Scientific Journal of Impact Factor (SJIF): 4.72

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 4, Issue 8, August -2017

Comparative Analysis on Path Planning Algorithm's on Mobile Robot in Image Processing

Rama kanta choudhury[1], Chandra kanta Samal[2]

[1] *Computer science, Kalinga University, New Raipur, Chatisgarh, India,*
[2] *Computer Science, ANDC,Delhi University,New Delhi, India*

Abstract:- *This paper presents an analysis on different path planning algorithms for mobile robot in static environment. It investigates three well known path panning algorithms and compared their performance on the basis of their working parameters and the computation time to find the solution. It also finds the shortest distance for the static environment and compared the computational distance from start to goal point at different levels for all the three algorithms. It has been tested at different shape of the image and computational time and length for all the three algorithms and verified. Image processing places an important role in the field of mobile robot navigation. There are different methods are used for navigation. In this paper, we reviewed around 23 papers are briefly described the amount of existence of the work for each motion planning approach. This paper includes a comparative study using C++ based approach on mobile robot with respect to image processing. Showing how the behavior and activity of the research paper has taken a change towards the navigation processes which controls navigation using NFT , A*and D*Lite algorithms. It has been seen that many times the researchers use image processing technique for obstacle avoidance or it use singles from gesture control or it takes the help of smart phone with sensor to find the obstacle to reach the destination. Finally some open areas and challenging topics are discussed. There are many different algorithms developed for the path planning by researchers in the field of robotics, water based resources, navigation and internet based games. The algorithms used in this field is A*, D* Lite and NFT (neighbour finding technique). Here we compared three algorithms and established a comparison path based approach to differentiate both the algorithms. We used a grid search technique using A* algorithm and quad tree approach using neighbour finding technique algorithm and also we used a DDA algorithm to optimize the NFT based path planning approach and compared.*

Keywords: *Path Planning, Algorithms, Grid search, Quad Tree, NFT (Neighbour finding technique), D*Lite, Digital Differential Analyzer (DDA).*

1. INTRODUCTION

The field shortest path planning is closely related to robotics where as the algorithms used here applied in different fields as per the requirement. Planning mobile robot is a challenging task for mobile robot navigation and also researchers provided different approaches for the shortest path planning strategies in the field of mobile robot games and many more applications. Each plan has its own advantage and disadvantage the basic goal is to reach the robot from source to destination. Here we used grid search and quad tree based path planning method for path planning with the help of A* algorithm[3,16] , Neighbour finding method[6,7] and D* Lite[8] to verify the difference in their approach .Here we have presented a real image presented in the form of square picture shape and represented at different part of the block as shown in the fig1 (a, b, c).The boundary is taken as (80.80) and (400,500).In this paper we planned for robots for different algorithms. First we represent the image as a square of different size then embedded these different size images in the (80.80) and (400,500) screen size. After the design is proposed all used algorithms introduced and applied on the image to find the shortest path. These all are it are simulated and the result is compared.

II. THE APPROCH

The quadtree initially come up with [4] , It is an approach where the complete tree is represented as a square and the square is divided in to four sub branches of equal size ad shown in fig 1(a) as row1 and coum1(R1C1) and row1colum2(R1C2) and also the tree form in a said form that each node has four different quadrants as shown in fig 1(b,c) While dividing the [6] image into four quadrants each node has four sons each node represented with(R1C1,R1C2,R2C1,R2C2).

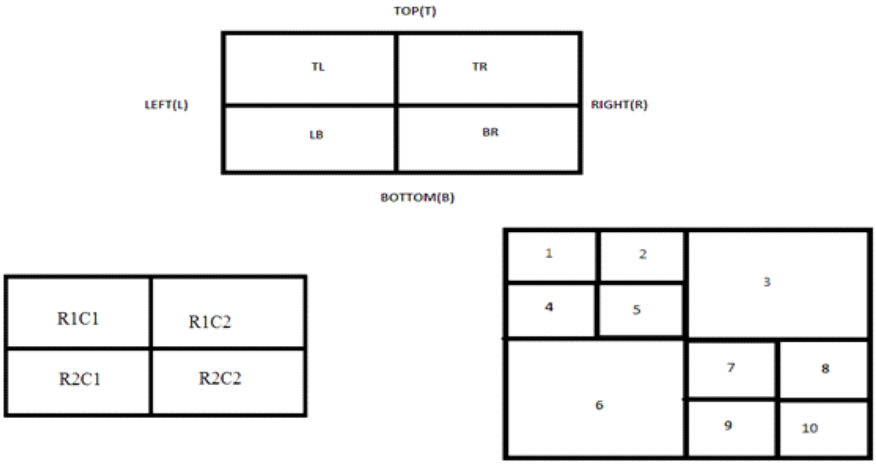

Figure 1(a) Representation of image with notation Figure 1 (b) Location of the position of an image Figure1 (c) Numbering the image as per the quadtree.

Here the quadtree is a well organized composite configuration which helps to represent the image in a batter way Assumption of existence of link in not permissible here the extra space that require will be reduced and optimized path will be find. In diagonal direction the adjacent nodes is taken into consideration rather than horizontal and vertical direction. We validate the diagonally adjacent nodes sequentially with related to previous node, once this approach is finished we get the source of shortest path shortest path is completed then final path is drawn [1,2]. In this quad tree approach the image is first imposed in the square and is checked weather the image is completely occupied the square or not, if not then the square is further sub divided in to sub squares until unless, the image is completely comes in side a square, once it comes completely inside a square then it treated as "BLACK" otherwise it is treated a "GRAY". The black and white nodes are treated as nodes. Where as the "GRAY" node is further subdivided into sub squares. An image with obstacle as shown in Fig 2 (a) and Fig2 (b) shows its quad tree and the image is imbedded in a black having notations and with respect to the R1C1,R1C2,R2C1and R2C2 the tree and its quadtree image related to the location of the image is marked in the tree. The black is marked as obstacles [3, 5].

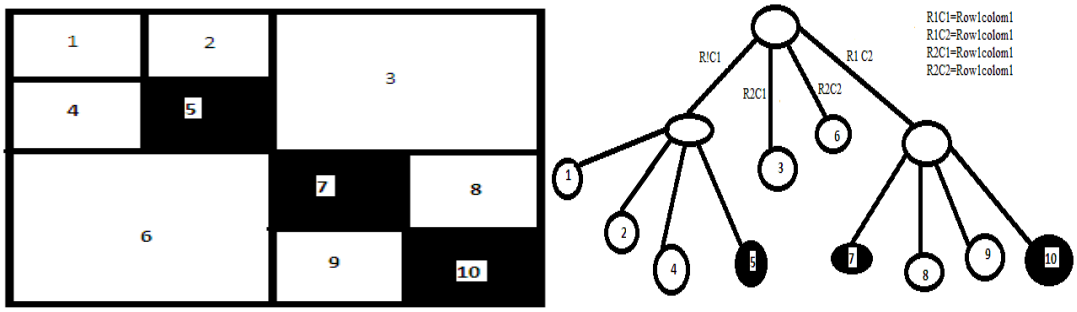

Figure 2(a) complete region with obstacle ***Figure 2 (b) Quad tree representation.***

III. SEARCH ALGORITHUM

Here we assume that each node is stored with six fields , out of which five fields main field and having its four sub fields called as braches. If M will be the main node and Q is the branch, it will be treated as MAIN (M) and SUBFIELD(M,Q). We can treat relative branches to its main by using subfield as SUBFIELD (M) .having value of Q if SUBFIELD (Main(M),Q)=M, the field are node type, specifies inside the branch of the image which represent "WHITE" if block contain no image or "BLACK" contains only unit number if having image ,and "GRAY" zero or one. BLACK and WHITE are

represented as branch nodes where as GRAY are non terminal nodes. Let the four sides of a node is called as R1C1, R1C2,R2C1, R2C2 shown in Fig2 (a, b). We use Some anticipation and functions involving certain quadrants and boundaries ADJECENCY(R,S) it is safe only if the branches S is adjacent to boundary R of the node block that is Adjacency (L,BL) is safe. REFLECTOR(R,S) represent as the SONTYPE having block of equal size that is the Nearest side of R of the block having SONTYPE,REFLECTOR(T,BL)=TL,SAMEBOUNDRY(R1,R2) and representing block R1 and R2 are not adjacent blocks.

Table 1 Adjecency ,Table 2 Mirror ,Table 3 opposite block ,Table 4 Same boundary

ADJECENCY	TL	TR	BL	BR
T(TOP)	T	T	F	F
B(BUTTOM)	F	T	F	T
L(Left)	F	F	T	T
R(right)	T	F	T	F

TABLE 1 ADJECENCY

MIRROR	TL	TR	BL	BR
T(TOP)	BL	BR	BR	BL
B(BUTTOM)	TR	TL	F	T
L(Left)	BL	BR	TL	TR
R(right)	TR	TL	BR	BL

TABLE 2 REFLECTOR

OPOSITE BLOCK	BLOCK
TL	BR
TR	BL
BL	NE
BR	TL

TABLE 3 OPPOSITEBLOCK

BLOCK1 TO BLOCK2	T L	TR	BL	BR
TL	NO	T	L	NO
TR	T	NO	NO	R
BL	L	NO	NO	B
BR	NO	R	B	NO

TABLE 4 SAME BOUNDRY

IV. SHORTEST PATH PLANNING APPROCH USING QUADTREE

Here the algorithm is used for the path planning for mobile robot based on the quad tree representation for immediate use of the robot .If there is a big free space then the complete area is taken as the free space and is represented in a square shaped block as shown in Fig. 1,and the free space is represented as Top (T),Left(L),Right(R),Bottom(B).The square is represented as TL=Top Left , TR = Top Right , LB=Left Bottom , BR = Bottom Right. The static image is represented in black colour squares which are having numbered as 5,7,10. In quadtree approach we first divide the complete square to four quadrants and is named as shown in Fig Fig2(a), after naming four quadrants and four quadrants are forming four branches of a tree and we divide the quadrant which is having the image in black colour as TL (Top Left) quadrant and BR(bottom Right) quadrant to equal shaped 4 quadrants .Dividing both TL, BR we start numbering the quadrants horizontally as shown in the table1,2,3,4.While dividing the quadrant it should checked weather the image is completely filled or not otherwise the quadrant is again divided in to four quadrants till the image is filled with image in black colour[5].The complete tree as shown in Fig2 (b).

4.1 The A* Algorithm

Implementation using A* algorithm is very easy and also very fast when it comes to practical purpose. In has an limitation is it uses neighbours so that it creates more such sub optimal paths which helps to reach the destination with zig-zag manner. Here in A* representation. The f'(n) = g(n)+h'(n) g(n) is the total distance it has taken to get from the starting position to the current location, h'(n) is the estimated distance from the current position to the goal destination/state. A heuristic function is used to create this estimate on how far away it will take to reach the goal state. f'(n) is the sum of g(n) and h'(n). This is the current estimated shortest path. f(n) is the true shortest path which is not discovered until the A* algorithm is finished.

4.1.2 Algorithm as follows

Establish a search graph G, where start node is P_0,Put P_0 in the list SAFE. Generate a list which is initially empty. If SAFE is empty, then treat it as unsafe. Select the first node on SAFE and remove it from SAFE, AND put is on CLOSED, and name it as node P.
If P is the goal node terminates at goal, obtain a path from P to P_o, establish step 7.

- Expand node P, establish the set Z of its successors that are not already ancestors of n in G. Install these members of Z as successors of P in G.
- Establish a pointer to n from each of those members of Z that were not already in G (i.e., not already on either SAFE or CLOSED). Add these members of Z to SAFE. For each member, z, of Z that was already on SAFE or CLOSED, redirect its pointer to P if the best path to m found so far is through P. For each member of Z already on CLOSED, redirect the pointers of each of its descendants in G so that they point backward along the best paths found so far to these descendants.
- Reorder the list SAFE in order of increasing f values. (Ties among minimal f values are resolved in favor of the deepest node in the search tree.)
  Go to Step 3.

4.2. The Neighbour Finding Algorithm
In this algorithm we proposed four different functions followed by SD (M,I) is same if the quadrant I is just beside of boundary M of the node block. JUST OPPOSITE (M,I) approaches to SONTYPE of the same block that is adjacent to boundary B of the node block. COMMONSIDE (S_1 and S_2) indicates the same side boundaries. The opposite quadrant do not share block boundary with the quadrant that is OPPOSITEQUAD(S).If S_1 and S_2 are not adjacent brothers then they share same side which is undefined. This S_1 and S_2 represent adjacent brother quadrants.
The steps of neighbour finding algorithm can be listed as follows:

- A node is given to specific block of the image.
- Here each node has its own horizontal or vertical direction.
- We search for SD or JUSTOPPOSITE or COMMON_FOR_BOTH or OPPOSITE_QUAD .
- After this it is followed by certain functions which are
- Identical_ adj_ neighbor (K, L) that is It locate an equal sized neighbour of node “P” in the Y direction or X direction.(Horizontal X and Vertical Y).
- Identical_corner_neighbour which identifies same size neighbour of node K in the direction of O.
- Getidentical_adj_neighbor (K, L) to identify a neighbour of node K in X or Y direction of L. If it is not available then return back. Get identical_adj_neighbor (K, L, E, F) which represent return of ‘S’ the neighbour of node K in X or Y direction in L and F represents the level of the tree at which node K is normally found and the tree at which node S is ultimately found. If such node does not exist then return to NULL.

4.3. D* Lite Algorithm.
In this algorithm the robot Main ( ) need to move the robot along the path determined by calculate path ( ) ,Main could calculate the priorities of the vertices in the priories of the vertices in the priority queue every time the robot notice a change in the edges costs after it has moved.

- First it will set start and goal point.
- From the starting point the robot will search for eight adjacent neighbors and continuously he will calculate the cost of the nodes, the small node cost will put in a list .
- The neighbor may be ≥ 8.
- It will continue as f = shortest to neighbor+ the distance to neighbor to goal.
- It will continue till the current node = node ± goal.
- Stops.
- Finally it finds the path.
  Compute shortest path ( ) of D* Lite shares many properties like it shares each vertex at most twice until it returns. The result shoes that Compute Shortest path( ) of D* Lite is terminates and correct.

V. SIMULATION RESULT

The comparison of A*, NFT,D* Lite and using DDA as shown in figure Fig 3(a, b, c) with difference in their time and distance graph plotted Fig 4 (a , b) .

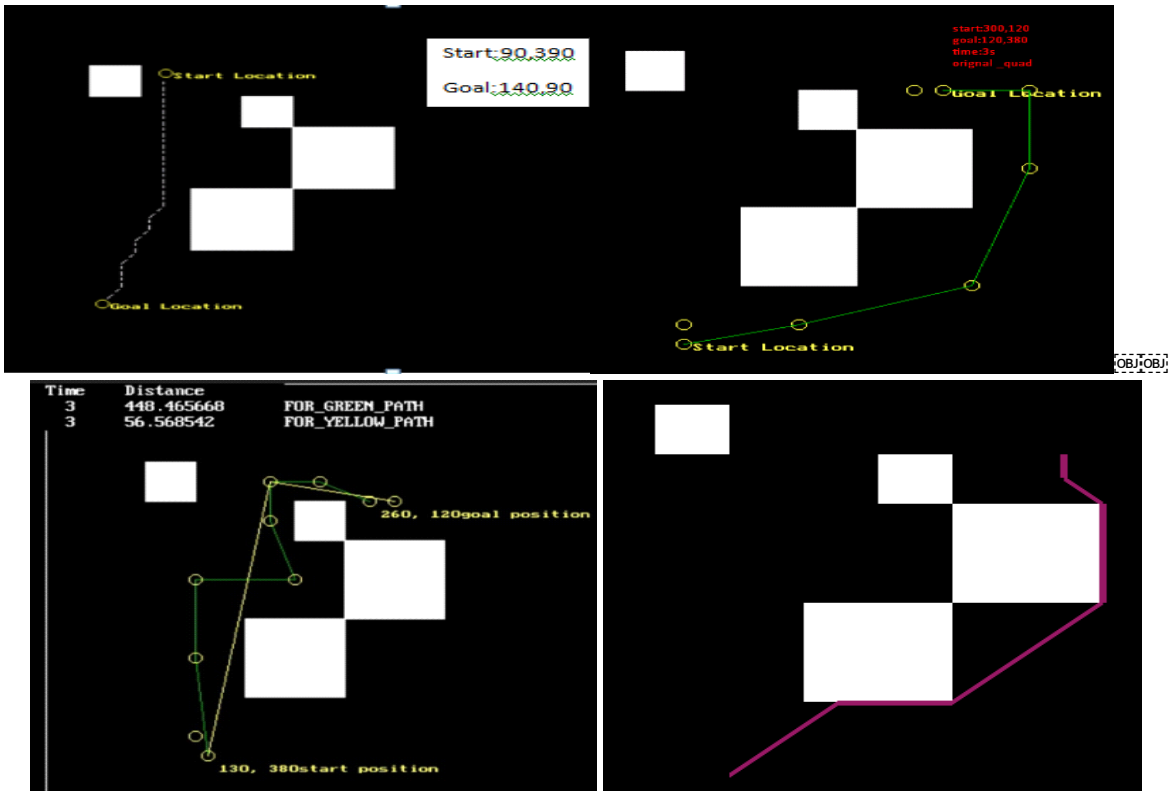

Figure3(a) A Grid search figure 3(b) NFT with quadtree Figure 3(c) DDA with NFT*

	START POINT	GOAL POINT	GRID SEARCH WITH A* ALGORITM	QUADTREE APPROCHED NFT ALGORITM	D* Lite ALGORITHM
1	140,90	330,330	1	3	3.195
2	90,390	140,90	2	3	0.102
3	130,130	330,330	2	4	3.385
4	90,380	240,90	1	3	2.912
5	150,150	350,350	1	2	2.624
6	90,390	130,130	1	3	0
7	240,90	230,230	1	3	0.601
8	130,130	130,380	1	3	0
9	150,150	350,350	2	3	2.62
10	120,350	260,100	2	3	2.11
11	80,350	260,80	2	3	2.524
12	260,120	220,380	2	3	2.386
13	160,160	220,380	1	3	0.608
14	300,120	120,380	1	3	2.555
15	180,180	300,300	2	2	2.442

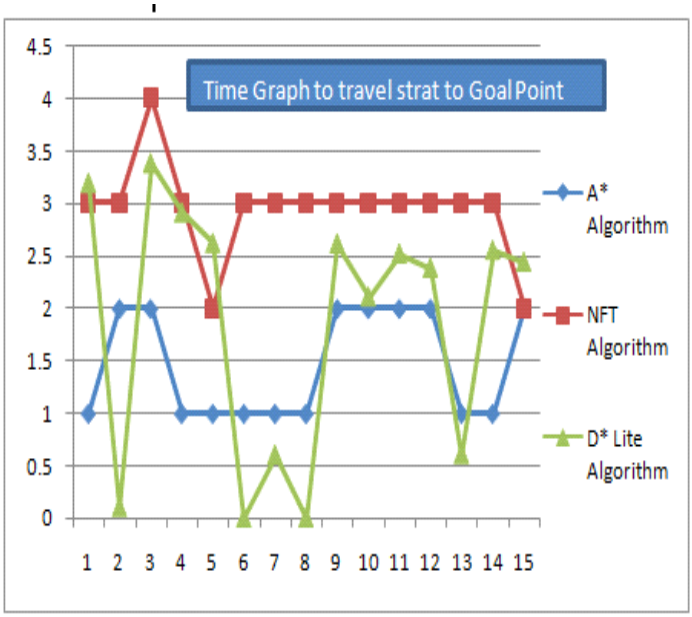

Figure 4 (a) The Time and distance result graph for A, NFT ,D* LIte*

	START POINT	GOAL POINT	GRID SEARCH WITH A* ALGORITM	QUADTREE APPROCHED NFT ALGORITM	D*Lite ALGORITHM
1	140,90	330,330	800.42043	437.893219	829.877
2	90,390	140,90	299.404301	307.263092	356.274
3	130,130	330,330	559.010752	444.127319	695.62
4	90,380	240,90	343.010752	430.009308	404.637
5	150,150	350,350	544.010752	505.800900	706.836
6	90,390	130,130	250.803226	264.72821	288.765
7	240,90	230,230	220.802866	205.316986	220.117
8	130,130	130,380	240	251.209427	316.207
9	150,150	350,350	544.010752	448.692456	706.386
10	120,350	260,100	423.409677	413.245544	502.471
11	80,350	260,80	338.611828	428.589508	467.419
12	260,120	220,380	686.005376	448.46568	426.66
13	160,160	220,380	686.005376	698.46345	415.094
14	300,120	120,380	483.611828	485.707672	639.2
15	180,180	300,300	556.212903	419.864594	782.754

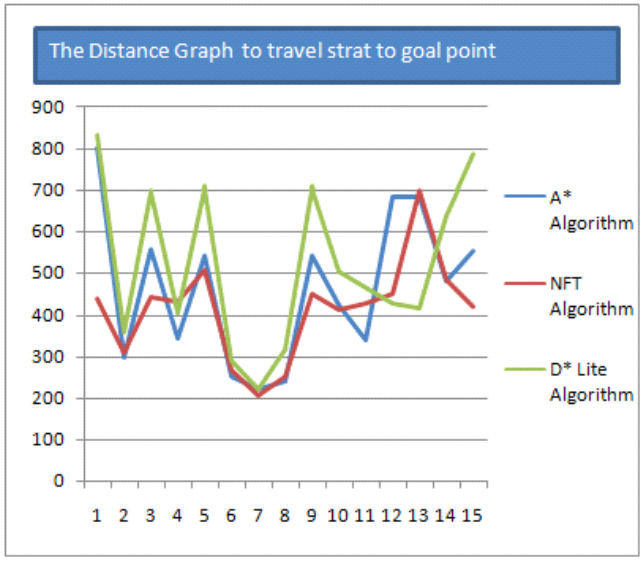

Figure 4 (b) The Time and distance result graph for A*, NFT ,D* LIte .

	START POINT	GOAL POINT	GRID SEARCH WITH A* ALGORITM WITH DDA	QUAD TREE APPROCHED NFT AND DDA (TIME)	D* Lite ALGORITHM WITH QUADTREE AND DDA
1	140,90	330,330	1	3	0.341
2	90,390	140,90	2	3	0.024
3	130,130	330,330	2	4	0.028
4	90,380	240,90	1	3	0.021
5	150,150	350,350	1	2	0.019
6	90,390	130,130	1	3	0.005
7	240,90	230,230	1	3	0.019
8	130,130	130,380	1	3	0.001
9	150,150	350,350	2	3	0.022
10	120,350	260,100	2	3	0.015
11	80,350	260,80	2	3	0.021
12	260,120	220,380	2	3	0.022
13	160,160	220,380	1	3	0.016
14	300,120	120,380	1	3	0.016
15	180,180	300,300	2	2	0.025

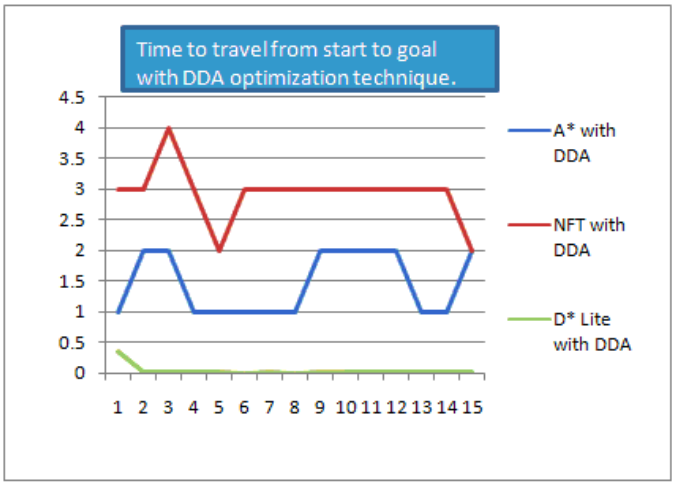

Figure5(a) The distance graph for A*, NFT, D* LIte and DDA optimization.

	START POINT	GOAL POINT	GRID SEARCH WITH A* ALGORITM	QUAD TREE APPROCHED NFT AND DDA	D*Lite ALGORITHM WITH QUADTREE AND DDA
1	140,90	330,330	800.42043	279.216736	534.7
2	90,390	140,90	299.404301	191.049728	258.8
3	130,130	330,330	559.010752	416.867828	391.45
4	90,380	240,90	343.010752	288.814484	285.8
5	150,150	350,350	544.010752	388.623457	434.776
6	90,390	130,130	250.803226	150.332962	288.765
7	240,90	230,230	220.802866	258.069763	220.117
8	130,130	130,380	240	220.227158	312.112
9	150,150	350,350	544.010752	441.413818	434.776
10	120,350	260,100	423.409677	296.619019	412.723
11	80,350	260,80	338.611828	309.756104	368.324
12	260,120	220,380	686.005376	356.568542	345.236
13	160,160	220,380	686.005376	448.56235	372.428
14	300,120	120,380	483.611828	400.081573	405.324
15	180,180	300,300	556.212903	116.619041	419.788

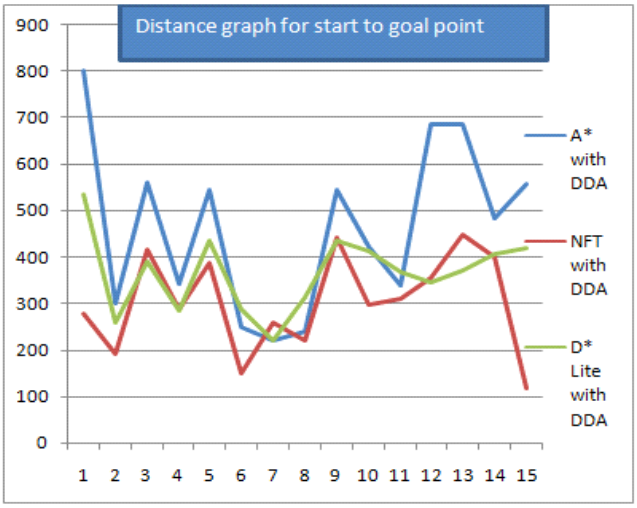

Figure 5(b) The time graph for A*, NFT , D*LIte and DDA optimization.

Figure 6 (a) Grid search using A* Figure (b) NFT search using Quad Figure 6 (c) D* lite

The difference of the time taken by A* algorithm using grid search and NGT (Neighbour Finding Technique) using quad tree , D*Lite using DDA optimization as shown in the Fig4(a) and Fig 4(b) and also shown in the graphical form as shown in Fig 5(a) and Fig5(b).We have tested 50 start and goal points at different locations and noted down the time taken by the CPU ,the configuration of the CPU as Intel(R) core™,i3-3220 CPU with 3.3GHz with 2 GB and 32 bit operating system .In both the cases when the shape and size of the image changes it do not work as efficient as a square size image. It is found that the path following in not proper if the shape and size of the image is changed which is shown in Fig 6(a, b, c).The D*Lite works batter then A* and NFT while time, distance, optimization if taken in to account.

VI. CONCLUSION

This paper presents three different path planning algorithms which is tested on C++.The obstacle used here is square shape Image and it is found that both work in square shape perfectly where as image in circular or other shape it do not work efficiently as shown in fig 6(a, b).In case of different shape image the D* lite work perfectly if the shape and size of the image changes 6(c). Where as if the environment changed i.e. If the image is circle or rectangle the path fallow in zig –zag manner. The time and distance taken by NFT (Neighbour finding) is more than that of the A* algorithm where as in D* Lite it is very less as shown in fig 4(a , b),which is presented in 5(a, b). Where as in NFT the speed is slow and the accuracy is batter then grid search but as per D* Lite is concerned, it is much batter then A* and NFT. In the implemented software architecture the resource-providing plug-in do not use a standardized method. Such a standardized method for resource sharing should be provided to facilitate a more decentralized maintenance structure.

REFERENCES

[1] Gargantini, I " An effective way to represent quadtrees communications of the ACM,25", pp.905-10,1982.
[2] Kunio Aizawa, Koyo Motomura, Shintaro Kimura, Ryosuke Kandowaki and jia Fan "Constant Time neighbor Finding In quad Trees: An Experimental Result ISCCSP @))*,Malta12-14,March 2008,978-1-4244-1688-2/08 2008,IEEE.
[3] Finkel.R.A and j. l.Bentley, Quadtree :A data structure for retrieval on composite keys , Acta Informatica ,4,pp.1-9,1974.
[4] G.M. Hunter and K. Steiglitz, Operation of image using quad tree, IEEE Trans. Pattern Anal. Mach Intell.1, 1979, 145-153.
[5] H.Samet, Region representation, quad trees from binary arrays, computer graphics and image processing 13,1980,88-93.
[6] H.Samet, computing perimeters of an image s represented by quad tree, IEEE Trans, pattern Anal. Mach, Intel press.
[7] H. Samet. An algorithm converting from roster to quad tree. IEEE Transaction, Pattern Anal, Mach. Intel press 3,1981,93-95.
[8] Sven Koenig maxim Likhachev "D*Lite" American Association for Artificial Intelligence copyright 2002.www.aaai.org.
[9] Cheol-Taek Kim and Ju-Jang Lee "Mobile Robot Navigation using Multi-resolution Electrostatic Potential Field" 0-7803-9252-3/05/ ©2005 IEEE.
[10] Ronald C. Arkin, *Behavior-Based Robotics*, The MIT Press, 1998.
[11] Alberto Elfes, "Using Occupancy Grids for Mobile Robot Perception and Navigation," *IEEE. Computer*, vol. 6, pp. 46–57, June 1989.
[12] Nikos C. Tsourveloudis, Kimon P. Valavanis, and Timonthy Hebert, "Autonomous Vehicle Navigation Utilizing Electrostatic Potential Fields and Fuzzy Logic,"*IEEE. Trans. On. Robotics and Automation*, vol.17, pp. 490–497, August 2001.

[13] Kimon P. Valavanis, Timonthy Hebert, Ramesho Kolloru and Nikos C. Tsourveloudis, "Mobile Robot Navigation in 2-D Dynamic environments Using an Electrostatic Potential Field," *IEEE*. Trans. On. Systems, Man, Cybernetics. Part A, vol. 30, pp. 187–196, March 2000.
[14] J. Hutchinson, C. Koch, J. Lea, and C. Mead, "Computing motion using analog and binary resistive *networks," IEEE. Comput. Mag.*, vol. 21, pp. 52–63, 1988.
[15] L.Tarassenko and A. Blake, "Analogue computation of collision free paths," in *Proc*. IEEE Int. Conf. Robotics and Automation, pp. 540–545, 1991.
[16] Sebastian Thrun, *Robotic Mapping*: A Survey, School of Computer Science Carnegie Mellon University, February 2002.
[17] Sebastian Thrun, *Probabilistic Algorithms in Robotics*, School of Computer Science Carnegie Mellon University, April 2000.
[18] Raghvendra V.Cowlagi and PanagiotisTsiotras "Multi-resolution Path Planning: Theoretical Analysis Efficient Implementation, and Extensions to Dynamic Environments" School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332.
[19] Abrahamsanchezlopez, Rene Zapata and MariaA. Osariolama"Sampling based motion planning : A servey" computation y systems Vol.12 No,1,2008, pp5-24,Issn 1405-5546.
[20] Huosheng Hu, Michael Brady, Penelope Probert "Navigation and Control of a Mobile Robot among Moving Obstacles" CH3076-7/91/0000-0698$01 .OO 0 1991 IEEE.
[21] LeelaRani .P.Rajalakshmi.P "Clustering Gene Expression Data using Quad Tree based Expectation Maximization Approach" International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868, Foundation of Computer Science FCS, New York, USA, Volume 2– No.2, June 2012.
[22] N. Shobha Rani, "ArunGopi "A Quad Tree Based Binarization Approach to Improve quality of Degraded Document Images" International Journal of Computer Science Engineering (IJCSE), ISSN : 2319-7323, Vol. 3 No.01 Jan 2014 1.
[23] Rahul Kala,Dr. Anupam Shukla, Dr. RituTiwari ,Sourabh Rungta, R RJanghe "Mobile Robot Navigation Control in Moving Obstacle Environment using Genetic Algorithm, Artificial Neural Networks and A* Algorithm]" 2009 World Congress on Computer Science and Information Engineering,978-0-7695-3507-4/08©2008IEEE,DOI 10.1109/CSIE.2009.854.

