The Nucleus **51**, *No. 4 (2014) 397-404*

www.thenucleuspak.org.pk

The Nucleus

ISSN 0029-5698 (Print)
ISSN 2306-6539 (Online)

Selection of Optimal Path Planning Algorithm for Autonomous Robots in Structured Environment

H. Bashir and M.H. Yousaf[*]

Department of Computer Engineering, University of Engineering and Technology, Taxila, Pakistan

ARTICLE INFO

Article history:
Received : 27 December 2013
Revised : 17 October 2014
Accepted : 20 October 2014

Keywords:
Autonomous robot path planning
Optimal paths
Shortest path algorithms
Cost search algorithm

ABSTRACT

To determine a collision free path for a robot between start and goal positions in an environment filled with obstacles is a very challenging task in the design of an autonomous robot path planning. This paper aims to select an optimal path planning algorithm for a mobile robot in structured environment. To achieve the goal, comprehensive strengths and weaknesses of different path planning algorithm are discussed and evaluated. A wooden box with some fixed obstacles and robot inside it is basically the environment. Information about the environment is used to build a roadmap or graph of the environment. After getting a convenient representation of the environment, then graph search methods can be used to obtain shortest possible path through this roadmap. It is well known that computing shortest paths for autonomous robots is an important task in many path planning applications. Selecting a suitable algorithm from the various algorithms reported in the literature is a decisive step in many applications including path planning task. A set of three shortest path algorithms that compute optimal path from start to goal location has been identified and these are Uniform Cost Search, Greedy Search and A* algorithm. After comparing execution time and path length of path computed by these three algorithms, A * algorithm proves to be best suited for this particular application of path planning.

1. Introduction

Robot path planning or motion planning can be classified as a class of algorithms that accept high level description tasks and generate valid and accurate path combinations for the robot to follow. In simple words, path planning can be taken as a task in which the robot, whether it is a mobile robot has to navigate from its start point to a specific destination or goal point by preventing collisions with the obstacles in its way. Path planning has helped to solve a huge number of problems in the modern world. Surgical planning [1], automation, animation of digital characters [2], autonomy, mapping of unexplored environments and drug design [3] are some of the major applications of path planning.

Piano mover's problem [4] is one of the traditional version of path planning in which a path planning algorithm accepts a piano and a precisely computer aided designed house as input. The algorithm then has to compute a path for moving the piano from one location to another location inside the house avoiding collisions with the walls of the house and with the obstacles. In this case the obstacles are assumed to be fixed and their exact locations are known. Thus an accurate path can be planned from start location to goal location. Planning is done before execution that’s why termed as off line path planning. Another case can be when the environment is dynamic and robot comes across an obstacle in its path from start to goal. In that case, re-planning of path is required. So this type of planning is called as on line path planning and is common in outdoor scenes and unstructured environments.

First step is always to identify obstacles so segmentation is very important phase of path planning. Different segmentation techniques are used in the field of robotics depending on the requirement of application. Researchers have used threshold based methods [5] for image segmentation in structured environments, when the goal is to identify few objects based on their color properties. This is case where specific colors are assigned to the objects in the environment.

Then, region based segmentation [6] for object recognition has also been used by the researchers. For unstructured environments, a segmentation technique, proposed by Sundaresan and Konolige, which combines both texture and color information [7] has been used. In case of indoor environments, another technique has also been proposed by the researchers, in which the robot will fixate at various points in the scene and then perform the segmentation of the object containing the fixation point [8].

Segmentation provides the location of obstacles then corner points of these obstacles must be detected in order to create visibility graph. Various corner detection techniques exist in literature. Harris corner detector [9] is one of the popular corner detector techniques and it is based on the

[*] Corresponding author : haroon.yousaf@uettaxila.edu.pk

signal local auto correlation function where this function calculates the local changes of the signal with patches that are shifted in different directions by small amount. Susan [10] another detector proposed by Smith and Brady which uses a circular mask for edge and corner detection. A technique is proposed in [11], in which the total curvature of the gray level image is directly proportional to the second-order directional derivative in the direction to edge normal and curvature is inversely proportional to the strength of edge. A multi scale algorithm has been proposed by Rattarangsi and Chin [12] that is based on curvature scale space (CSS) which helps in corner detection of planar curves. Corner detector based on local and global curvature properties [13] is another technique which detects both fine and coarse features precisely at low computational cost.

After corner detection, obstacles are identified so it makes sense to first obtain a data structure i.e. a graph or map of the environment and searching algorithm then uses this information to find the optimal path. So, it is required to build a graph or map of the environment before implementation of any path planning algorithm. Road maps are included among the various classes of topological maps [14, 15]. These are basically graph like structures in which edges represent adjacency relationship between the nodes and nodes represent some distinguished features. Another prominent technique is one proposed in [16] in which they use the concept of rotation trees. After that, another technique proposed by Ghosh and Mount [17] in which they introduced a planar scan technique using funnel and triangulation splits to achieve running time of O (e+nlogn).Another popular technique for generating a roadmap is Voronoi diagram [18].

In road map techniques, another notable technique used by researchers is cell decomposition technique. Cellde composition methods present another mean of representing the free space of environment. Notable exact cell decomposition techniques include the trapezoidal decomposition technique [19] which basically depends on the polygonal representation of the configuration space. Moarse decomposition [20] is another method which belongs to a general class of exact cell decomposition and is used for non planar configuration spaces and non polygonal obstacles.

The output of these methods is either a graph or a map containing the possible path combinations to reach the destination point from the start point. The two basic search algorithms are Depth-first and Breadth-first algorithms [2]. Dijkstra's algorithm is a type of breadth first search algorithm conceived by Dutch computer scientist Edsger Dijkstra's in 1959. Uniform-cost search (UCS) is a tree search algorithm used for searching a tree structure, weighted tree, or graph to find optimal path. Best first search algorithm is a greedy algorithm which optimizes the breadth first search in a way that it has some heuristic estimate of distance of a node from the destination. A* is a very well-known algorithm developed by Hart et al. [20]. It is an extension of Dijkstra's algorithm that tries to minimize the number of nodes to be explored by incorporating in its search, the heuristic estimate of the cost to get to the destination node from a given node. So it is combination of both best first search and Dijkstra's algorithm.

Some motivational examples are used to show the need for studying the path planning algorithms. Path planning algorithms have widespread success and usage in the field of automation, mobile robotics, aerospace applications and manufacturing. Progress in these fields regarding the path planning indicates much more fascinating applications in coming years. Some of the path planning applications includes:

- Toyota Humanoid robot that works by planning the path in order to grasp specific objects avoiding the obstacles in its way.
- 'Opportunity' robot that is sent by NASA on Mars for research purposes. In order to maneuver itself on the planet, the robot combines vision with path planning to avoid the obstacles.

2. Problem Statement

Automatic path planning for autonomous robots is a very challenging task. Generating optimal paths for autonomous robots is one of the difficult topics in mobile robotics applications. The environments in which the robots work can be structured or unstructured. Cameras are used mostly to sense environment which is structured in this particular case as obstacles are fixed and no re-planning is needed. Once the information about the environment is known, then next step is to build a roadmap or graph of the environment. For building the roadmaps, Different algorithms usually exist in this field of path planning. After a convenient representation of the environment is obtained, then graph search methods can be used to find the optimal paths through this roadmap.

It is well known that computing shortest paths for autonomous robots is an important task in many path planning applications. Selecting a suitable algorithm from the various algorithms reported in the literature is a decisive step in many applications including path planning task. A set of three shortest path algorithms that compute optimal path from start to goal location has been identified. These three algorithms are:1) Uniform Cost Search 2) Greedy Search Algorithm 3) A* Search Algorithm [2] and then performance evaluation of these algorithms is also a challenging task.

3. Proposed Methodology

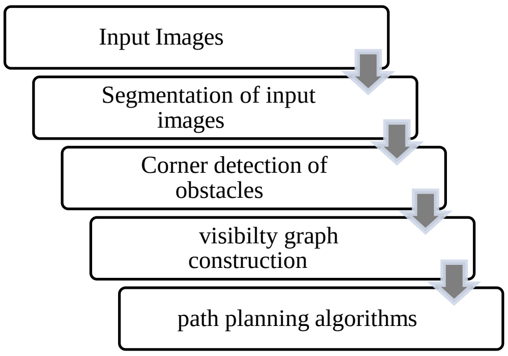

Input Images
Segmentation of input images
Corner detection of obstacles
visibilty graph construction
path planning algorithms

Fig. 1. Proposed methodology.

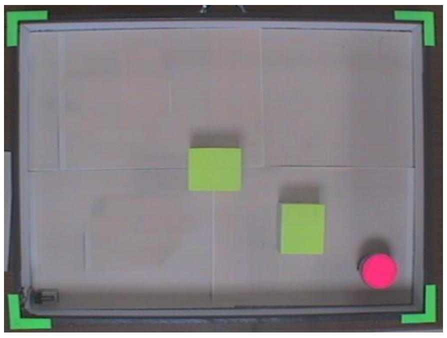
Fig. 1 illustrates the basic methodology that is used in research work. First step is the segmentation of input image captured from camera to separate free space from obstacles. Next step is the detection of corner points of obstacles and these corner points serve as nodes for visibility graph construction. Visibility graph provides all the possible paths from start to goal point and then path planning algorithms find the optimal path for robots to follow from start to goal point. Detailed methodology is explained in the following sub-sections.

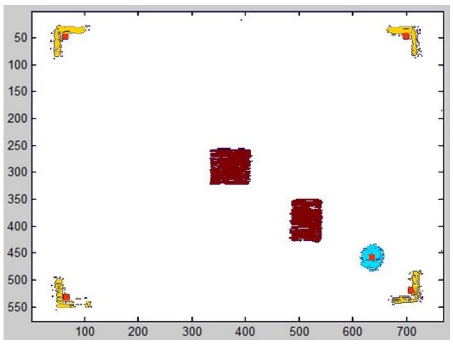
3.1 Segmentation

The first step is to segment the image captured through the camera to separate obstacles from free space. Fig. 2a shows the input image. Obstacles are represented by light green color in input image, robot by magenta color and boundaries by dark green color. A common thresholding approach was used and it performed quite well in desired regions segmentation. The image obtained from the camera is an RGB image as shown. Firstly RGB image is converted to HSV image. After that in order to select the threshold values for the three colors, any homogeneous part of the image that only contains the required color is selected and the maximum and minimum values of the intensity, hue, and saturation are calculated from this region. These values serve as the threshold values.

This selection of threshold values can be done every time an image is obtained or preset threshold values (obtained at the start from an image) can be used for segmenting all the remaining images captured through the camera. The former way is better in case when the lightening conditions are varying a lot during the time images are being captured. Due to this, the HSV values of the pixels can change and effect the segmentation. On the other hand, segmentation is not affected by small variations in lightening conditions and fixed threshold values can be set in the beginning. After selecting the threshold values, checking of all the image pixels is done and if pixel HSV values fall within the threshold values of any of the three color classes, a specific value is assigned to that pixel and consequently pixel is represented by unique color in the output image as shown in Fig. 2b.

Alongwith the segmentation it is also important that robot should not collide with the obstacles present in environment. So, configuration space obstacle should also be considered. Configuration space obstacle comprises of robot configurations in which collision with obstacles can occur. The robot is circular in current case and the robot center is taken as a reference point. As a result sliding the robot around the obstacles and curve traced by reference point will provide the configuration space obstacle. An easy way to achieve this configuration space is to dilate the obstacles in image. Dilation adds the pixels to the boundary of obstacles in the image. A structuring element is used to define the way pixels that are going to be added. Curve traced by robot around obstacles is circular so a disk shaped structuring element can be used but in the current case, dilation of the obstacles in image is done with a square shaped structuring element in order to get the details related to corners because these corner points are needed to build visibility graph. In order to keep the robot at safe distance from the obstacles, the number of pixels was multiplied with a factor 1.5. So in this manner, the obstacles boundary in the image was dilated. The corners of the obstacle will be calculated by considering this virtual boundary. Thus when the robot will follow the path to goal point, it will follow these corner points i.e. nodes in the visibility graphand keep itself at a safe distance from the original obstacles.


(a)



(b)

Fig. 2. (a) Original image, (b) result obtained after segmentation.

3.2 Corner Detection

The next step is to detect the corners of the obstacles. Curvature based edge detector [11] is used for corner detection. Binary edge map is the output of the canny edge detector. In case if there is only one obstacle in the image then this means image will be having only one contour and the points obtained through edge detection will represent this contour. In multiple obstacles case, it is all the contours are required to extract and to know which edge points belong to which contour.

A duplicate image (BW1) is created for this purpose from the original binary image (BW). Then all the edge points (value=1) are extracted from this image. The first edge point is selected which will be part of a contour in the image. The value of this edge point is set to zero in image BW1 and this edge point is saved. Then 3×3 neighborhood of this edge point is searched for more edge points. All the points found in the neighborhood are then saved and 3×3 neighborhoods of these newly found points are searched. In this manner the boundary of contour is traced by searching the neighborhoods of edge points on contour until all the edge points on a contour have been explored. The edge points of this contour are then saved and let it be contour 1. Zero value has been given to all these edge points in the search image (BW1) so that these are not selected again. In other words, the contour that has been searched is removed from the image. Then using the same procedure, selection of an edge point of one of remaining contour will be made and then all edge points of this contour will be saved. In this manner, this procedure goes on until the edge points on all contours have been extracted. The stored edge points basically provide the x and y coordinates of the pixels that lie on the contours. Smoothing of these edge points is then performed using a Gaussian function with σ = 3 [11].

After the contours have been obtained, the next step is the calculation of the curvature value of pixel of each contour. The curvature value for an edge point will be lower as compared to that of a corner point. The formula for calculation of curvature is

$$K_i^j = \frac{\Delta x_i^j \Delta^2 y_i^j - \Delta^2 x_i^j \Delta y_i^j}{\left[(\Delta x_i^j)^2 + (\Delta y_i^j)^2\right]^{1.5}} \qquad \text{for } i = 1,2,\dots,N \qquad (1)$$

Where x_i^j and y_i^j are the coordinates of ith pixel on jth contour. $\Delta x_i^j = \frac{x_{i+1}^j - x_{i-1}^j}{2}$, $\Delta y_i^j = \frac{y_{i+1}^j - y_{i-1}^j}{2}$, $\Delta^2 x_i^j = \frac{\Delta x_{i+1}^j - \Delta x_{i-1}^j}{2}$ and $\Delta^2 y_i^j = \frac{\Delta y_{i+1}^j - \Delta y_{i-1}^j}{2}$.

By using above equation, all the local maxima of the curvature function will give us the initial list of corner candidates [11].

3.3 Visibility Graph

A visibility graph is constructed using initial and goal locations and the corner points calculated in previous step and. Before moving to the next step i.e. construction of visibility graph, the convex hull of the obstacles is calculated. The convex hull of a geometric object (such as a polygon or a point set) is the smallest set of points containing that object. The reason for generating convex hull is that the robot will always follow an optimal path to reach the destination. Equation for calculating convex hull [22] is given as:

$$K_j^i = \left(K_{j-1} \times x^i\right) \cup y \qquad (2)$$

In order to reduce the computational time, a technique was implemented termed as D.T Lee's rotational plane sweep algorithm [20]. This technique was found in most of the literature regarding the visibility graph and is considered as a very accurate technique. Fig. 3 represents the final result of rotational plane sweep algorithm after corner detection of obstacles shown in test image of Fig. 2a.

The implementation of whole algorithm in this research work is done in a way that first visibility among the vertices of all the obstacles is checked and then visibility from the start and goal location is checked.

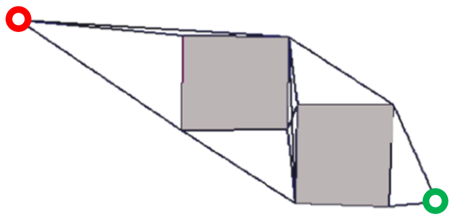


Fig. 3. Visibility graph construction.

3.4 Path Planning

The visibility graph gives us all the possible paths that robot can follow to reach the goal point. The next task is to find the optimal path i.e. shortest path among all these possible paths.

For this purpose following three algorithms are used.

3.4.1 Uniform Cost Search Algorithm

The start point, vertices of the obstacle and goal point are represented by nodes of the visibility graph and lines which are used to join them represent all possible paths. The search begins at the start node. The algorithm continues its search by exploring the next node which has the least cost from the start node. Nodes are visited in this manner until a goal state is reached. Cost is represented by g(n) and it is calculated by computing the distances to reach from one node to other.

The start node starts this algorithm and priority queue of the nodes to be visited alongwith their costs (value of g (n)) are also maintained. This priority queue is termed as “open list”. Another list called “closed list” is also used. Closed list includes the nodes that have been visited and it also holds the back pointer to the visited node. This back pointer points to the node (parent node) from which visited node originated.

At each iteration, from the open list, a promising node (one with lowest g (n) value) is selected. The successor nodes of the current node are determined and are then added to the open list if these successor nodes have not been visited yet. The current node alongwith its back pointer is removed from the open list and then this node and its pointer is added to the closed list. In this manner the algorithm repeats this process of exploring until the destination i.e. goal node is reached or the open list becomes empty. The path is then traced through back pointers starting from the back pointer to goal node and then chaining back until the first point i.e. start node is reached as shown in Fig. 4. Green line represents the path followed by robot.

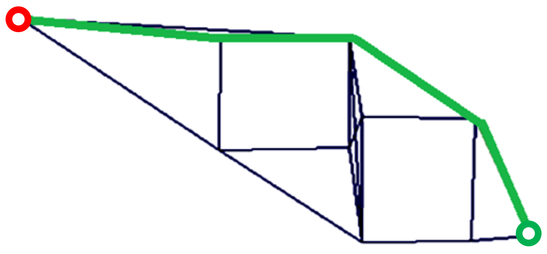


Fig. 4 UCS algorithm output. Green point indicates the start point and red point indicates the goal point and green line indicates the path.

3.4.2 *Greedy Search Algorithm*

A heuristic estimate is introduced in this search and it is given by h (n). The heuristic estimate (h (n)) is the straight line distance between the nodes and goal point assuming no obstacles exist between the given node and goal node and given as:

$$h = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2} \tag{3}$$

Where x and y are the coordinates of two nodes a and b. The search continues by visiting the next node which has the least value of h (n). Nodes are visited in this manner until a goal state is reached.

The start node starts this algorithm and priority queue of the nodes to be visited alongwith their heuristic estimates are also maintained. This priority queue is termed as “open list”. Another list called “closed list” is also used. Closed list includes the nodes that have been visited and it also holds the back pointer to the visited node. This back pointer points to the node (parent node) from which visited node originated.

At each iteration, from the open list, a promising node (having least value of h (n)) is selected. The successor nodes of the current node are determined and are then added to the open list if these successor nodes have not been visited yet. The current node along with its back pointer is removed from the open list and then this node and its pointer is added to the closed list. In this manner the algorithm repeats this process of exploring until the destination i.e. goal node is reached or the open list becomes empty. The path is then traced through back pointers starting from the back pointer to goal node and then chaining back until the first point i.e. start node is reached as shown in Fig. 5. Green line represents the final path followed by robot.

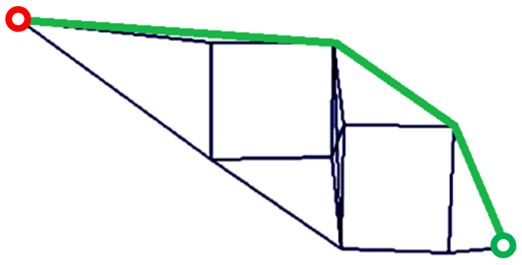


Fig. 5. Greedy search algorithm output. Green point indicates the start point and red point indicates the goal point and green line indicates the path.

3.4.3 *A* Algorithm*

A* search is based on an evaluation function f (n) that depends on the values of both these functions and its value is given as:

$$f(n) = g(n) + h(n) \tag{4}$$

The start node starts this A* algorithm and in this algorithm priority queue of the nodes to be visited along with their costs (value of f(n)) are also maintained. This priority queue is termed as “open list”. Another list called “closed list” is also used. Closed list includes the nodes that have been visited and it also holds the back pointer to the visited node. This back pointer points to the node (parent node) from which visited node originated.

At each iteration, from the open list, a promising node (one with lowest f(n) value) is selected. The successor nodes of the current node are determined and are then added to the open list if these successor nodes have not been visited yet. The current node alongwith its back pointer is removed from the open list and then this node and its pointer is added to the closed list. In this manner the algorithm repeats this process of exploring until the destination i.e. goal node is reached or the open list becomes empty. The path is then traced through back pointers starting from the back pointer to goal node and then chaining back until the first point as shown in Fig. 6. Green line represents the final path followed by robot.

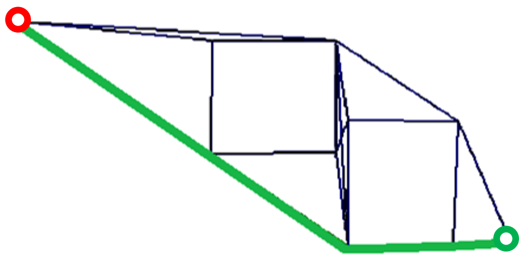
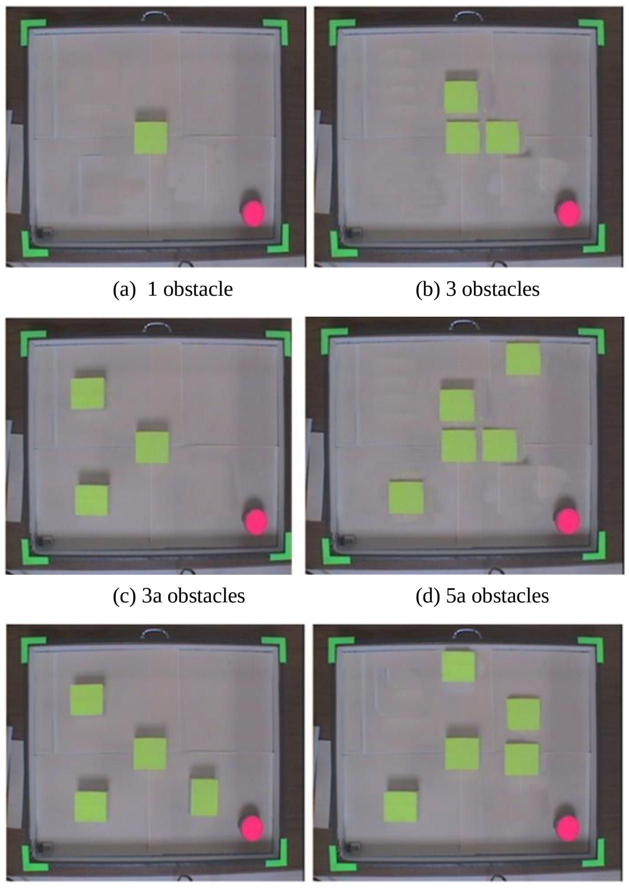


Fig. 6. A* Algorithm output. Green point indicates the start point and red point indicates the goal point and green line indicates the path.

Euclidean length of path is then calculated for each algorithm. Length of path as well as execution time obtained for A* algorithm is less as compared to greedy search and UCS algorithm. So, both UCS and Greedy search algorithms fails to provide optimal solution for the test image shown in Fig. 2a but A* provides the shortest path for robot to follow and to reach its destination avoiding obstacles.



(a) 1 obstacle (b) 3 obstacles

(c) 3a obstacles (d) 5a obstacles

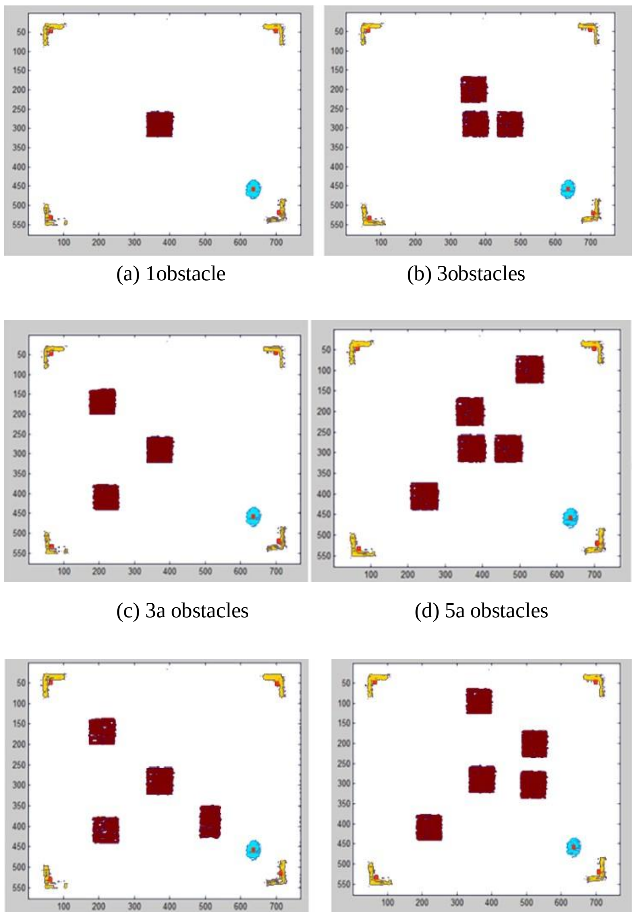
(e) 4 obstacles (f) 5 obstacles

Fig. 7. Test images.

4. Experimental Results and Discussions

Six different test images are shown in Fig. 7 which are used for experimentation. Results obtained after segmentation are shown in Fig. 8. After corner detection the visibility graph construction is shown in Fig. 9, where green point indicates the start point and red point indicates the goal point. Path planning algorithm results are shown in Fig. 10 where red line indicates UCS output, yellow line indicates Greedy Algorithm output and Green line indicates A * algorithm output.

All the three algorithms used for path planning are compared by varying no of obstacles. Table 1 and Table 2represent the performance of all the three techniques used for computing shortest path. n is no of obstacles in test images, N represents no and Y represents yes. These three algorithms are evaluated on the basis of two main factors, time complexity and whether they are able to compute optimal solution or not. Euclidean length of path is a decisive factor to find which algorithm provides optimal solution. It can be seen in Table 1 that length of path computed from A* is always less than UCS and Greedy



(a) 1obstacle (b) 3obstacles

(c) 3a obstacles (d) 5a obstacles

(e) 4 obstacles (f) 5 obstacles

Fig. 8. Segmented output.

Search. Thus A* algorithm always provide optimal path for robot to follow. In case when two algorithms compute same path length then execution time is decisive factor to find best solution for example first case shown in Table 1, Greedy search and A* algorithms compute same path length but execution time of A* is still less as compared to greedy search algorithm. Thus A* is best solution.

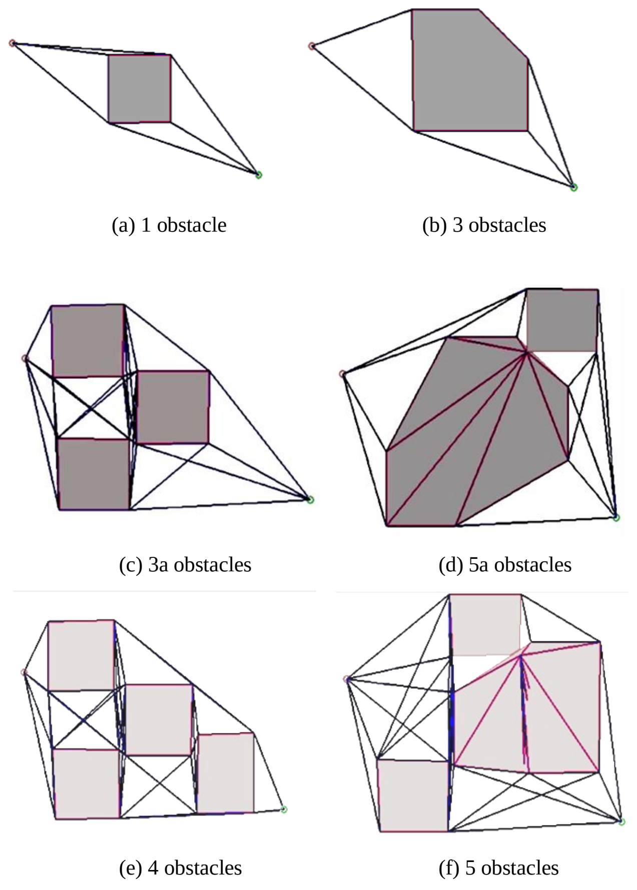

(a) 1 obstacle (b) 3 obstacles (c) 3a obstacles (d) 5a obstacles (e) 4 obstacles (f) 5 obstacles

Fig. 9. Visibility graph construction.

When images containing one obstacle and three obstacles are used then A* and greedy search algorithms both provides optimal solution but computation time of A* is less as compared to UCS and Greedy Search algorithms shown in Table 2. In case of image containing 4 obstacles A* and UCS algorithms both provides optimal solution but again execution time of A* algorithm is less. Thus A* provides the optimal result for all the data set provided with least computation time but other two algorithms fails to do so.

A graph is shown in Fig. 11 that provides comparison of execution time for all the three path algorithms used in experiments. Green line indicates the time taken by A* Algorithm, UCS and Greedy Search Algorithms are represented by red and blue line respectively. It can be noted from the graph that A* provides the optimal path for robot in shortest time as its execution time is always less than other two algorithms. So it is best suited for this particular application. Time taken by other two algorithms is almost similar but greater than computation time of A* algorithm and they fail to provide optimal solution in most cases.

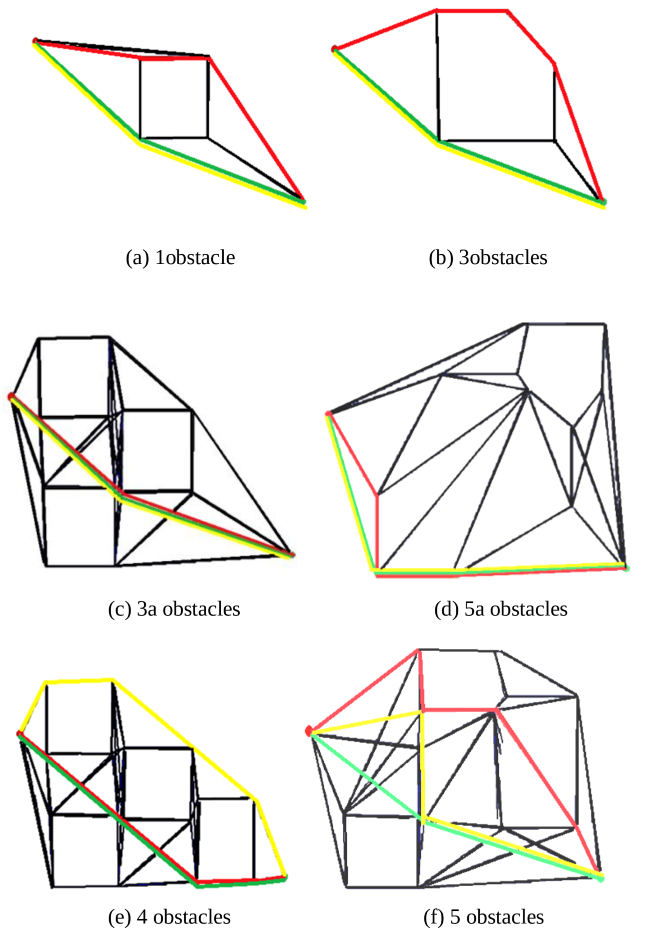

(a) 1obstacle (b) 3obstacles (c) 3a obstacles (d) 5a obstacles (e) 4 obstacles (f) 5 obstacles

Fig. 10. Output of path finding algorithms.

Table 1. Comparison of path lengths.

No.	No of corners detected	Length of path			Optimal Solution		
		UCS	Greedy search	A*	UCS	Greedy search	A*
1	4	109.6	109.5	109.5	N	Y	Y
2	8	113.8	113.7	113.6	N	N	Y
3	6	110.3	110.1	110.1	N	Y	Y
3	12	122.0	122.0	122.0	Y	Y	Y
4	16	127.4	128.2	127.4	Y	N	Y
5	14	118.1	118.0	118.0	N	Y	Y
5	20	130.8	130.7	130.6	N	N	Y

Table 2. Comparison of execution time.

No.	No. of Corners Detected	Time(s)		
		UCS	Greedy Search	A*
1	4	0.832	0.854	0.798
2	8	1.001	1.011	0.994
3	6	0.874	0.877	0.766
3	12	1.447	1.458	1.398
4	16	2.285	2.295	2.123
5	14	1.878	1.986	1.532
5	20	2.310	2.327	2.212

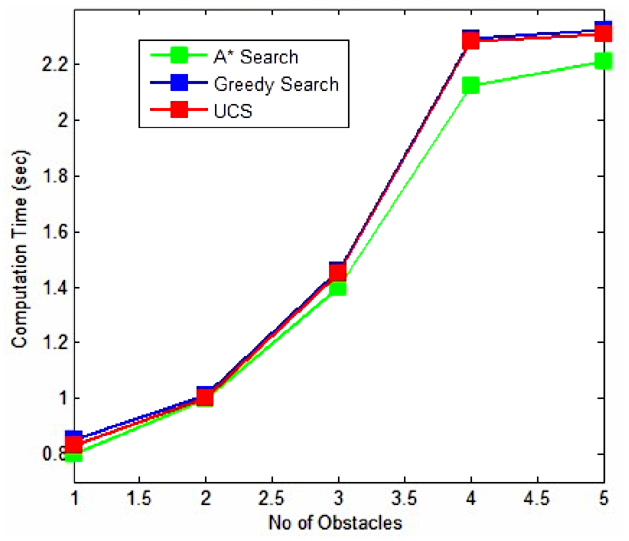


Fig. 11. Graph showing the performance of path planning algorithms.

5. Conclusion

Execution time and path length both factors are very important in real time implementation. All the three path planning algorithms used in research are evaluated on the basis of these two factors. After evaluation it is concluded that A* algorithm is the one that provides optimal paths for all the data set used in experimentation because it always provide shortest length path for robot to follow in least execution time as compared to UCS and Greedy search algorithms that is why it is best suited for this particular application. The robot then followed this path to arrive at the required goal.

Future work may include D* algorithm implementation for dynamic environments [21]. Structured or static environment is assumed in research work i.e. the obstacles are fixed. However, when there are moving obstacles in the environment i.e. in case of dynamic environment, re-planning of path after specific intervals is required so as to know if any change occurred in the environment. D* algorithm which is an extension of A*algorithm can be implemented in that scenario.

References

[1] N. Padoy, T. Blum, A. Ahmadi, H. Feußner, M.O. Berger and N. Navab, J. Med. Image Analysis **16** (2010) 632.

[2] M. V. Kreveld, M. de Berg and M. Overmars, Computational Geometry: Algorithms and Applications, Springer, Berlin (1997).

[3] D. Dolgov, S. Thrun, M. Montemerlo and J. Diebel, Springer Tracts in Advanced Robotics **54** (2009) 55.

[4] J.Reif, Complexity of the Mover's Problem and Generalizations, Proc. of 20th Symp. on the Foundations of Computer Science (1979) pp. 421–427.

[5] J. Bruce, T. Balch and Manuela Veloso, Fast and Inexpensive Color Image Segmentation for Interactive Robots, Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Takamatsu (2000) pp. 2061-2066.

[6] R. T. McKeon, M. Krishnan and M. Paulik, Obstacle Recognition Using Region- Based Color Segmentation Techniques for Mobile Robot Navigation, Proc. SPIE 6384, Intelligent Robots and Computer Vision (2006) 63840R.

[7] M. Agrawal, A. Sundaresan, M.R. Blas and K. Konolige, Fast Color/ Texture Segmentation for Outdoor Robots, IEEE/RSJ Inter. Conf. on Intelligent Robots and Systems (2008) pp. 22-26.

[8] A. Mishra, Y. Aloimonos, and C. Fermuller, Active Segmentation for Robotics, IEEE/RSJ Inter. Conf. on Intelligent Robots and Systems (2009) pp. 3133-3139.

[9] C. Harris and M. Stephens, A Combined Corner and Edge Detector, 4th ALVEY Vision Conf.(1988) pp. 147-151.

[10] J.M.B. Susan and S.M. Smith, Inter. J. of Comp. Vision **23** (1997) 45.

[11] X. C. He and N. H. C. Yung, Optical Engg.**47** (2008) 5.

[12] A. Rattarangsi and R. T. Chin, IEEE Trans. Pattern Anal. Mach. Intell. **14** (1992) 430.

[13] J. Canny, Artificial Intelligence **37** (1988) 203.

[14] S.R. Lindemann and S.M. LaValle, Incremental Low-Discrepancy Lattice Methods for Motion Planning, IEEE Inter. Conf. on Robotics and Automation (2003) pp. 2920-2927.

[15] M.H. Overmars and E. Welzl, New Methods for Computing Visibility Graphs, 4th Annual Symp.on Computational Geometry, New York, NY, USA (1988) pp. 164-171.

[16] S. K. Ghosh and D.M. Mount, SIAM Journal on Computing **20** (1991) 888.

[17] H. Choset, Sensor Based Motion Planning: The Hierarchical Generalized Voronoi Graph, Ph.D Thesis, California Institute of Technology (1996).

[18] F.P. Preparata and M.I. Shamos, Computational Geometry: An Introduction, Springer-Verlag (1985).

[19] A.A. Rizzi, P. Atkar, E. U. Acar, H. Choset and D. Hull, Inter. J. of Robotics Res. **21** (2002) 331.

[20] P. E. Hart, N. J. Nilsson and B. Raphael, SIGART Bulletin **37** (1972) 28.

[21] A. Stentz, The focused D* Algorithm for Real-time Re-planning, 14th Inter. Joint Conf. on Artificial intelligence, San Francisco, CA, USA (1995) pp. 1652-1659.