objectp, the prior position ObjectpOLd, and the position of vehicle source.

INITIALIZE()
L1 ObjectpOld←Objectp
L2 Build the framed quadtree of map
L3 OBJECTIONQUARD(Objectp)
L4 Compute the shortest path by D*

Main()
L1 INITIALIZE()
L2 Vehicle follows the backpointer
L3 while vehicle does not catch the goal
L4 Find the object position Objectp
L5 if Objectp is in the same quad node with ObjectpOld
L6 for all frame nodes i in quad node
L7 Find the new diagonal node *j*
L8 backpointer(*i*)←*j*
L9 else
L10 Compute the Objectp and ObjectpOLd
L11 UPDATEBACKPOINTER()
L12 Vehicle follows the backpointer

The planner utilizes the function of INITIALIZE() to construct the data structure of framed quadtree and initialize

the priority queue of algorithm and compute the initial path for vehicle (L1~L4). In the main function, the vehicle follows the backpointer to goal (L1, L12). If the prior goal position and current position are in the same quadtree grid, it only needs to update the framed cells in that grid without the necessity to compute the new path (L5~L8). If they are not in the same grid, we need to update the backpointer through D* algorithm by computing the distance between the goals (L9~L11). The vehicle follows the backpointer until it reaches the goal.

5 Simulation

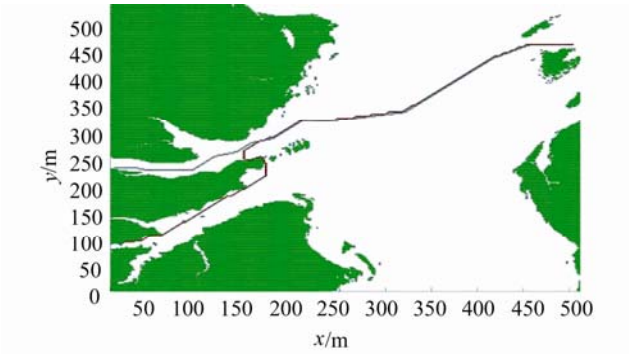
In order to verify the validity of our proposed planner, we used the digital ocean map from U.S. National Geophysical Data Center about Nargansett Bay. Throughout the discretization, the map can be represented with regular grids. Fig.6(a) shows the original digital maps and Fig.6(b) shows the corresponding obstacle maps if we set some depth of water as the limit to build the map for navigation. Then the digital map is only a 2-D map which could represent the free space for vehicle. As we here can set up the regular grid with the input data from digital map and the direction of y axis is opposite to the latitude compared with the latitude. The basic unit along x and y axis is one meter per discrepancy.

(a) The original map

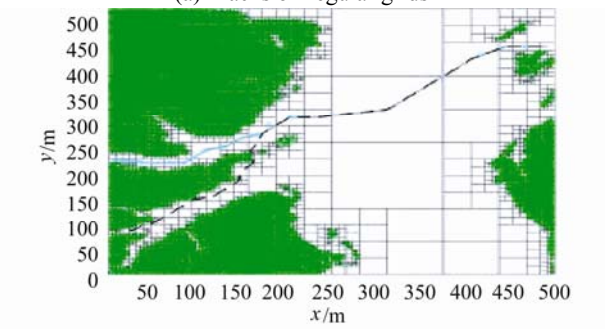
(b) The obstacle map

Fig.6 The simulation environment

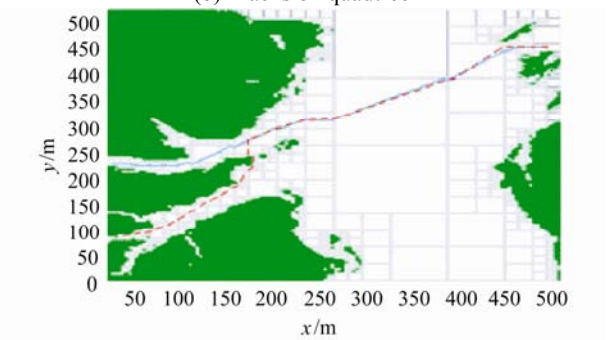
Having constructed the regular grids, the quadtree and framed quadtree can be built from the regular grids. In the same work space, with the same dynamic object and vehicle, the paths planned by three kind of representation are shown as follows.



(a) Tracks on regular grids



(b) Tracks on quadtree



(c) Tracks on framed quadtree

Fig.7 Tracking path from different structures

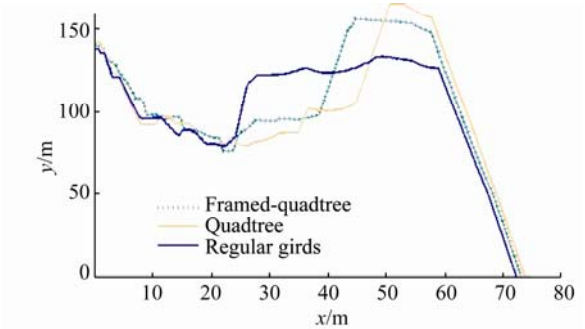


Fig.8 The distance between goal and vehicle

From Fig.7 and Fig.8, it can be concluded that the quality of the path planned from the vehicle has great relation to the resolution of the map. Therefore, the path from the regular grid has the most excellent result above all, the result from framed quadtree is worse, the quadtree is the worst. However, from Table 1 and Table 2 it can be seen that, although the path of regular grid could have the best result about the path, the cost of the computation, such as memory used for storage and

time used for computation, is much more than the other two. By comparison, the framed quadtree could reduce the cost of planning the path phenomenally and keep a better result for the vehicle at the same time. Table 1 and Table 2 show the results in simulation.

Table 1 Memory used for map

Kind of map	Regular grid	Quadtree	Framed quadtree
Memory used(bytes)	282 144	768	41 728

Table 2 Time used for planner

Computing time/s	Regular grid	Quadtree	Framed quadtree
1	5.3	0.025	0.3
11	6.2	0.01	0.3
21	0.8	0.02	0.2
31	8.2	0.005	0.4
41	7.3	0.005	0.2
51	7.3	0.15	0.2
61	2.8	0.005	0.1

6 Conclusion and Future work

In this paper, a kind of map representation called framed quadtree is introduced and a kind of data structure is proposed to store it. The reason for using it in path planning is that the regular grid has such disadvantages as high in the cost of computing, and quadtree has less accuracy. Through the analysis and simulation, it can be seen that our aim could be achieved by the proposed path planner.

As we can see from the construction of discretized representation of map, the representation is generated from the traditional representation, so the size of the grid is limited and not necessarily the best representation of the environment, the structure of the map may be improved by different shape but the generated path may be worse. We may take some proper sampling strategy to build the representation of map.

As we can see from the designed path in the Fig.7, the path is not like our real optimal path in the real world, there should be some additional procedure to improve the performance after the path planning in our future work.

References

Branicky MS, LaValle SM, Olson K, Yang LB (2001). Quasi-randomized path planning. *IEEE International Conference on Robotics and Automation*, Seoul, Korea, 1481-1487.

Chen D, Szczerba R, Uhran JJ Jr. (1995). Planning conditional shortest paths through an unknown environment: a framed-quadtree approach. *Proceedings of the IEEE International Conference on Robotics and Automation*, Albuquerque, USA, 33-38.

Choset H, Burdick J (2000). Sensor based motion planning: the hierarchical generalized Voronoi graph. *International Journal of Robotics Research*, **19**(2), 96-125.

Choset H, Lynch K, Hutchinson S, Kantor G, Burgard W, Kavraki LE, Thrun S (2005). *Principles of robot motion: theory, algorithms and implementation*. The MIT Press, Cambridge, 7-101.

Dijkstra EW (1959). A note on two problems in connexion with graphs. *Numerical Mathematics*, **1**(1), 269-271.

Latombe J (1991). *Robot motion planning*. Springer-Verlag, New York, 1-40.

Levitin A (2003). *Introduction to the design & analysis of algorithms*. Addison-Wesley, New York, 41-43.

Lindemann S, LaValle SM (2005). Current issues in sampling-based motion planning. *Robotics Research: The Eleventh International Symposium*, Springer-Verlag, Berlin, 36-54.

Lozano P (1983). Spatial planning: a configuration space approach. *IEEE Transactions on Computer*, **C-32**(2), 108-132.

Mark de Berg, Cheong O, Van KM, Overmars M (2008). *Computational geometry: algorithms and applications*, Springer-Verlag, New York, 11-21.

McKeever S (2000). Path planning for an Autonomous vehicle. Master thesis, Massachusetts Institute of Technology, Cambridge, USA, 25-45.

Murphy R (2000). *Introduction to AI robotics*. MIT Press, Cambridge, 200-281.

Pearl J (1984). *Heuristics*. Addison-Wesley, New York, 33-73.

Rohert H (1986). Shortest path in the plane with convex polygonal obstacles. *Information Processing Letters*, **23**, 71-76.

Samet H (1982). Neighbor finding techniques for images represented by quadtrees. *Computer Graphics and Image Processing*, **18**(1), 37-57.

Samet H (1988). An overview of quadtrees, octrees, and related hierarchical data structures. *NATO ASI Series*, **40**, 51-68.

Stentz A (1995a). Optimal and efficient path planning for unknown and dynamic environments. *International Journal of Robotics and Automation*, **10**(3), 89-100.

Stentz A (1995b). The focused D* algorithm for real-time replanning. *Proceedings International Joint Conference on Artificial Intelligence,* Montreal, Canada, 1652-1659.

Yahja A, Stentz A, Singh S, Brumitt L (1998). Framed-quadtree path planning for mobile robots operating in sparse environments. *Proceedings of the IEEE Conference on Robotics and Automation*, Leuven, Belgium, 650-655.

Bo Gao was born in 1982. He is a candidate for PhD at the Northwestern Polytechnical University. His current research interests include path planning for robot, industrial control.

De-min Xu was born in 1937. He is an academician of Chinese Academy of Engineering. He is a professor at the Northwestern Polytechnical University. His current research interests include precision guidance, control and simulation of fish torpedo, *etc*.

Wei-sheng Yan was born in 1968. He is a professor at the Northwestern Polytechnical University. His research interests include precision guidance and control of AUV, *etc*.