International Journal of Latest Trends in Engineering and Technology
Vol.(7)Issue(4), pp.027-38
DOI: http://dx.doi.org/10.21172/1.74.004
e-ISSN:2278-621X

COGNITIVE THINKING FOR MOBILE ROBOT IN PATH PLANNING

D. Ramesh[1] and Ch. Jayanth Babu[2]

Abstract—Today's mobile robots operate in natural industrial, commercial, or military environments and must interact with humans during the decision-making process. The natural environment is highly unstructured and often unknown, hence these robots must be able to process a large amount of information, and make planning and navigational decisions quickly. Autonomous mobile robot have additional processing requirements where they have to collect, plan, and execute their planned route as well as performing tasks relevant to their specific missions. The path planning task is usually one of many important tasks that these robots must accomplish. Autonomous mobile applications require efficient and adaptive path planning approaches. A number of heuristic algorithms exist for producing optimal traverses given changing arc costs. But, most of analytical algorithms are focused on the path finding procedure in a known environment and leave higher level functions, such as obstacle detection. The Cognitive Based Adaptive path planning (CBAPP) is an adaptive and cognitive based thinking system. And efficiency of the CBAPP was compared with analytical and a heuristic based algorithm results. Based on the simulation results, it was shown that CBAPP finds fast, efficient, and acceptable paths.
Index Terms— Autonomous mobile robot , Cognitive Based Adaptive path planning

I. INTRODUCTION

One of the greatest challenges for an autonomous robot is finding a path from one point to another point in any environment. So the path finding strategies are playing a major role in an autonomous system. Path finding strategies have the responsibility of finding a path from one coordinate to another coordinate in any environment by taking a starting point and a destination point. Then they find a series of points that together comprise a path to the destination. For this a set of pre-compiled instructions are employed into the system to guide the robot. And these instructions extract the information from the environment and returns the coordinates to which the robot have to move in an environment when we gives start and goal positions. So cognitive based adaptive path planning algorithm finds a path with using cognitive and adaptive based methodologies.

Despite today's powerful processing hardware and inexpensive memory and storage devices, the processing requirement of complex autonomous robots still demands high efficiency in algorithmic and software program execution. Besides, progressively little to scaled down versatile robots are fancied for observation, reconnaissance, and risky material recognitions for military and mechanical applications. The autonomous mobile robotic applications have limited power capacity as well as memory and processing resources. The efficient path finding algorithms are the key enablers of the autonomous robotic applications and they must be

[1] Department of Computer Science and Engineering, SREC- Warangal,506371,Telangana
[2] Department of Computer Science and Engineering, SREC- Warangal,506371,Telangana

integrated with a hierarchy of decision making software applications, from sensor manipulation layer to the concept of mission operation.

Although we do not understand how the human brain processes information, we can observe that cognitive path finding is very efficient. Humans have the ability to minimize the processing time for a task significantly when they have access to relevant information. Unlike analytical Approaches, humans can accelerate their information processing in a familiar environment. This adaptive path planning approach is present in humans and is performed effortlessly in our everyday tasks.

II. LITERATURE SURVEY

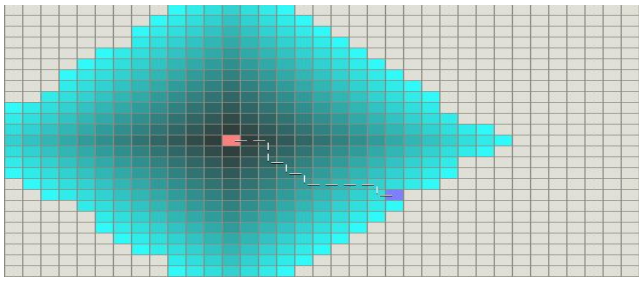
There are numerous Algorithms that are ordinarily known and utilized, running from simplex to complex, keeping in mind the end goal to have the capacity to tackle the way discovering issue. The least difficult methodology is to walk specifically towards the objective until met with any sort of impediments. At the point when met with any item, bearing will be changed and it can be passed by following around the snag. There additionally exist a few Algorithms that arrangement the entire way before moving anyplace. Best-first Algorithm extends hubs in light of a heuristic evaluation of the expense to the objective. Hubs, which are evaluated to give the best cost, are extended first.

The most generally utilized Algorithm is A* Algorithm [7], which is a blend of the Dijkstra Algorithm [4] and the best-first calculation. Dijkstra's Algorithm utilizes the uniform cost methodology to locate the ideal way while the A* Algorithm consolidates both methodologies in this way minimizing the aggregate way cost. What's more, D* Algorithm [3][17] it is utilized as a part of self-governing robot and in an obscure situation. This calculation upgrades their insight about the landscape when they move in it, and constantly perform re-arranging of the way.

The distinctive calculations work in various ways. The broadness first pursuit starts toward the begin hub, and after that inspects all hubs one stage away , then all hubs two stages away, then three stages, et cetera, until the objective hub is found. This calculation is ensured to locate a most brief way the length of all hubs have a uniform expense. The bi-directional expansiveness first inquiry is the place two broadness first quests are begun at the same time, one toward the begin and one at the objective, and they continue looking until there is a hub that both ventures have inspected. The last way is the blend of the way from the begin to the convergence hub, and the way from the objective to the crossing point hub.

2.1 Dijkstra’s algorithm

Dijkstra’s (named after its developer, E. Dijkstra [4]) algorithm looks at the neighbors of the node closest to the start, and sets or updates their distances from the start. The Dijkstra Algorithm extends the hub that is most remote from the begin hub, so it winds up "bumbling" into the objective hub. Much the same as the broadness first hunt; it is ensured to locate the most brief way. The profundity first hunt develops hubs until it either achieves the objective or a specific cut-off point, it then goes onto the following conceivable way.



2.1. Dijkstra algorithm searching area

The best-first pursuit calculation is a heuristic hunt calculation, implying that it can consider learning about the guide. It is like Dijkstra's calculation, however it goes to the hub nearest to the objective, instead of the hub furthest from the begin.

This Analytical Algorithm [15] is to a great degree wasteful since it tries all changes (of any length) of edges to process the briefest way. Most logical calculations are centered around the way discovering method in a known domain and leave larger amount capacities, for example, deterrent location, outside the extent of their destinations.

*2. 1 a). Genetic Algorithm***:**

Hereditary calculations [18] are a class of versatile techniques that can be utilized to take care of inquiry and streamlining issues including extensive hunt spaces. The inquiry is performed utilizing reproduced development. These calculations keep up and control "eras" of potential arrangements or "populaces". With every era, the best arrangements are hereditary controlled to frame the arrangement set for the accompanying era. As in nature, arrangements are joined (through hybrid) and/or experience irregular transformation.

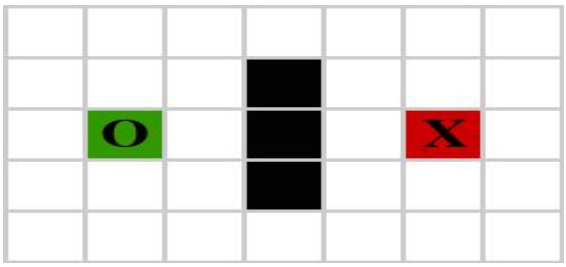
The path-planning component is divided in two sections: global path planning and local path planning. This two algorithm will work parallel to find final path.

2.1 b). Global vs. Local Path-Planning:

Global path planning [18] scan the complete environment up to the sensor area and prepares static path. Hear in this approach the algorithm generates a path from the start point to the destination point before the robot starts moving. On the other hand, local path planning [18] means that path planning is done while the robot is moving; in other words, in this the algorithm finds a new path in response to environmental changes. It assumes that there are no obstacles in the navigation area, the shortest path between the start point and the end point is a straight line.

2.2. A* Algorithm

A* algorithm [7] [16] is a space-search algorithm that can be used to find solutions too many problems. This Algorithm works same like the Dijkstra and best-first Algorithm just it offers qualities to the hubs in an unexpected way. Every hub's worth is the total of the genuine expense to that hub from the begin and the heuristic assessment of the rest of the expense from the hub to the objective. Thusly it consolidates the following of past length from Dijkstra's Algorithm with the heuristic assessment of the rest of the way from the best first inquiry. Truth be told, the Dijkstra inquiry is an A* look, where the heuristic is dependably 0. This calculation likewise makes the most productive utilization of the heuristic capacity, implying that no other Algorithm utilizing the same heuristic will extend less hubs and locate an ideal way.



2.2.1.Starting and Destination Points in A* Search Algorithm

A* Algorithm utilizes a beginning stage and a destination point to deliver the wanted way, the cell set apart with "O" is our beginning stage/hub and the cell set apart with "X" is the destination. White squares are walkable hubs and the dark ones are dividers, racks or whatever other impediments.

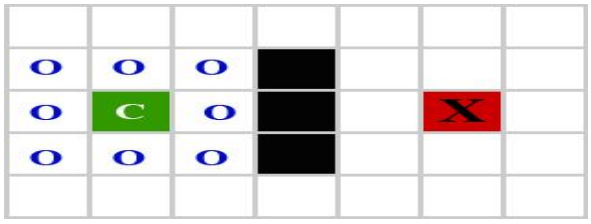
A* algorithm [11] starts to execute by looking at the starting node first and then expanding to the surrounding nodes. This operation proceeds until the destination hub is found. So as to know the hubs, which will be utilized as a part of hunt, A* calculation needs an approach to monitor the hubs. So the hubs to be inspected are held in a rundown, called Open List. The open line is utilized to store the hubs which have been recognized, yet their contiguousness has not put away. Since the operations of insertion and cancellation will happen much of the time, the open line takes a connected rundown usage At the starting it put the beginning hub to the Open List, and subsequent to looking at all of its encompassing hubs it will move it from the Open List and place in another rundown called the Closed List. Shut List holds the hubs that are gone by and there is no compelling reason to return to its individuals.

At the point when fabricating the Open List, calculation checks if the hub is walkable. On the off chance that the hub is not walkable, it is not added to the Open List. It will use the parent links to trace a path back to the starting node when we reach the destination. At this point we start over process. It now has to choose a new node to check from the Open List. At the first iteration it had only a single node in the Open List. It now has eight hubs in the Open List, and the hub which will first be assessed is dictated by allocating a score to every hub.

This f (n) , is the combination of two scores:

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

- g is the cost of getting from the start node to the current node i.e. the sum of all the values in the path between the start node to the current node.
- h stands for heuristic which is an estimated cost from the current node to the goal node (usually the straight line distance from current node to the goal node).
- f is the sum of *g* and *h* and is the best estimate of the cost of the path going through the current node. In essence the lower the value of f(n) the more efficient the path.



2.2.1 Neighboring cell to open list

1.1. D* Algorithm

D* algorithm which focused the cost updates to minimize state expansions and further reduced computational costs [3] [17] . D* algorithm works on least-cost paths between a start state and any number of goal states as the cost of arcs between states change. This algorithm can handle increasing or decreasing arc costs and dynamic start states. So this algorithm suited to solving the goal-directed mobile robot navigation problem, which entails a robot moving from some initial state to one of a number of goal states while updating its map information through an onboard sensor.

rhs(s) =0 if s = sgoal
min (all alternative paths) otherwise,

D* algorithm [3][17] updates the rhs-values of each state immediately affected by the changed arc costs and places those states that have been made inconsistent onto the queue. D* algorithm [17] which has been found to be slightly more efficient for some navigation tasks and easier to analyze.

D* algorithm [3][17] maintains an open list of states. The open list is used to propagate information about changes to the arc cost function and to calculate path costs to states in the space. And it maintains estimate of sum of arc cost from each state to destination node this estimate is equivalent to the optimal (minimal) cost from current state to destination state. In each step it compares the arc cost with minimum cost. Path costs less than or equal to are optimal, and those greater than may not be optimal. D*algorithm is a provably optimal and efficient path planning algorithm. This algorithm can handle the full spectrum of a priori map information, ranging from complete and accurate map information to the absence of map information.

The Heuristic algorithms [14] that planned an initial path with A* algorithm by using the prior map information, moved the robot along the path until it reached the goal. D* which focused the cost updates to minimize state expansions and further reduced computational costs. The algorithm used a heuristic function similar to A* to both propagate cost increases and focus cost reductions. A biasing function was used to compensate for robot motion between re-planning operations. Based on a qualitative estimate, 40% of the total area was searched by the algorithm to find a path.

III. PROBLEM DEFINITION

Most analytical algorithms are focused on the path finding procedure in a known environment and leave higher level functions, such as obstacle detection, outside the scope of their objectives.

A number of heuristic algorithms exist for producing optimal traverses given changing arc costs. The A* algorithm that planned an initial path by using the prior map information, moved the robot along the path until it reached the goal. D* which focused the cost updates to minimize state expansions and further reduced computational costs. The algorithm used a heuristic function similar to A* algorithm it focus on cost reductions. The searched area for finding a path in a 2 dimensional Cartesian coordinates is more. The efficiency of the algorithms is less. But complex autonomous robots requires high efficiency in algorithmic and software execution. So finding a path in an unknown environment with using cognitive and adaptive methodologies by reducing the searching area and increasing the efficiency is the problem.

IV. PROPOSED SYSTEM

The proposed system is cognitive based adaptive path planning it is based on cognitive and adaptive methodologies. It is on observed behaviors of the biological units and paid attention to the behavior of ignoring the irrelevant information from surroundings and trying to reach the target quickly.

Definition of Terms

Baseline definitions in this proposed system are used for the development and comparison of algorithms.

a). Adaptive Systems:

The general characteristics of adaptive systems. These elements included the retrieval operations used by an adaptive system to access previously stored information, and the mechanisms used to control the processing in the system. An adaptive methodology must have the following capabilities:

A mechanism by which incoming knowledge is stored in the system.Retrieval operations used to access previously stored information. A mechanism used to control information processing in the system.

b). Efficiency

The efficiency of the path finding algorithm is the ability of the algorithm to determine a collision free path in a dynamic environment with minimal resources. In surroundings where the robot's environment is described as a collection of two dimensional cells, the efficiency of a path is described as the ratio of the number of cells traveled in the final path divided by the total number of cells searched for that path.

$$efficiency = \frac{numberofcellstraveled}{totalnumberofcellssearched}$$

c). System overview

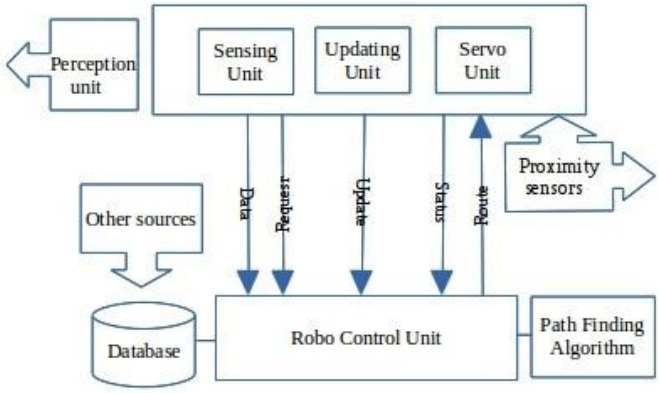
The Cognitive Based Adaptive Path Planning (CBAPP) is the path finding unit of the autonomous robotic system. The sub-systems of the robotic unit including the Robot Management System, the Scanning Module, the Updating Module, and the Servo Planning Module. The Robot Management System is the central program of the robot where the internal and external data traffic is managed.

The Servo Planning Module translates the selected path by CBAPPA into the control signals that control the drive mechanism. The Scanning Module is responsible for providing the sensory information to the Robot Management System. The Scanning Module receives the three dimensional data from the perception system and converts the format and the resolution of the data to match the format and the resolution of the CBAPPA.

Updating Module provides the true position of the robot in the global environment. This unit updates the position of the robot based on external references. The handshaking and the format of the data transfer between the Scanning Module and Robot Management System depends on the design of the Scanning Module. The implementation of this module is more application dependent than other modules.

The Servo Planning Module controls the motor drive mechanism and responds to the travel route data that the Robot Management System sends. A sensory signal from the short range sensors that enables the motor drive when the path is clear. The status of the path execution is communicated back to the Robot Management System during the robot's movement. The Global Database holds all the relevant information about the environment and obstacles.

System Architecture:



d). Environment setup

It is necessary to define the CBAPP's environment before the discussion of the individual algorithms. This environment is shared by all algorithms and is a medium where the outside information, such as the location of obstacles, are interpreted in a two dimensional system.

This environment is a multidimensional array which was developed in java applet. Total environment is 500×500 area which is occupied by squares each has 10mm size and total squares are 2500.

This environment should be realized by a global database in the design and implementation of the mobile robot using the CBAPPA approach. The coordinate frame is used to identify the heading of the robot in the global coordinate system. The Position algorithm uses this coordinate frame to determine 16 orientation vectors that are used to specify the direction that mobile robot is taking toward the next sub-goal.The cells which are filled with black are walls. We can set any environment with using wall button.

V. SYSTEM IMPLEMENTATION

The approach of the CBAPPA is based on cognitive based steps. Each step is implemented by a number of heuristic algorithms. Functionally, CBAPPA is broken down into two distinct stages:*The Primary Path. The Refined Path.*

Only a portion of the Refined Path that would fall within the robot's sensory area is passed down to other functions of the robot to be executed.

The primary path procedure is the first stage of the path finding.

For a human to plan a draft path to a destination, it could follow three simple steps:

(1) Determine the heading and the direction of the travel,

(2) Count for known obstacles based on his understanding of the environment,

(3) Plan a draft path before moving toward the destination. The Primary Path starts with a straight line approach from the starting position to the goal position. A predetermined number of numeric rings are formed around all obstacles in the environment. When the Primary Path strikes these rings, it uses them to go around the obstacle and back to the straight-line path on the other side of the obstacle. The Primary Path continues on this line toward the goal. If another ring is intercepted, the same procedure will be followed until all obstacles along the way are cleared and the final position is reached. These techniques are similar to the Means-end Analysis and Hill Climbing techniques. The Refined Path procedure is the second stage of the path finding

algorithm. Similar to the Primary Path routine, there are 3 cognitive based steps toward final path and a destination. These steps include moving along the primary path while setting intermediate goal states. The travel toward the final goal will be augmented by necessary adjustments, as the person confronts unexpected obstacles.

The Refined Path procedure uses the Primary Path as a guide to anticipate objects that will be encountered. The result of this procedure is a smooth, more efficient path that connects the starting position to the goal position. The Refined Path procedure uses techniques similar to the Means-end-Analysis techniques.

The Local Path algorithm is the final step of the Refined Path procedure. It produces the path that is executed by the motor control mechanism of the robot. This procedure selects a portion of the Refined Path that is immediate to the current position. This path must fall into the local sensing region of the robot. This security measure guarantees that the robot will always travel in a known area.

The CBAPP is controlled by a central program named Execution Sequencer (ES). The ES determines the flow of the execution of algorithms by calling the Primary and Refined Path routines. The ES starts with the Primary Path routine and repeats the Refined Path routine until the final goal is reached. The ES also orchestrates the data exchange with peripheral devices such as SCM, UPM, SPM, GED, and the user interface module by interfacing with RMS. The ES uses a multi dimensional array which is 50*50 rectangles to store the global information. And each rectangle is 10mm height and 10mm width.

This array is shared among all algorithms. Obstacle coordinates in the multidimensional array hold the numerical value. And holds the coordinates of obstacle positions in a separate array list. When we give obstacle positions then only it adds the coordinates into the array list. The servo module of the robot receives the output of the local path algorithm. This is the portion of the Refined Path, stored in path ( ) array that falls within the local sensory area of the robot. The robot moves along this path which is return by refined path. In the case of running into an object during the Local Path execution, the robot must stop, store the current position, and scan the environment for new obstacles.

According to the sensor information which comes under robot sensor area it finds the next line of sight coordinate on the primary path as the next intermediate goal. The Straight Path algorithm finds a straight route to the intermediate goal. The Find Path algorithm finds an obstacle free path, using the straight path as a guide, to the intermediate goal.

The Local Path algorithm selects a portion of the Path that falls within the Active Window and sends it to the Servo Planning module for execution. When the robot is traveling in an environment that has not been mapped previously, the Primary Path uses a straight path as a guide connecting the starting position to the goal position. This means that the robot does not have a global picture of the environment, and it is detecting obstacles as it travels toward the goal, relying only on its sensory system.

This path planning is more complex than determining a collision free path in an environment with stationary obstacles. This path finding is referred to as Blind Path Finding. Since the local sensory system is limited in range, the robot’s ability to determine the proper direction is limited when it is faced with a large obstacle that is larger than its local sensory range. Blind Path Finding occurs when there is no intelligence available about the obstacles and the robot is facing an obstacle that is larger than the active sensory window.

In this mode, the robot will choose a direction (toward the goal if one is available) and follow that path while keeping track of the distance and the cost of the travel, hoping to go around the obstacle. After exceeding a certain threshold, the robot will go back and try an alternative path. While traveling around the obstacle, the robot is learning about the obstacle and updates the database after each attempt. The ES activates the Primary Path routine after a failed attempt to start with a new draft path using new data that was learned from the last update.

CBAPP Algorithm:

A. *Finding the path*:

Step1: find a primary path from start node to destination node
Step2: find refined path in a primary path until destination node reach
Step3: update robot potion according to refined path
Step4: repeat to step2

b)Finding Primary path

Input: start point, destination point.
Output: primary path
Step1: take start node and destination node.
Step2: blindly increment the start node towards the destination node.
Step3: add each node to ArrayList.
Step4: repeat step2 until reach destination node.
Step5: return primary path.

c)*Refined path*:

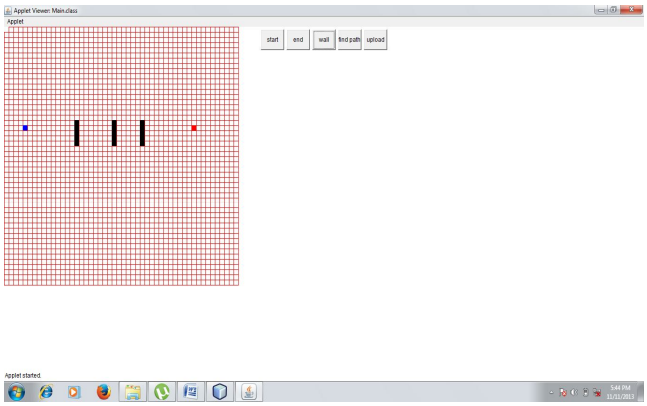
Input: start point, primary path.
Output: next node
Step1: take a portion of area according to primary path.
Step2: check for obstacle in that primary path which is in a portion of area. If obstacle is found in the primary path. Check right and left side of that node for obstacle. If obstacle is found both sides then move back and repeat step 2.
Else return which side has found no obstacle and repeat the step i until it reaches other side of the obstacle.
Return that node and update current node.Repeat step I until the selected portion is completed.
Step3: select next portion of the primary path repeat step2.
Step4: repeat step3 until destination found.

VI. RESULT ANALYSIS

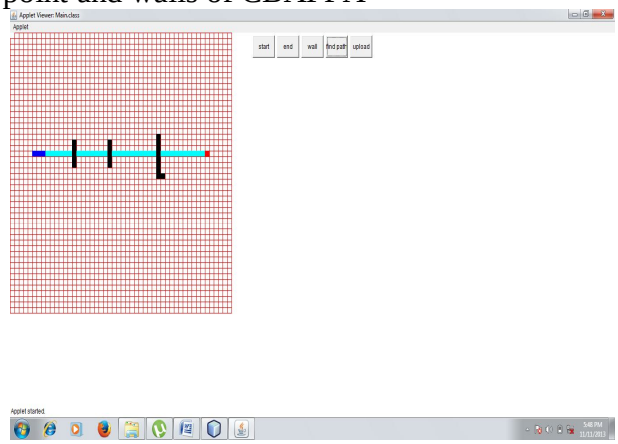
After developing a cognitive based adaptive path planning algorithm in a 50*50 multidimensional arrays. First it finds a draft path according to known information of starting and end position. After that it moves along the draft path by considering obstacles and according to the sensor information. So it is observed that the object finds a path in like a biological system how it finds a path in an unknown environment.

Output:

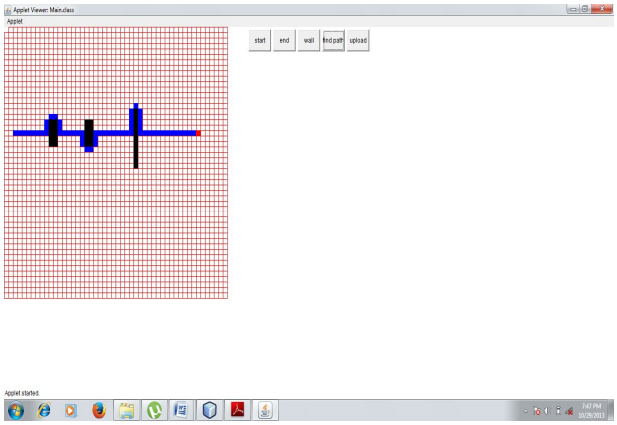
Final output in a java Applet.



5.1. Starting and ending point and walls of CBAPPA



5.2. Draft path of an algorithm



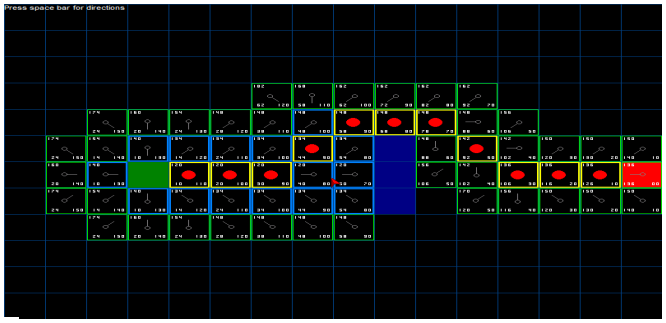
5. 3. Final output

EVALUATION:

It is evaluated by considering the output of both algorithms those are A* algorithm and Cognitive based adaptive path algorithm by giving same distance from starting point to ending point and developed in same environment. It found that searching area of proposed system is reduced and efficiency of proposed is increased when compare to A* algorithm.

A* algorithm:

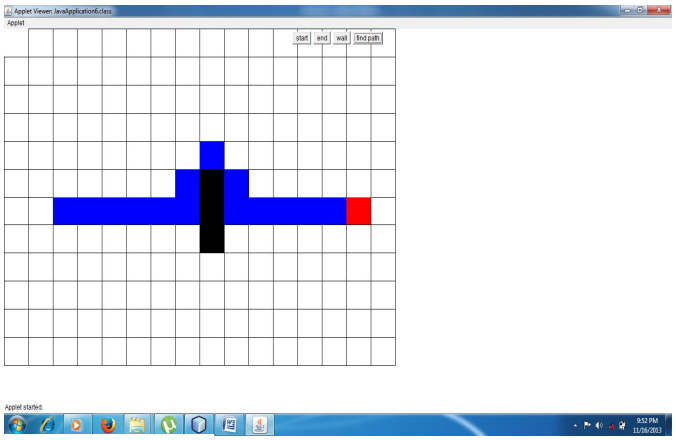
Searching area and efficiency is in a 192 cells environment that is 12*16 multidimensional arrays. For finding a path of length 11 cells by considering all possible ways from starting position to ending. It searches 64 cells in total of 192 cells



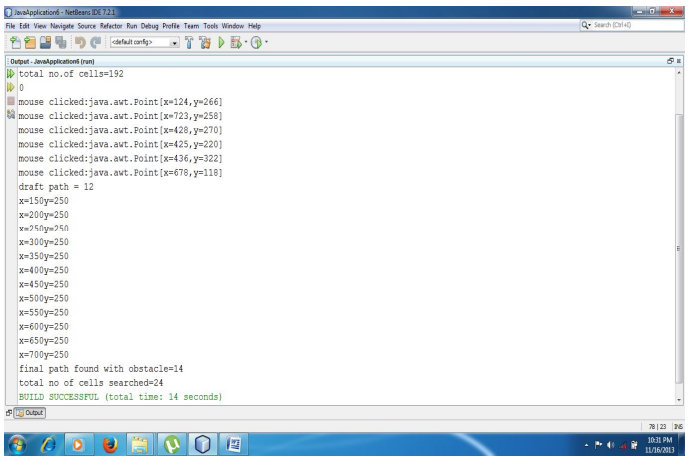
5.2 A* algorithm output

Cognitive based adaptive path planning algorithm:

Searching area and efficiency of CBAPPA in same environment. For finding a path of length 13 cells by neglecting all irrelevant information from starting position to ending. It searches 24 cells in total of 192 cells.



1. 3.output of CBAPPA for evaluation



5. 4. result of CBAPPA

VI. CONCLUSION

CBAPP completely fallow the adaptive methodology and cognitive based methodology. That is the system is equipped with a mechanism by which it can store information, and the system is capable of retrieving previously stored information, and the system is able to control the information processing in the system. And path finding efficiency exceeds the efficiency of analytical and heuristic based algorithm results. And searching area is decreased.

CBAPP finds path how a biological organ behaves when it is finding path. By taking decisions dynamically and store the information which used further and ignore irrelevant information and quickly process non optimum but efficient paths. So that CBAPPA is also without depending on any other it takes its decisions dynamically.

In this project CBAPPA was applied for only on static environment. And for static obstacles. So Future work will involve developing an algorithm with using cognitive and adaptive methodologies on dynamic environment and moving obstacles.

REFERENCES

[1] B. Krose, "Environment Representations for Cognitive Robot Companions", ERCM News, (53), 2003, pp. 29-30.

[2] Y. Goto and A. Stentz, " Mobile Robot Navigation: The CMU System", IEEE Expert, 2(4), 1987, pp. 44-55.

[3] A. Stenz and M. Hebert, "Navigation System for Goal Acquisition In Unknown Environments," Proceedings of IROS 95, USA, 1995, pp. 425-453.

[4] E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik. 1 (1959), S. 269–271

[5] A. Stevens and P. Coupe, "Distortions in Judged Spatial Relations," Cognitive Psychology, 10(4), 1978, pp. 526–550.

[6] T. McNamara, "Mental Representations of Spatial Relations", Cognitive Psychology, 18(1), 1986, pp. 87–121.

[7] Hart P. E., Nilsson N. J., Raphael B., A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems Science and Cybernetics SSC4 (2): pp. 100–107, 1968.

[8] J. Hopfield and D. Tank, "Neural Computation of Decisions in Optimization Problems", Biological Cybernetics, 52, 1985, pp. 141-152.

[9] O. Miglino, F. Menczer, and P. Bovet, "A Neuro-Ethological Approach for the TSP: Changing Metaphors in Connectionist Models", Journal of Biological Systems, 2(3), 1994, pp. 357-366.

[10] D. Durand, "Rationale, Goal, and Scope" Journal of Neural Engineering. Retrieved January 20, 2005, from:http://www.iop.org/EJ/journal/-page=about/1741-2552/1.

[11] Planar Map Pathfinding Based On The A* Algorithm Li Mingcui

[12] J. C. Latombe and J. Barraquand, "Robot Motion Planning: A Distributed Representation Approach.," International Journal of Robotics Research, 10(6), 1991, pp. 628-649.

[13] A. Razavian, "The Numerical Object Rings Path Planning Algorithm (NORPPA)," Proceedings of 35th IEEE Conference on Decision and Control, 1996, pp. 4406-4411.

[14] W. Hyun and H. Suh, "A Hierarchical Collision-Free Path Planning Algorithm for Robotics", Proceedings of the International Conference on Intelligent Robots and Systems, USA, 1995, pp. 488-995.

[15] V. Akman, Unobstraucted Shortest Path in Polyhedral Environments. New York: Springer-Verlag, 1987.

[16] J. Nilsson, Principles of Artificial Intelligence, Morgan Kaufmann, 1994.

[17] A. Stentz, "The Focussed D* Algorithm for Real-Time Replanning", Proceedings of International Joint Conference on Artificial Intelligence, 1995, pp. 1213-1221.

[18] Autonomous Local Path-Planning for a Mobile Robot Using a Genetic algorithm Kamran H. Sedighi, Kaveh Ashenayi, Theodore W. Manikas , Roger L. Wainwright, Heng-Ming Tai.