Mobile Robot Navigation Control in Moving Obstacle Environment using Genetic Algorithm, Artificial Neural Networks and A* Algorithm

Rahul Kala[1]
rahulkalaiiitm@yahoo.co.in

Dr. Anupam Shukla[2]
dranupamshukla@gmail.com

Dr. Ritu Tiwari[3]
rt_twr@yahoo.co.in

Sourabh Rungta[4]
sourabh@rungta.org

R R Janghel[5]
janghel1310@gmail.com

[1,2,3,5] *(Department of Information Technology)*
(Indian Institute of Information Technology and Management, Gwalior, MP, INDIA)

[4]*(Rungta College of Engineering and Technology, Bhilai, Chhattisgarh, INDIA)*

Citation: R. Kala, A. Shukla, R. Tiwari, S. Roongta, R. R. Janghel (2009) Mobile Robot Navigation Control in Moving Obstacle Environment using Genetic Algorithm, Artificial Neural Networks and A* Algorithm, *Proceedings of the IEEE World Congress on Computer Science and Information Engineering*, Los Angeles/Anaheim, USA, pp 705-713.

Final Version Available At: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5171088

Abstract

The pace of development and automation urge the need of robots controlling much of the work which used to be done mainly by humans. The modern technology has emphasized on the need to move a robot in an environment which is dynamically changing. An example of such an application may be the use of robots in industry to carry tools and other materials from one place to other. Since many robots would be working together, we need to ensure a collision free navigation plan for each of the robots. In this paper we find out the nearly most optimal path of the robot using Genetic, ANN and A algorithms at each instant of time of robot travel. It may be used by the industry to send robots for surveys, data acquisition, doing specific work etc. The collision free movement of robot in a moving obstacle environment can be used to move robot in a world of robots. Results show that all 3 algorithms are able to move the robot without any collisions.*

1. Introduction

Consider a situation where many robots/moving obstacles are together in a place, moving constantly just like humans move in a market. The problem is to make your robot move from the starting position to the final position [1, 20]. We need to optimize the path it travels. We also need to ensure that the robot does not collide with any of the other obstacles. We need to move the robot using this navigation plan.

The paper proposes the use of Genetic, Neural and A* algorithms to find out the most optimal path of the robot at every instant of time [2]. The robot tries to find a path that would most optimally take it to the goal position. Based on these results a single unit move is made. At the next unit of time again the algorithm is run to calculate the next move. Hence at every instant of time the algorithm runs and gives the next move of the robot. The robot physically moves according to these results and the move is completed. It is not ensured that every move will make the robot closer to the goal. But at the end, if there is a path possible to reach the goal at any instant of time, the robot reaches the goal. In case it is not possible to reach the goal at all, the robot becomes stationary.

Genetic Algorithm is the first algorithm implemented. This algorithm works by the principle of trying to find the best solution by combination of existing solutions. The second algorithm we have implemented is Artificial Neural Networks. This algorithm believes in learning from the past results. We have also implemented A* algorithm for the same problem. The A* algorithm tries to minimize the path travelled and the path which is left to be travelled. It hence tries to optimize the total path to be travelled by the robot. We take a heuristic function to predict the number of moves that are left in a particular position.

The elementary model of cognition [5] includes three main cycles. Among these, the 'sensing-action' cycle is most common for mobile robots. This cycle inputs the location of the obstacles and subsequently generates the control commands for the motors to set them in motion. The second cycle passes through perception and planning states of cognition, while the third includes all possible states including sensing, acquisition, perception, planning and action [6]. Sensing here is done by ultrasonic sensors/camera or by both. There are many algorithms for construction of the robot's world map [7]. The term Planning of Navigation [8] refers to the generation of sequences of action in order to reach a given goal state from a predefined starting state.

2. Motivation

The problem of robot navigation control, due to its applicability, is of a great interest. We have already seen good research in various modules. A lot of work exists to model the entire problem [1-7]. There exist good algorithms to scan the environment and represent all the obstacles in form of a grid [3]. Also various algorithms have been proposed to plan the movement of the robot using various conditions.

The whole problem till now has been seen under separate heads of planning navigation control of static environment and planning navigation control of dynamic environment. If we come to static environment, many algorithms have been implemented and results verified [8, 10, 11, 19]. In planning dynamic environment the steps are a little different, as the environment continuously changes.

We have various works of research in which people have tried to solve the navigation problem using genetic algorithm [8, 10, 11, 16]. The basic principles in all these have been to take a fixed solution length and find the solutions by using genetic operators. In this paper we propose a graphical node representation which will work for all sorts of highly chaotic conditions where the number of obstacles is very large.

Also similar work exists in neural network [6, 11, 15]. Here neural network has been applied mainly on static data. Here we have adopted a representation that minimizes the errors that might come due to neural noise or incomplete database. Some work also exists in A* algorithm [21]. Mainly people use Manhattan distance, or simple distance between current position and goal position. In this paper we have used a heuristic function that optimizes the path, at the same time resolves the conflicts when two paths may have same heuristic values by considering the rotational factor as well.

3. Simulation Model

In this section we will discuss the way we model the whole problem and also the way we would be applying all the three algorithms in this problem. The whole modeling includes modeling of the robot and environment, modeling of the algorithms and modeling of the results, so that they may be implemented.

3.1. General Assumptions

In order to visualize and implement the solution, we have made the following generalizations/assumptions [17]. Various models may be proposed to model the robot [12]. These have been made considering the practical implementation of the proposed solution.

- The entire space where the robot can move is a finite space. This space is divided in a grid of size M X N [3, 13]. This space is finite and can be simulated using Genetic, ANN and A* algorithm.
- Each obstacle as well as robot can make only a unit move in a unit time called the threshold time (ή) (see section 3.2).

3.2. Robotic Design and Assumptions

The robot we consider here consists of two wheels. Hence it can travel forward (both wheels rotating in same direction) or rotate clockwise or anticlockwise (both wheels rotating in opposite directions) [4]. For this problem we take the following movements of the robots as valid that can be performed in a minimum threshold time. The movements have been quantized for algorithmic purposes:

- Move forward (unit step)
- Move at an angle of 45 degrees forward (unit step) from the current direction
- Move clockwise/anti-clockwise (45 degrees)
- Move clockwise/anti-clockwise (90 degrees)

It is assumed that the robot only rotates/moves in angles of a multiple of 45 degrees. Hence we can only move the robot in the directions (by rotations and forward move) North, North-East, North-West, South, South-East, South-West as given in Figure 1.

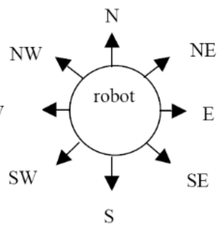

Figure 1: Various directions the robot can move in

For algorithmic purposes the directions have been denoted as N by 0, NE by 1, E by 2, SE by 3, S by 4 and so on. Hence if the robot is at direction 5, it has the following possible moves:

- No motion
- Change direction to 4, 3, 6 or 7
- Move forward in direction 5
- Move forward in direction 6 or 4

In this paper we assume that the information of every obstacle is available and is being constantly updated in the threshold time (ῄ) which is the unit time assumed by the algorithm.

3.3. Algorithmic Details

Another aspect of the algorithms used is their real time nature. If any Algorithm, due to excessive input or any other reason fails to produce a result in half the threshold time, then no move is made at all. This avoids the wrong calculations being made. As after this duration it is possible that the obstacles might have changed their location and the algorithm was using the old data.

4. CASE I: Genetic Algorithm

The algorithm used is the Genetic algorithm to find out the goal (final position) starting from the initial position. Genetic algorithm [14, 16] applied by us in this problem differs from the other genetic algorithms taking the fact that we have considered a graphical node as a chromosome, in place of taking a predetermined length bit sequence [8, 10, 11]. In this algorithm we use the following specifications for the functioning of the genetic algorithm.

4.1. Representation

The entire grid of MXN size is available. We know that the problem is to get the final path from the source (initial point) to the destination (final point). This consists of a series of points on the grid. Each point has three attributes associated with it. These are the x coordinate, y coordinate and the direction in which the robot is facing. Any point can thus be represented by (x,y,d)
where x=x coordinate
y=y coordinate
d=direction in which robot is facing (between 0 to 7)

The final solution generated is in the form
$(x_1,y_1,d_1)(x_2,y_2,d_2)(x_3,y_3,d_3)\ldots\ldots\ldots\ldots\ldots(x_n,y_n,d_n)$

Each solution is stored in the solution set. We have considered three types of solutions:

- **Full**: They are complete solution and fully connect source to destination
- **Left**: The start from the source but are unable to reach to the destination
- **Right**: They do not start from the source but reach the destination

For the representation of the problem, we take each point (x,y,d) as a node on a graph. Each node can be connected to other nodes if the transition from the first node to the other node is a valid move. This means (0,0,0) can be connected to (0,0,3) as we can turn from direction 0 to 3. But (0,0,0) cannot be connected to (1,1,0), as the move is not possible.

4.2. Evaluation of fitness

We measure the fitness of the chromosomes by the evaluation function. The lesser the value of the evaluation function, the more fit is the chromosome. The formula of the fitness function is given by the following:

- **Full Solution**: No of steps needed to traverse from source to destination.
- **Left Solution**: Number of steps needed to traverse from source to the last point traversed + (Square of the distance of this point to the final destination + R(n))
- **Right Solution**: (Square of the distance of source point to the first point traversed + R(n)) + Number of steps needed to traverse from this point to the final solution

Here R(n) = minimum rotation time required for the robot in its entire journey assuming no obstacles

If the direction is 2 (East), we have R(n) as the following (Refer Figure 2) Region 1: 0, Region 2: 2, Region 3: 1, Region 4: 1

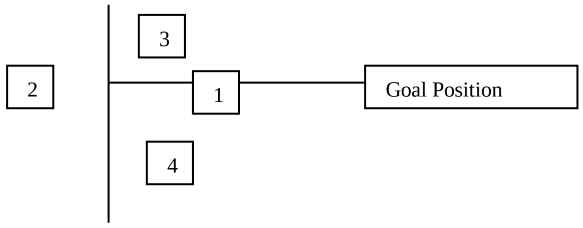

Figure 2: The values of R(n) at various places

4.3. Initial Solution

The algorithm starts by trying to find an initial solution. The layout of the initial solution is explained in figure 3. The initial solution is found out by applying the following strategies:

- **Find a straight path solution between source and destination**: From the source march straight ahead towards the destination. Whenever travelling at any point, there is a possibility of collision, take random number of moves. Now repeat this process of marching towards the destination.
- **Find random left solution between source and destination**: Start from the source. At any point take random moves at any direction. Repeat doing this.
- **Find random right solution between source and destination**: Start from the destination. At any point take random moves at any direction. Add every solution at the head of the final solutions sequence; so that it looks we approached the destination from a point. Repeat doing this
- **Find Full Solutions:** Once we have sufficient entries for the left and right solution, there are chances of getting more full solutions from these data. Search all the left and right solutions obtained so far. If you find any point common between any solution of the left and right solutions, mix them to get a full solution.

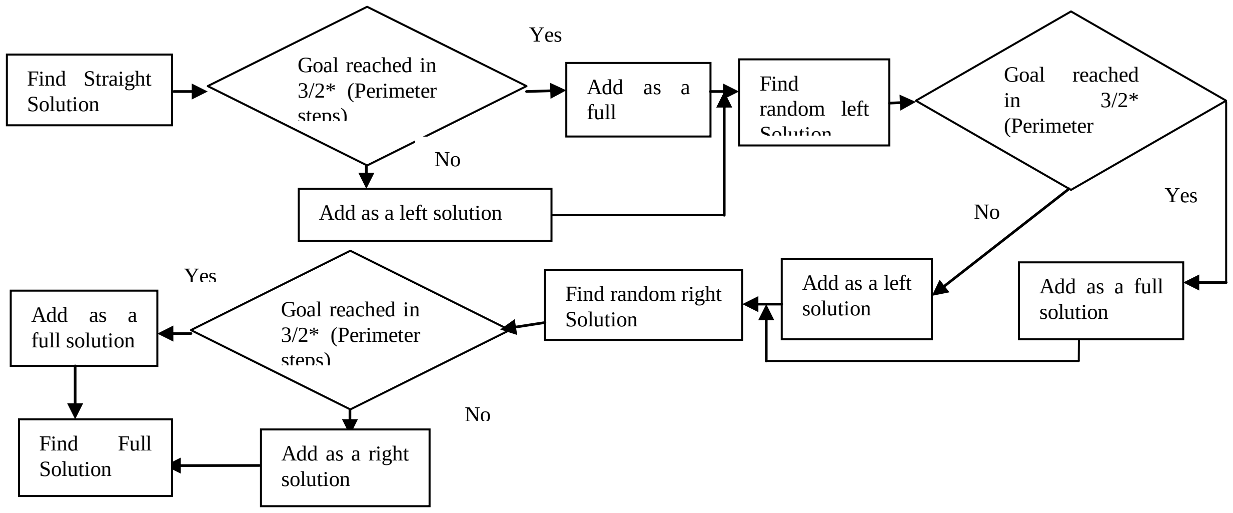

Figure 3: The Steps for generating initial solution for the robot

The R(n) term ensures that the number of rotations are minimum. As in the practical scenario, we need to avoid the rotations as much as possible. The lesser the number of rotations, the smoother the travel. Hence closer a point to the destination and traverses this in lesser steps, lesser will be the value of the function and better will be the solution

4.4. Crossover

This process means to mix two solutions already known, and to generate a solution from the mixture of the two solutions. The new solutions may be better solution than its parents. Hence two solutions are taken for this step. The crossover can occur between any of the following pairs of solutions

- Full Solution and Full Solution
- Left Solution and Full Solution
- Full Solution and Right Solution
- Left Solution and Right solution

In all of these we have two sequences of points (solutions). One is chosen from the left entities mentioned above and the other from the corresponding right entities of the chosen set above. The crossover process is similar to the one used in initial process

If the left sequence selected is
$(x_{11},y_{11},d_{11})(x_{12},y_{12},d_{12})$………..$(x_i,y_i,d_i)(x_{1i+1},y_{1i+1},d_{1i+1})$ ………………...(x_{1n},y_{1n},d_{1n})

and right sequence selected is
$(x_{21},y_{21},d_{21})(x_{22},y_{22},d_{22})$………$(x_i,y_i,d_i)(x_{2i+1},y_{2i+1},d_{2i+1})$ …………….(x_{2n},y_{2n},d_{2n})

The integrated full solution will be
$(x_{11},y_{11},d_{11})(x_{12},y_{12},d_{12})$………$(x_i,y_i,d_i)(x_{2i+1},y_{2i+1},d_{2i+1})$ …………….(x_{2n},y_{2n},d_{2n})

We take the left part from left sequence and right part from the right sequence.

The final solution is added to the solution set, if crossover is possible. If there is more than one common point, the crossover takes place for each of these points. All these are added to the solution set.

4.5. Mutation

In this method we try to find a random solution out of the given solutions. This is done to introduce randomness in the algorithm, so that the algorithm does not get trapped in local minima. This process occurs with a probability of 0.02. In this process any one full solution is chosen. We take any two arbitrary points on this solution. Then we try to find out full solutions between these two points. Hence we try to find out a path between these points using algorithm similar to the algorithm used to find the initial solution. If such a path is found we replace the path between these two points with this new path. If more than one solution is found, we replace the old path by the one with the least count of fitness function (most fit). This is added to the solution set.

5. CASE II: Algorithm Artificial Neural Network (Back Propagation)

This algorithm tries to find out the solution by learning from the historical data. The algorithm used is the conventional ANN Back Propagation to find out the goal (final position) starting from the initial position. We know that if we train the robot how to act in various obstacle situations, the robot can ‘learn’ the various actions to be taken [9, 15]. In this algorithm we use the following:

No. Of Inputs: 26
No. Of Hidden Layers: 2
No. of Output Layers: 3
No. of Neurons in hidden Layers: 20, 15
Activation Function for the three layers: tansig, purelin, purelin
Range of Inputs: 0 to 1
Range of each output 0 to 1

Each position is marked as (x,y,d)
Where x=x coordinate
y=y coordinate
d=direction (0-7)

5.1. Explanation of the Inputs

There are total 26 inputs to the neural network (I_0-I_{25}). The I_0 and I_1 denote the rotations needed for the robot to rotate from its present direction to the direction in which the goal is present. We know that the rotation can be either -2 or +2 (180 degrees left or 180 degrees right), -1 (90 degrees left), 0 (no rotation), +1 (90 degrees right). These are represented by these 2 inputs as follows:

- $(I_0,I_1)=(0,0)$ represents +2 or -2
- $(I_0,I_1)=(0,1)$ represents +1
- $(I_0,I_1)=(1,1)$ represents 0
- $(I_0,I_1)=(1,0)$ represents -1

It should be noted that these are numbered according to gray codes so that even if one of the bits is corrupted by noise, the effect on the neural network is the minimum.

The other bits (I_2 to I_{25}) represent the condition of the map. Considering the whole map as a grid of (MXN), we take a small portion of it (5X5), with the robot at the center of the map. Each coordinate of this map is marked as 1 if the robot can move to this point in the next step, or 0 if the point does not exist or the robot cannot move to this point because of some obstacle. So we have taken a small part of the graph. If (x,y) be the coordinate of the robot, we take (x-2,y-2)……(x+2,y+2) as the input. These are 25 points. These points, excluding the center of the graph (where our robot stands) are fed into the neural network.

5.2. Explanation of the Outputs

There are total 3 outputs (O_0,O_1,O_2), each can be either 0 or 1. We can make 8 combinations of outputs using these. The meanings of these outputs are as under:

- (O_0,O_1,O_2)=(0,0,0) Rotate left 90 degrees.
- (O_0,O_1,O_2)=(0,0,1) Rotate left 45 degrees.
- (O_0,O_1,O_2)=(0,1,0) Move forward.
- (O_0,O_1,O_2)=(0,1,1) Rotate left 45 degrees and move forward.
- (O_0,O_1,O_2)=(1,0,0) Do not move.
- (O_0,O_1,O_2)=(1,0,1) Rotate right 90 degrees.
- (O_0,O_1,O_2)=(1,1,0) Rotate right 45 degrees and move forward.
- (O_0,O_1,O_2)=(1,1,1) Move right 45 degrees.

It should be noted that this sequence is in gray code as well, if we arrange it in gray code sequence, the order of the moves will be 90 degrees left rotate, 45 degree left rotate, 45 degree left rotate and move forward, move forward, 45 degrees rotate right and move forward, 45 degrees rotate right, 90 degrees rotate right. Hence there is a transition in series. This ensures that the effect of noise is minimum.

5.3. Special Constraints put in the algorithm

There were a few limitations of the algorithm, these were removed by applying some constraints to the algorithm, which ensured the smooth working of the algorithm

- It may happen that the robot does not move or keeps turning its position continuously. Since there is no feedback in test mode, this is likely to continue indefinitely. For this we apply the constraint that if the robot sits idle for more than 5 time stamps and does not make any move (excluding rotations), we take a random move.
- It may happen that the robot develops affinity to walk straight in a direction, hence if it is situated in the opposite direction, it may go farther and farther from the destination. For this we limit that if the robot makes 5 consecutive moves (excluding rotations) which increase its distance from the destination, we make the next one move only which decreases its distance (excluding rotations).

5.4. Procedure

The algorithm first generated test cases, so that the established neural network can be trained. These are done by generating input conditions and solving them using A* algorithm and getting the inputs and outputs.

The neural network is established and trained using these test cases. We lay more stress on the rotational parameters of the robot, ie the capability of the robot to rotate from its present direction to the goal direction. After the training is over, we enter into the test case. Looking at the condition of the map, the input cases are generated and are fed into the Neural Network. The output is fetched and decoded. The same is followed and robot is moved.

6. CASE III: A* Algorithm

This algorithm uses heuristic function to optimize the path traversed from the start to the end. The algorithm used is the conventional A* algorithm to find out the goal (final position) starting from the initial position [18, 19]. In this algorithm we use the following costs:

g(n) = depth from the initial node, increases by 1 in every step
h(n) = square of the distance of the current position and the final position + R(n)
R(n) = minimum rotational time required for the robot in its entire journey assuming no obstacles
f(n) = g(n)+h(n)
where n is any node.

The function R(n) is exactly similar to what we had taken in Genetic Algorithm discussed earlier.

7. Comparison of Genetic, ANN and A* Algorithms

We have seen the working of the algorithms. We saw all these algorithms working and guiding the robot ensuring that the robot gets to the final destination without collisions. The algorithms used different ideologies but guided the robot well to reach the final destination.

The following are the key points of comparison between these algorithms:

- The Genetic Algorithm takes a lot of time to generate the result as compared to Artificial Neural Network and A*.
- The overall path traveled by the robot using genetic algorithm was less as compared to Artificial Neural Networks.
- The Artificial Neural Network Algorithm failed if highly chaotic environment was applied. Also if a semi maze like condition was given, the neural network algorithm failed for similar reasons.
- The memory requirements for the Genetic Algorithms are very high as compared to Artificial Neural Network.
- A* Algorithm, when applied to low size inputs works best both in terms of time and speed. The problem comes when the input is too large and exceeds the limits of memory or takes too long time to calculate moves.

All these algorithms were highly efficient when only some obstacles were applied or no obstacle was applied. Even the path traced was very straight, which proves the efficiency of these algorithms.

8. Results

For the testing of the algorithm we made a program using Java Applets that generated n number of obstacles each controlled by an independent state. These obstacles were moved completely randomly independent of each other. These were also moved on the lines of collision free movement. Another thread was started that moved our robot from the initial to the final position. This thread used the algorithm discussed to move the robot in its path.

All the obstacles and our robot were displayed in the applets. The directions were displayed as the head pointing towards the direction. An open figure denotes obstacle and a closed denotes robot. The applet also collected the complete path followed by the robot to give a path trace of the robot.

The algorithms were tested using this technique. In all the movement of the robot at each step was recorded using all the three algorithms. The results are given in next sections.

8.1. Genetic Algorithm

The first algorithm used was genetic algorithm. Here we had used a sample grid space on which the robot is to be moved of [100 X 100] dimension. The coordinates could vary from (0,0) to (99,99). We used total 750 obstacles, which all could move a unit step at any unit time. We moved the robot from position (0,0) to the goal position (99,99). The path traced by the robot is given in figure 4(a).

8.2. Artificial Neural Networks

A similar run was performed for the Artificial Neural Networks as well. The parameters were quite similar to the ones used in the previous algorithm. Sample grid space on which the robot is to be moved was of [100 X 100] dimension where the coordinates could vary from (1,1) to (100,100). MATLAB was used for the simulation purpose. We used total 500 obstacles, which all could move a unit step at any unit time. The threshold time was fixed to be 1 seconds. We moved the robot from position (1,1) to the goal position (99,99). The path traced by the robot is given in figure 4(b).

8.3. A* Algorithm

The third algorithm used for the problem was A* Algorithm. Here also the grid size was [100X100]. The coordinates could vary from (0,0) to (99,99). We used total 1000 obstacles, which all could move a unit step at any unit time. The threshold time was fixed to be 1 second. We moved the robot from position (0,0) to the goal position (99,99). The path traced by the robot is given in Figure 4(c).

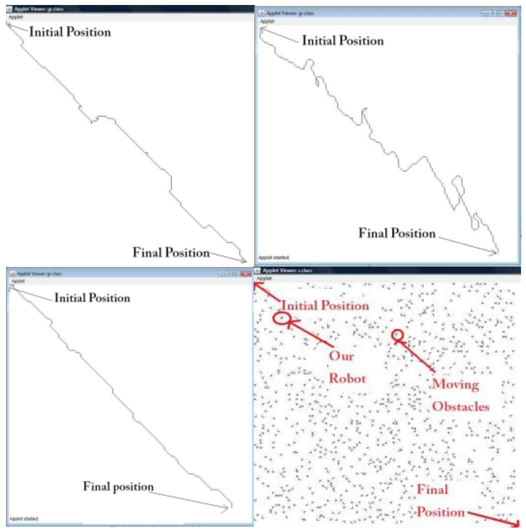

Figure 4(a-d): Path traced by robot using Genetic, ANN and A*algorithm and condition of board at any general time

The condition of the board at a random time is given in figure 4(d) (Note: coordinate (0,0) is the top left coordinate).

Closely watching the robot go towards its goal, we see that there is no collision on its way. Hence we have been successful in avoiding collision. Also we find looking at the path traced, that the path is optimal with respect to the conditions given.

Carefully observing the path traced us finds the following kinds of paths:

- **Stationary Phase**: In this the robot is unable to make a move. Due to the extremely chaotic conditions, it is not possible to find a solution. The robot waits for the time it gets a path. If there are a set of obstacles in very close vicinity, it may cause this problem.
- **Straight Phase**: Here the robot moves straight towards the goal. The obstacles have no effect on its movement. If we take very few obstacles, this is what normally happens.
- **Collision Avoidance Phase**: If an obstacle is very close to the robot, it takes a stern turn to avoid collision and keep going. This phase is known as the collision avoidance phase.
- **Backtracking Phase**: If the robot happens to deviate a lot from its path due to the excessive number of obstacles in its close vicinity, it backtracks to its path and then continues to move further.

These are shown in figure 5.

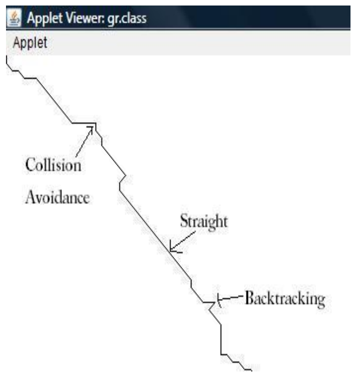

Figure 5: The various types of paths in the motion of robot

The more chaotic the scene becomes, we see the number of straight paths decreasing and the number of stationary, collision avoidance and backtracking paths increasing. These paths are more prone to vary depending on the number of obstacles in close vicinity rather than those in the entire grid. The efficiency of the algorithm lies in the measure of more the straight paths and lesser the backtracking paths it produces.

9. Conclusions

We have been able to move the robot from initial to final position in an environment of dynamically moving obstacles without collision. The path chosen is optimum. This technique can be used to enable the movement of many robots together in a common place.

We used first Genetic Algorithm and solved the navigation control problem. We saw that the algorithm does take time for the generation of initial solutions, but generates good results. The results keep getting better at each iteration. The net result is that the algorithm is able to guide the robot well, making the path collision free.

We also saw Artificial Neural Networks. We need to train the network once, which is time consuming process. But once done, the network gives a very fast response. This algorithm is very suited to collision avoidance in path when the obstacles come one by one. The network can be easily trained for such conditions.

We also saw the working of A* algorithm. This algorithm very efficiently optimized the path of the robot from the start to the goal ensuring that there are no collisions. Since we used A* algorithm, it can be said that every unit of time, the best paths were generated and based on them the moves were made. This algorithm worked very efficiently, even in highly chaotic times, to generate good solutions.

The paper clearly shows the effective working of the three algorithms. The comparative analysis can very easily be done looking at the various runs. We found the A* algorithm was much efficient and exceeded its counterparts. It was better in finding the results early. Although we know that performance of various algorithms will change with the change of parameters and input size.

The types of points (stationary, straight, collision avoidance or backtracking) are another measure of the effectiveness of the three algorithms. Here also, looking at the output sequence, we find that the A* algorithm working better.

Looking at the three algorithms we can easily make out that how the algorithms will behave for different inputs. Genetic Algorithm is the poorest for a maze like situation. On the contrary if input size is immense, ANN will be the best. Keeping these algorithms as base a better comparative analysis may be done in the future. Also some new results are likely to come up for different input parameters which may be experimented for the future.

10. References

[1] Hutchinson, S. A. and Kak, A. C., "Planning sensing strategies in a robot work cell with Multi-sensor capabilities," *IEEE Trans. On Robotics and Automation*, vol.5, no.6, 1989.
[2] Rich, E. and Knight, K., *Artificial Intelligence*, McGraw-Hill, New York, pp. 29-98, 1991.
[3] Takahashi, O. and Schilling, R. J., "Motion planning in a plane using generalized voronoi diagrams," *IEEE Trans. on Robotics and Automation*, vol.5, no.2, 1989.
[4] Borenstain, J., Everett, H. R., and Feng, L., Navigating "Mobile Robots: Systems and Techniques", A. K. Peters, *Wellesley*, 1996
[5] Matlin, W. Margaret, Cognition, Hault Sounders, printed and circulated by Prism books, India, 1996.
[6] Konar, A. and Pal, S., "Modeling cognition with fuzzy neural nets" In Fuzzy Systems Theory: Techniques and Applications, Leondes, C. T., Ed., Academic Press, New York, 1999.
[7] Pagac, D., Nebot, E. M. and Durrant. W., H., "An evidential approach to map building for autonomous robots," *IEEE Trans. On Robotics and Automation*, vol.14, no.2, pp. 623-629, Aug. 1998.
[8] V. Ayala-Ramirez, A. Perez-Garcia, E J. Montecillo-Puente, R.E. Sanchez-Yanez, "Path planning using genetic algorithms for mini-robotic tasks", 2004 *IEEE International Conference on Systems, Man and Cybernetics*
[9] Hem Fkezza-Buet, FrBd6ric Alexandre "Modeling prefrontal functions for robot navigation"
[10] Theodore W. Manikas, Kaveh Ashenayi, and Roger L. Wainwright, "Genetic Algorithms for Autonomous Robot Navigation", *IEEE Instrumentation & Measurement Magazine December 2007*
[11] Du Xin, Chen Hua-hua, Gu Wei-kang, "Neural network and genetic algorithm based global path planning in a static environment", *Journal of Zhejiang University SCIENCE*
[12] Zhang Huan-cheng, Zhu Miao-liang, "Self-organized architecture for outdoor mobile robot navigation", *Journal of Zhejiang University Science*
[13] Peter Corke, Ron Peterson, Daniela Rus, "Networked Robots: Flying Robot Navigation using a Sensor Net", April 18, 2003
[14] Cory Quammen, "Evolutionary learning in mobile robot navigation", *The ACM Student Magazine*
[15] Yong-Kyun Na and Se-Young Oh, "Hybrid Control for Autonomous Mobile Robot Navigation Using Neural Network Based Behavior Modules and Environment Classification", 2003 *Kluwer Academic Publishers*, Manufactured in The Netherlands
[16] Seyyed Ehsan Mahmoudi, Ali Akhavan Bitaghsir, Behjat Forouzandeh and Ali Reza Marandi, "A New Genetic Method for Mobile Robot Navigation", *10th IEEE International Conference* on *Methods and Models in Automation and Robotics*, 30 August - 2 September 2004, Miedzyzdroje, Poland
[17] Torvald Ersson and Xiaoming Hu, "Path Planning and Navigation of Mobile Robots in Unknown Environments"
[18] László Kiss, Annamária R. Várkonyi-Kóczy, "A Universal Vision-based Navigation System for Autonomous Indoor Robots"
[19] Sven Behnke, "Local Multiresolution Path Planning", Preliminary version in *Proc. of 7th RoboCup Int. Symposium*, Padua, Italy, 2003
[20] S. Veera Ragavan, and V. Ganapathy, "A Unified Framework for a Robust Conflict-Free Robot Navigation", *Proceedings of World Academy of Science, Engineering and Technology*, Volume 21 January 2007 ISSN 1307-6884
[21] Shukla, Anupam; Tiwari Ritu and Kala, Rahul, "Mobile Robot Navigation Control in Moving Obstacle Environment using A* Algorithm", *Intelligent Systems Engineering Systems through Artificial Neural Networks, ASME Publications*, Vol. 18, pp 113-120, Nov 2008

About the authors

Dr. Anupam Shukla

Dr. Anupam Shukla is an Associate Professor in the IT Department of the Indian Institute of Information Technology and Management Gwalior. He has 20 years of teaching experience. His research interest includes Speech processing, Artificial Intelligence, Soft Computing and Bioinformatics. He has published around 62 papers in various national and international journals/conferences. He received Young Scientist Award from Madhya Pradesh Government and Gold Medal from Jadavpur University.

Dr. Ritu Tiwari

Dr. Ritu Tiwari is an Assistant Professor in the IT Department of Indian Institute of Information Technology and Management Gwalior. Her field of research includes Biometrics, Artificial Neural Networks, Signal Processing, Robotics and Soft Computing. She has published around 25 papers in various national and international journals/conferences. She has received Young Scientist Award from Chhattisgarh Council of Science & Technology in the year 2006. She also received Gold Medal in her post graduation from NIT, Raipur.

Rahul Kala

Rahul Kala is a student of 4th Year Integrated Post Graduate Course (BTech + MTech in IT) in Indian Institute of Information Technology and Management Gwalior. His fields of research are robotics, design and analysis of algorithms, artificial intelligence and soft computing. He secured All India 8th position in Graduates Aptitude Test in Engineeging-2008 with a percentile of 99.84. He has published 8 papers in various national and international journals/conferences.

Sourabh Rungta
Sourabh Rungta is presently serving as a reader in Department of Computer Science and Engineering at Rungta College of Engineering.

RR Janghel

RR Janghel is a PhD student in Department of Information Technology at Indian Institute of Information Technology and Management Gwalior.