Optimal Path Planning In Autonomous Robots Using Genetic Algorithm

[1] Basavanna M [2] Dr. M. Shivakumar,
[1] Assistant Professor, [2] Professor & Dean
[1][2] Dept of E&IE ,GSSS Institute of Engineering & Technology for Women, Mysore

***Abstract:* -- Determination of a collision free path for a mobile robot between start and goal points through obstacles in a workspace is important to the design of an autonomous robot path planning. One way to make sure the robot's path is the best path for the given environment is by making use of various path planning algorithms. In this paper we use a genetic algorithm approach to solve the path planning problem. The obstacle free path is chosen such that the robot avoids all the obstacles in its path and finds the shortest path. Then the Processing Time and Path Length for the given workspace is obtained. The results obtained for genetic algorithm are compared with the results obtained for the same workspace using other two algorithms such as A – Star and Bidirectional RRT Algorithm.**

***Index Terms* – Genetic algorithm, A-star algorithm, robots, path planning, optimal path, robot map, MATLAB.**

I. INTRODUCTION

With advance in technology many industries are employing robots to perform many simple as well as complex tasks. These tasks can be time consuming, laborious or pose health hazards for human begins. Robots can be more efficient and save time and labour when compared to human beings. The workspace\environment of the robot is often cluttered with obstacles. And suitable computations have to be done to obtain an optimal path from the source to the destination. This problem of finding an optimal path from the source to the destination is called as the path planning problem. There are many path planning algorithms available to solve this problem. In this paper, we use genetic algorithm to find an optimal path. The codes for genetic algorithm are written using matlab. The code is written such that it accepts a robot map and then applies the algorithm on that robot map. The final output obtained is the time taken by the robot to reach the destination from the source, the path length in pixels and the path established on the robot map.

A – star algorithm is a spanning algorithm. It is a heuristic algorithm. The formula for this algorithm is given by: $f(n) = g(n) + h(n)$; where n is the current pixel. F(n) is the score associated with n and g(n) is the cost to reach from the source pixel to n [2]. Bidirectional rrt algorithmis a branching algorithm. As the name says, it starts branching from the source as well as the destination. Hence, this algorithm starts searching for path from both the directions. When both these path frontiers meet, a path is established. [3]

II METHODLOGY

Let us assume that a workspace consists of a overhead camera as shown in the Figure 1.

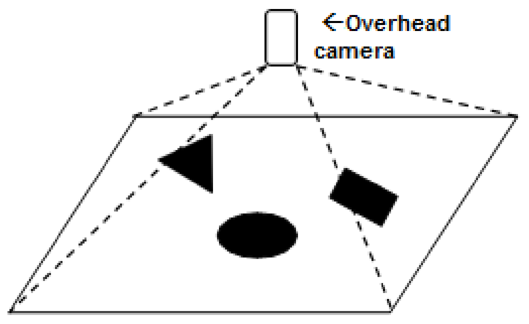

Figure 1: Overhead camera

The image coming from the camera can be used to create a robot map. The robot map is shown in Figure 2. This is a simplistic implementation of the real life scenarios where multiple cameras are used to capture different parts of the entire workspace, and their outputs are fused to create an overall map used by the path planning algorithms. The same camera can also be used to capture the location of the robot at the start of the planning and also as the robot moves. The robot is not shown in the map in Figure 1 & 2.

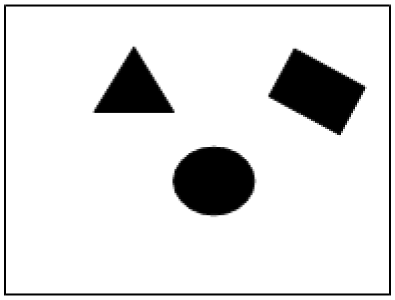
Figure 2: Robot map

Here, Figure 2 shows a robot map which consists of obstacles of various shapes. Genetic algorithm is applied to a similar robot map. We obtain processing time and path length and an output image which will be a robot map with the shortest path (as shown in the results section). This output image will be loaded onto the robot using AVR Bootloader. The robot will then reach the destination following the path that has been fed to it.

III GENETIC ALGORITHM

The path planning problem is solved using genetic algorithm. It is an optimal search algorithm. In a genetic algorithm the first step is to define the chromosomes and genomes. A chromosome is a complete path from the robot source to the destination. Genomes are a set of points. A path can be completely made by these set of points (genomes). The algorithm can be implemented as follows.1) Generation of initial population. 2) Defining the crossover operator. 3) Defining the mutation operator. 4) Defining the elitism operator. 5) Choosing the selection method. 6) Defining the fitness function. 7) Choosing the algorithm termination condition. 8) Defining the overall loop. [1]

1) Generation of initial population – This is the first and most important step in solving path planning problem using genetic algorithm. An initial population should be generated such that it should be distributed uniformly throughout the workspace\environment. Hence, to generate an initial population few points are chosen in the workspace that forms a path, this path joins the source to a point and this point is joined to another point, it is joined until the robot destination is reached. The following steps are followed for the generation of initial population. a) The workspace is converted into grid by parallel lines both in *X* and *Y* directions. Each junction point is assigned with some value say a_i, where i = 1,2,....,n. b) choosing *k* = 1. c) Now, finding a path from the source to $a_k$and from $a_k$to the destination. d) Let *k* = *k* + *1*. e) Repeat step c through d if *k* is less than or equal to n. If any one of the point lies on the obstacle then that point cannot be considered for path planning.

2) Defining the crossover operation –This algorithm is based on Darwin's theory of evolution. Hence, natural processes occur. Crossover operation is the most important operation. The steps for the crossover operation is as followed. a) Let n be the number of genomes in a chromosome and g(i) be the ith genome of the chromosome. b) Let i = 1. c) For j = n down to i + 1 ; find g(j) that has a direct sight to g(i). d) for k = i + I to j - 1 ; delete g(k) from the chromosome. e) Let i = j. f) Repeat steps c to e until i = n .

3) Defining the mutation operator –This is done in order to maintain the diversity amongst the chromosomes (mutations of genomes). The genome of a chromosome is not a single number it is made up of two parts; *X* and *Y*. To mutate a genome two random numbers in the interval [-50,50] are generated with uniform distribution over the workspace. These two random numbers are added to *X* and *Y* parts of the genome. If it is placed on the obstacle then some other random number is chosen and it is added with the parts. That is add the new random number with *X* and *Y*.

4) Defining the elitism operator – The best chromosomes are not changed and passing of these unchanged chromosomes from one generation to the next is the elitism.

5) Choosing the selection method – The selection method is based on Roulette Wheel Selection. The Roulette Wheel Selection randomly selects chromosomes from one generation. This will form the basis for the further generation. The requirement of this is that fittest chromosomes will have a greater chance compared to the weaker ones.

6) Defining the fitness function - Some fitness value has to be given to each chromosome in order to separate the accepted chromosomes and rejected chromosomes. Since we are mainly focused on finding the shortest path we have to choose fitness function such that it gives small values for short path and large values for longer paths. The fitness function is designed as – fitness =

$$\frac{10^6}{\sum_k \left|\left|G(k) - G(k+1)\right|\right| + 1}$$

Where *G(k) is the Kth*genome of a chromosome.

7) Choosing the algorithm termination condition – The algorithm is terminated based on the improvement rate formula.

8) Defining the overall loop – The overall loop is the above 7 seven steps combined together.

IV PROBLEM SOLVING USING GENETIC ALGORITHM

To model the problem as an optimization problem, we need an objective function and specification of variables of that objective function (along with their bounds). Let a path be characterized by a fixed number of points in the robotic map. In order to make some path from this set of points, we start from the source and connect it to the first point by a straight line. The first point is connected to the second point by a straight line, and so on. At the end the last point is connected to the goal. The objective function is the length of this path. A heavy penalty is added if any part of the path lies inside the obstacle, while the penalty is proportional to the length of the path inside the obstacle. The locations of each of these fixed number of points (both x and y axis positions) are the optimization variables. The variable bounds are such that the point lies inside the map (lower bound 1 and upper bound as the length/width of the map for the x/y axis). All points (both x and y axis values) put one by one make the genetic individual used for optimization. The concept is shown in figure 3.

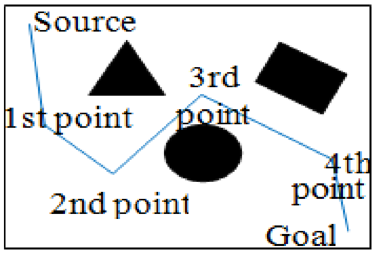

Figure 3: GA individual Representation

Each point in the path marks a point of turn. The total number of points is an algorithm parameter and should be equal to the maximum number of turns a robot is expected to make in the robot map. Setting this number too high would result in very large computational requirements. If the algorithm is not allowed a large computational time, random results may be the output. Setting a large value in simple scenarios will result in useless turns and hence a high path length. Too small value of the parameter may not give enough flexibility to the algorithm to model the optimal path, thus resulting in collision-prone paths.

V RESULTS AND COMPARISION

Here we apply three different algorithms to the same workspace (the algorithms are applied to the robot map of the workspace). We then compare the processing time and path length. We can also see from the figures 4, 5, and 6 the path a robot takes depends on the algorithm used.

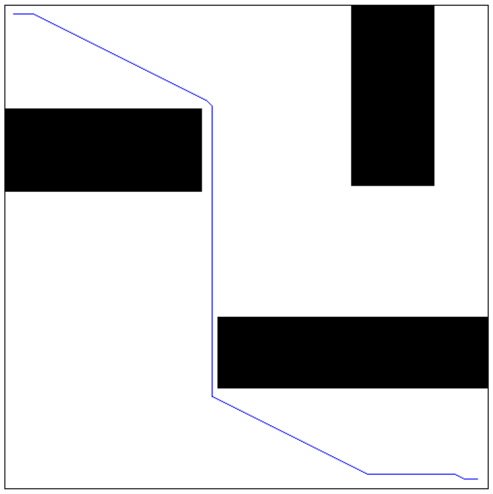

Figure 4: robot map of a workspace; the robot follows the path shown in blue; when a – star algorithm is used.

The processing time and path length for the above robot map using a – star algorithm are 252 seconds and 1833 pixels.

Figure 6 shows the robot map and path a robot takes when bidirectional rrt algorithm is applied to it.

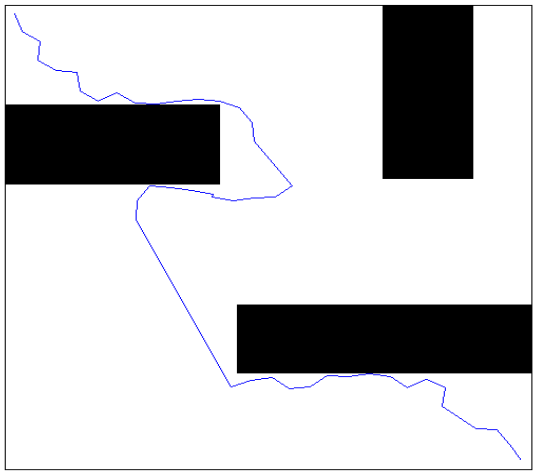

Figure 5: robot map of a workspace; the robot follows the path shown in blue; when bidirectional rrt algorithm is used.

The processing time and path length for the above robot map using bidirectional rrt algorithm are 148 seconds and 1581 pixels.

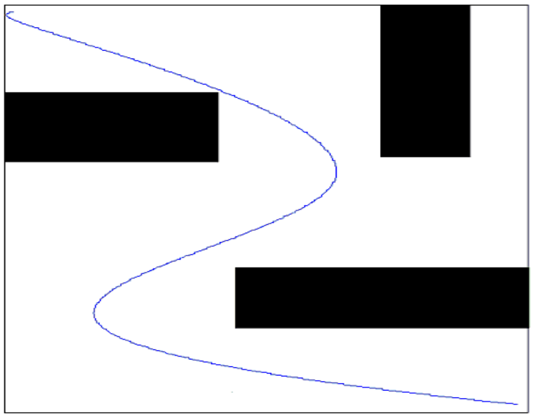

Figure 6: robot map of a workspace; the path taken by the robot is shown in blue. This output path is obtained when genetic algorithm is used.

The processing time and path length for the above robot map using ga is 111 seconds and 1312 pixels (the robot has travelled 1312 pixels out of 100*100 pixels). From the above figure we can see, at each point the robot changes its direction.

We confer from the results that it is advantageous to use the genetic path planning algorithm over the other two since this algorithm is more efficient in terms of the computational time required to calculate the shortest path (takes less time) and calculates the path to be taken in a very efficient way.

VI CONCLUSION

By comparing the processing time and path length of the three algorithms, we can see that genetic algorithm takes the least time and the distance it travels to reach the destination is also least of the three algorithms. By choosing genetic algorithm the robot can reach the destination in less time and travel accurately. Hence, the battery is conserved

Acknowledgements

The current work was made possible because of the grant provided by Vision Group On Science and Technology, Department of Information Technology, Biotechnology and Science & Technology, Government of Karnataka, Grant No.VGST/CISEE/2012-13/GRD188/282. We acknowledge the VGST for sanctioning a grant amount of Rs. 30 Lakhs for the "Establishment of Innovative Robotics Awareness Lab" at GSSS Institute ofEngineering and Technology for Women, Mysuru, and Karnataka, India.

REFERENCES

[1] Nasser sadati and javidtaheri, "genetic algorithm in robot path planning problem in crisp and fuzzified environments", intelligent systems laboratory electrical engineering department sharif university of technology.

[2] Zhanying zhang1 and ziping zhao2, "a multiple mobile robots path planning algorithm based on a-star and dijkstra algorithm", 1,2 college of computer and information engineering, tianjin normal university, tianjin 300387, china .

[3] Georges s. Aoude, jonathan p. How and ian m. Garcia, "two-stage path planning approach for designing multiple spacecraft reconfiguration maneuvers".

[4] N. Sariff and n. Buniyamin, "an overview of autonomous mobile robot path planning algorithms", 4th student conference on research and development (scored 2006), june 20061-4244-0527-0/06/$20.00 © 2006 ieee.

[5] O.khatib, "real time obstacle avoidance for manipulators and mobile robots," in international journal of robotics research, 5(10):90-98, spring 1986.

[6] Christophniederberger,ejanradovie, and markus gross, "generic path planning for real time application," in proceedings of the computer graphics international (cgi 04), 2004, pp 1-8.