ROBOT MOTION PLANNING

Mentor:
Dr. Rahul Kala

Assistant Professor

IIIT Allahabad

By

Akanksha Bhardwaj (RIT2015035)

Bobbili Vineela Moses (RIT2015041)

Shivani Pal (RIT2015059)

TABLE OF CONTENTS

TABLE OF FIGURES

INDEX OF TABLE

1. ABSTRACT

Robot motion planning is an important topic in the domain of robotics, specifically for mobile robots. It is a fundamental problem that forms basis for solving other complex robotics problems. There are many approaches for path optimization including potential field method, genetic algorithm, A*, Ant Colony optimisation algorithms. Here we discuss the evolutionary method for devising an acceptable path solution by following Genetic Algorithm. Since mobile robots have certain hardware and non-holonomic constraints, we make use of concepts such as Splines to overcome the limitations to some extent. Planning is done for static environment.

2. INTRODUCTION

An autonomous robot has the ability to carry out some specific tasks without human intervention or other external guidance. This adds up to the importance of developing software for such autonomous machines in order to solve problems. The problem we are going to discuss here is the motion planning of such robots, in particular mobile robots collectively known as AMR (Autonomous Mobile Robots) .
Have the abilities to navigate in an uncontrolled environment without the need for physical or electro-mechanical guidance devices.

Motion planning or navigation problem is a term used in robotics for finding a collision free path which should be shortest and optimal is other terms as per requirement for robot.

There are several robotic functions which fall under applications of motion planning such as automation, architectural design, video gaming, robotic surgery, robot design in CAD software, and for biological research.
We explore **Evolutionary Computation** for optimisation purpose in the project**.** According to Darwin's theory of evolution, every species in the world evolve through natural selection of those characteristics that increases their chances of surviving in the world.
Evolutionary algorithms generate a number of random solutions to the problem and evolve them towards better characteristics.

Genetic Algorithm (GA), a subset of evolutionary algorithms is widely used in solving optimization problems. Algorithm generate randomly the possible solution of initial set of individuals and then we apply the the objective function and calculate fitness value of each solution. To generate a new population of fitter candidates we apply genetic operations like mutation

and crossover . This is an iterative procedure till we reach satisfactory fitness value.
A robotic controller is a vital aspect of path planning.

3. BACKGROUND

3.1 Path Planning Algorithms

According to sensor type, robot capabilities, environment various approaches have been proposed for path planning. These approaches are gradually deployed towards achieving better performance in terms of time, distance, cost and complexity.
It is necessary that the algorithm finds a feasible solution to the problem when there exists one. If a feasible path from given source to goal if there a path exists. If no such path exists it must be capable of concluding and reporting the same in finite time.

3.2 Convergence

In evolutionary computation convergence is a term said to occur when all the individuals belonging to a population converge to similar values in terms of fitness or solution quality. Full and partial convergence can happen in case of genetic algorithm. It is one of the stopping criterion, when the solution is not improving any further the GA is said to achieve a stable state and should be terminated.

3.3 Overview

The project deals with both hardware and software aspects of path planning. The hardware part deals receiving encoder readings from the position encoders and driving the robot as it traces the path supplied by adjusting power supplied to the wheel motors in real time. For minimizing the error between ideal trajectory and actual trajectory points a PID controller has to be implemented. PID stands for Proportional, Integral and Derivative. It is explained in section 5.5.

The primary focus is to derive a trajectory which gets as close as possible to the best trajectory possible between a given start point S and a goal configuration G. This involves exploring various path configurations and testing them against our criteria of optimum path. The challenge is to get as close as we can get to our definition of the best path[1] with the help of software and hardware tools at our disposal.

Path in motion planning can be achieved in two type of environments, static and dynamic.

- In **Static** Environment the environment is completely known beforehand as the obstacles are immobile.
- **Dynamic** environment planning is a real time job as obstructions may change their positions with respect to time. This requires path updation in upon every collision.

We would be dealing with static environment for this project.

The obstacle geometry of path and the workspace of robot is described in a 2D configuration such as using a matrix where each cell denoted a real location in map. The optimization done using grid like these is called grid search. In fig. 1 an example map grid is shown. For indicating presence of obstacles in environment 1 is used, 0 refers to empty space. A feasible path refers to obstacle free trajectory. For improving the path in terms of distance and safety we require maximization of the fitness values of path.

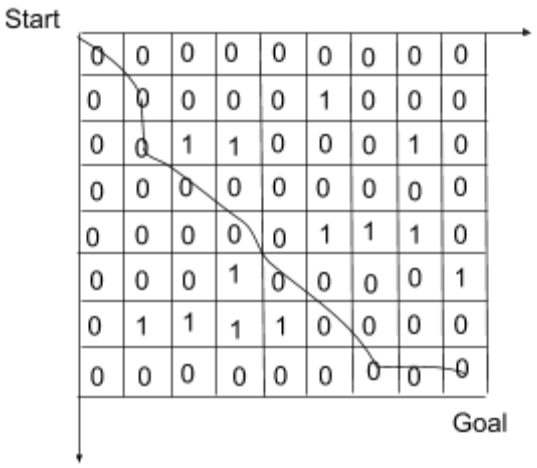


Fig.1 : Example map grid

We have to find a collision free continuous and shortest path between source and destination points in an environment of 2D configuration of surrounding objects (obstacles,wall etc.) Here we are dealing with Amigobot(Mobile Robot) moving in a static environment.

Goals are defined below.

1) Obtain collision free trajectories.
2) Optimize the path between start and goal positions till Genetic algorithm converges.
3) Implement a PID controller to deal with crosstrack errors during motion of robot.
4) A smooth trajectory should be obtained as steering angle constraints don't allow a robot in real life scenarios to take sharp turns.
5) Robot should reach the goal location in minimum time.

4. PROBLEM DESCRIPTION

4.1 INPUT

The map of the environment represented by a 2D matrix, a start configuration and a goal configuration both denoted by cartesian coordinates is given as input to the path optimization algorithm. Initially randomly generated path points are encoded as a chromosome of real numbers and supplied as input to the genetic algorithm, the start and the destination points not being a part the individual. The length of this individual is taken as fixed. The map is expected to have obstructions such as walls, objects marked. We require a map of minimized noise for accurate location of obstacles.

4.2 OUTPUT

The output of the genetic algorithm is a set of points that define an optimised trajectory joining the source and destination points. In case no feasible path exists it should return some known indicative value. Since this path is not suitable for a mobile robot to follow yet due to non-holonomic constraints it is then converted to B-spline curve for smoothing. The resultant path is given to the robot controller.

4.3 CONSTRAINTS

The parameters involved in genetic algorithms are to be tuned in order to arrive at values that provide the best results. The parameters in question are :

- Population size
- Mutation rate
- Crossover rate
- Maximum generation size
- Tournament size (selection)
- Range for real uniform mutation

Encoding the map of the environment into a 2D matrix, representing each pixel with a cell is advantageous for saving computation time but is memory consuming if map size is considerably large or matrix is sparse in such cases we can adopt 1D representation method where only obstacle coordinates are stored in sorted manner. This approach increases the calculations and hence lookup time goes up. The feasible path needs to address the non-holonomic constraints (refer section 5.4) of the robot as well.

5. LITERATURE REVIEW

5.1 Genetic algorithms

It is a stochastic search technique, used to optimize an objective function over a known search space. According to [research paper x]

Generating an initial population involves random generation of n individuals, where n is chromosome size, for our case it is fixed.

5.1.1 Individual

Individual is analogous to chromosomes. In GA each solution is represented as an individual. Data string comprised of either binary or real values of chromosomes is divided into different sections known as genes.

5.1.2 Population

Set of individuals is called as population. Initially we fill the population with random individuals. Number of chromosomes is called population size.The set of chromosomes forms individual, set of individuals forms population pool. The population undergo three operators for generating new offsprings - Selection, Mutation, Crossover[9].

5.1.3 Fitness Evolution

Each individual has fitness value associated which suggests the relevance of that solution against our problem and is used for offspring generation. The characteristics that promote optimality in solution are passed forward to successive generations. In the research paper x it is shown that the main CPU cost goes into fitness evaluation, about 99% in a typical Evolution Algorithm. Usually in path planning problem the fitness is judged by the path length and path points in obstacle penalties.

5.1.4 Selection

The process in which genomes are selected for next generation. It can be stochastic or deterministic but is directly proportional to the fitness values in both cases. To help keep the population diverse enough mutation and crossover operators are used along with it. Ways to implement selection, one way might favour some individuals over others:

1. Tournament selection :Repeatedly selecting the best solution from the random population by playing a certain number of tournaments
2. Roulette wheel selection: This is implemented by assuming a wheel divided into sections, the area of each section being proportional to the fitness value of each individual. Selection is made randomly in the same way as a roulette wheel rotated.
3. Rank selection: In this the individuals are sorted according to their fitness values and the individual on the top of the list has more probability for selection.

5.1.5 Crossover

The process in which genes of multiple individuals are exchanged for generating one or more offsprings.

5.1.6 Mutation

The individuals selected for mutation stochastically undergo some

modification of genes to produce new ones. This operation is especially done to escape the local optimum solutions, increase diversity in population.

5.1.7 Crossover rate

It is probability of selecting an individual for undergoing crossover operation, highly influenced by fitness value.

5.1.8 Mutation rate

It is the probability of selecting of a particular gene for mutation.

After this step, combining all these individuals a new population is generated.

5.1.9 Crossover and Mutation in Real-Coded Genetic Algorithms

The implementation of crossover and mutation operators differ in binary and real GA although the high level framework remains the same. Methods to achieve crossover operation for real encoded GA are Single Point Crossover, Linear Crossover, Blend Crossover and Simulated binary crossover. In the Fig.1, Single point crossover is shown, the marked line depicts crossover point. A point is selected randomly along the chromosomes length and offsprings are formed by inheriting left set of genes from one parent and right from other. This results in two offsprings.

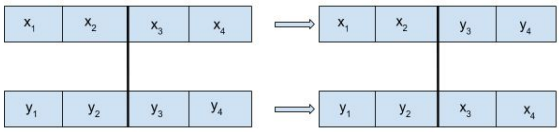


Fig.2: Single Point Crossover

Random Mutation, Non-uniform Mutation, Normally distributed Mutation can be used for Real GA. Mutation in binary-coded genetic algorithm is considerably less recommended than real-coded genetic algorithm.

In general the performance with real encoded individual has been found better than binary encoded ones in GA. The results of experiments conducted in the paper [3] by Bhushan Mahajan and Punam Marbate were in favour of the above argument. They compared performance of an implementation of floating point encoding with a binary encoding implementation by varying chromosome sizes and noting CPU time for each result.

5.1.10 Genotype

The variables involved in the problem are encoded in genotypic notation on which genetic operators are applied. Data structure can be a string of real values or binary values. We have used real coded genetic algorithm because it highly intuitive to represent path as a vector of coordinates with domain as real values.

5.1.11 Phenotype

Phenotype representation is used to evaluate the fitness function.

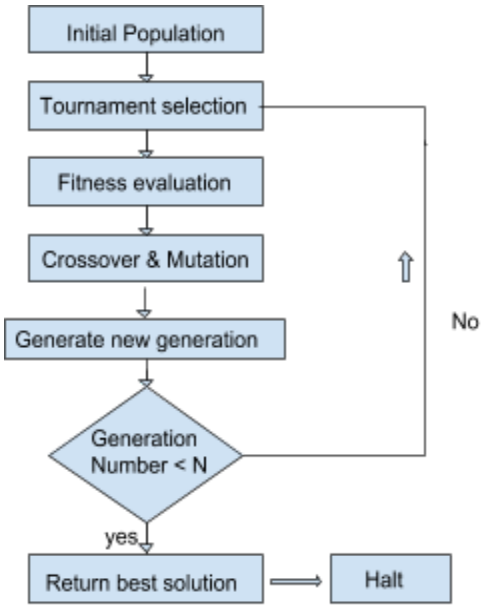
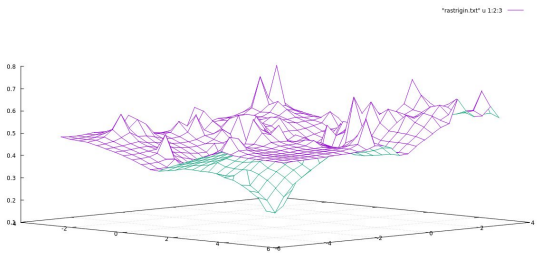


Fig. 3: Flow-chart of Genetic Algorithm

Fig. 2 shows the flow of a typical genetic algorithm with stopping condition as the maximum number of generations 'n'. There can be other criterias for choosing when to stop the algorithm like reaching an optimal which cannot be improved further with any number of iterations or if maximum allotted computation time is reached.

5.2 Benchmark functions

To test the performance of our genetic algorithm library implementation we used **rastrigin** function. It is a non-linear optimisation problem which is fairly hard to solve due its multimodal nature, it has a large number of local minima present, making it challenging for the algorithm.



$$f(x) = 10 * n + x1^2 + x2^2 - 10(\cos(2\pi x1) + \cos(2\pi x2))$$

Equation 1: Rastrigin function

For an n-dimensional rastrigin function global minima is present at
$f(x) = 0$, xi = 0, i = 1 to n

For n = 2, the algorithm returned near optimal results, within $10^{-5} - 10^{-6}$

$f(x1, x2)$	$x1$	$x2$
0.000059571967	-0.000038869627	0.000001002786

Table No. 1 Result of rastrigin function test

5.3 Kinematics of a differential wheel driven robot

We are planning the motion for a mobile robots with differential wheel drive. It defines how the robot moves. Such robots have typically two wheels on each side sharing a common axis, more than two wheels are present sometimes to avoid tipping over during motion. Each wheel is independent in the sense that each can be supplied different velocities. The point about which the wheels rotate is Instantaneous Center of Curvature.

5.4 Non Holonomic Constraints

Constraints on robot systems can be holonomic or nonholonomic. The following constraints are classified as non-holonomic and have to be kept in mind while motion planning.

- ❑ No-slip condition on wheels of mobile robots.
- ❑ The usual collision avoidance constraints with walls and obstacles present in surroundings.
- ❑ steering angle constraints, robot cannot turn on sharp angles abruptly.
- ❑ hardware construction limitations on joints movement of robot,etc.

Examples of vehicles having non-holonomic constraints are dual wheel and tricycle.

5.5 PID Controller

An intelligent robot consists of sensors and actuators. They use their encoder readings to create model of the world and form a sequence of behaviors through which it will achieve the desired goal/Robot through actuators. For localization and mapping of the world and identify unexplored areas , sensors are used.

To improving accuracy and achieve control on the robot, PID controller is used. The use of a PID becomes highly necessary when robot moves on high speeds

The main logic of PID controller is that we want to maintain a set of value, like speed of a vehicle or capturing data from a sensor. We then take the present data as input so that we can compare them to the setpoint. Through this process we can calculate the error, i.e, (error = setpoint - actual reading). We will use this error value to make the actual reading closer to the setpoint calculating how much corrections need to be made in real time .

- ❑ P-Factor(Kp) - (Kp) is constant value used to increase or decrease the effect of Proportional
- ❑ I-Factor (Ki) - (Ki) is constant value used to increase or decrease the effect of Integral
- ❑ D-Factor (Kd) - It is constant value which is used to increase or decrease the Derivative effect

Pseudo Code:

Here is a simple loop that implements the PID control:

start:

error = (target_position) - (theoretical_position)

integral = integral + (error *dt)*

derivative = ((error) - (previous_error))/dt

output = (K_p error) + (K_i *integral)* + (K_d derivative)

previous_error = error

wait (dt)

goto start

5.5.1 crosstrack error

Cross track error refers to the difference between the target point(on the original trajectory) and actual point (retrieved from position encoder readings of the bot) .

In order to achieve this a leader-follower scheme is adopted where the leader or ghost robot is always assumed to be on the accurate position at all times. It is calculated by noting the difference between position of actual robot with ghost robot or a virtual leader. The fig. 6 the robot forms an angle θ with virtual leader's position and α with the horizontal axis.

The formulas used have been described in section 6.

Integrating feedback into your control system "closes the loop" and is essential for creating robust robots

Use the simplest controller which achieves the desired result • Tuning PID constants is very tricky, especially for integral constants • Consult the literature for more controller tips and techniques [1]

5.6 CURVE SMOOTHING

Any robot is not designed to follow these path as they do not follow the non-holonomic constraints . for this we need to make the path smoother before the controller being used for the purpose of making a robot move. The smoothening of the path of robot is done by application of Bezier curve and B-spline curves. Each of these curves takes a set of control points as input. We further improve the algorithm and use of Bezier and B-spline curves for the motion planning. B-spline curves attempt to connect one point to another using polynomial function. Interpolation of supplied control points gives a smooth curve joining the supplied control points. B-spline gives local control whereas beiser provides global control. Hence, for better accuracy B-splines are used for the purpose of path smoothing.

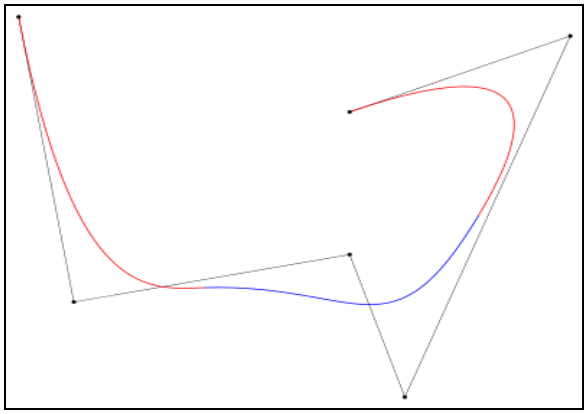


Fig. 4 : B-spline with control points

Fig. shows the B-spline curve for the corresponding path points, eliminating sharp and abrupt turns.

6. METHODOLOGY

6.1 Objective Function

$$PL = \sqrt{(x_S - x_1)^2 + (y_S - y_1)^2} + \sum_{i=1}^{n} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} + \sqrt{(x_G - x_n)^2 + (y_G - y_n)^2}$$

$$F = PL + alpha * (points\ in\ obstacle)$$

Equation 2:Objective function

F is the function we are looking to minimize. It is a straightforward fitness function which is composed of Path length (PL) and count of the points in obstacle penalties multiplied by penalty constant alpha. Path length is calculated using Euclidian Norm method. $Ps(x_S, y_S)$ is the start location and $Pg(x_G, y_G)$ is the destination with $P(x_i, y_i)$ $, 0 < i < n$, being an intermediate point on the n-point path hence the the path length is the piecewise sum of path lengths between consecutive points. Number of obstacles along the path are calculated and multiplied with alpha which is taken as a big positive constant because path consisting of points lying in obstacles is regarded as infeasible path and has to be discarded.

We have the following assumptions about the environment of the robot :

- The environment can be represented in the form of a 2D matrix, each cell in the matrix can be mapped to a real location.
- All obstructions in the environment lie in the horizontal plane (xy plane) of the robot.
- The obstacles can be roughly treated as polygons.
- The size of the robot can be taken equal to size of one cell.

The first task is to determine how to represent an individual. The choice is shaped by the nature of the variables involved in the objective function in question. For our problem the variables are path coordinates in a 2-Dimensional space. Any robotic path can be represented as set of points on map. We attempt to convert the robotic path into a

matrix of real numbers, first column for x coordinate and second for y, this is the phenotype notation for our genetic algorithm(GA). Real-encoded implementation of GA requires us to model the phenotypic notation into a linear array of real number, individual size being 2n, where n is the number of connecting points on path (S->P1->P2->P3->-->Pn->G).

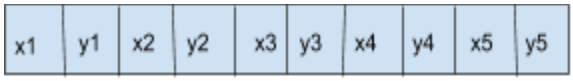


Fig.5 : Genotype for real-encoded path.

It needs to be noted that since we have a fixed individual length i.e. 2n, we have fixed the set of points forming the path consequently fixing the path complexity in terms of turns. This eliminates the production of feasible path for cases where more number of turns are required for a successful robot motion. This limitation can be overcome by considering variable sized individual. Making number of path points flexible leads to increase in computation time, hence a tradeoff has to be established.

We begin the procedure with randomly populating the individuals of the population with feasible values. Feasibility implies having path without obstacles, hence only paths having fitness values lesser than a threshold are considered. For example say for a map size of (539 x 441) we need to assign value of alpha > diagonal of map (d=715 for our case) to conclude this that fitness values as great as d are having obstacles and impose conditions accordingly.

Before calculation of fitness values, path smoothing has be conducted. This step cannot happen at a later stage as it could affect the solution feasibility. The path points are treated as control points and becomes input to the B Spline curve algorithm which returns set of points on corresponding spline at uniform intervals.

The b-spline is defined for t $\in$ $[t_0, t_m]$ as:

$$\overline{p} = \sum_{i=0}^{n} \overline{p} N_{i,k}(t)$$

Equation 3: calculating b-spline curve

Where for k=2,3,…K and for all needed values of i. The t_i are called knots and the knots $[t_i]$ are called a knot vector or knot sequence. [6]

6.2 GA Parameter tuning

The various parameters for GA have to be tuned in order to get the most optimum result.

The number of generations is directly proportional to optimality of the solution until stability or a steady state is

achieved after which no improvement is seen. The population size affects the diversity of the solutions which helps to escape local minima to some extent.

In the research paper ref.[8] by Maria Angelova & Tania Pencheva It conclude that "Different kind of modifications in SGA and MpGA are with exchanged operators' sequence of selection, crossover, and mutation operators. The impact of some of genetic algorithm parameters, namely, generation gap, crossover, and mutation rates, has been explored for all eight kinds of genetic algorithms aiming to refine the convergence time. The generation gap is the most sensitive one towards to convergence time, Among the three investigated parameters."

The results are based on the following parameter values:

Parameter	**Values**
Population size	40
Individual size	10
Crossover rate	0.8
Mutation rate	0.1
Tournament size	2
Uniform mutation width	0.5
Maximum generations	300
Minimum generations	100

Table No. 2 GA parameters

After receiving the optimal solution i.e. shortest and smoothest trajectory between the two location points we supply this to the controller program that directs the robot towards the destination from source.

A set of control points that form our robotic path are given as input to the controller program. The coordinates are taken in a matrix and the robot's position is initialised with the start point. Then a loop is applied which terminates when destination point is reached or is within a certain range from the current position of robot.

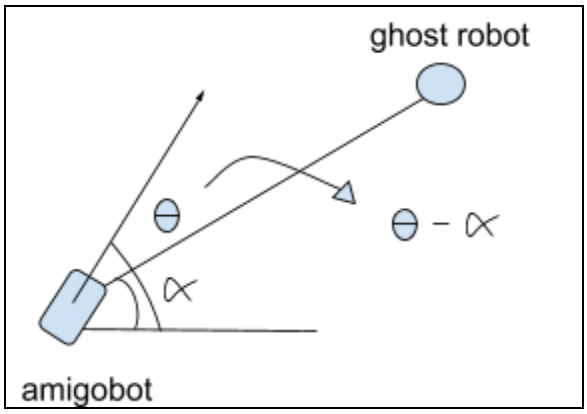


Fig.6 : Calculating crosstrack error

The following formulas have been used to calculate crosstrack error and implement P controller, refer Fig. 6

The tuned parameter values used in the program are as follows.

Parameter	**Values**
Linear velocity correction coefficient(Lc)	10.0
Rotational velocity correction coefficient (Rc)	0.5

Table No. 5 parameter values

dist	$\sqrt{((cx-px)^2+(cy-py)^2)}/100$
α - θ	$((atan2(py-cy, px-cx)*180/PI)-\theta$
Frx	$cos(\alpha * PI/180) * dist$
Fry	$sin(\alpha * PI/180) * dist$
F	$\sqrt{(Frx*Frx)+(Fry*Fry)}$
α*f*	$atan2(Fry, Frx)*180/PI;$
lvel	$abs(F)*Lc$
rotvel	$\alpha f * Rc;$

Table No. 4 Equations for crosstrack error

In the table of equations first distance between the leader robot position $P(px, py)$ and follower robot position $C(cx, cy)$ is calculated. C can be obtained from position encoder readings, ARIA provides methods for reading that. Angle to be covered (α-θ) is calculated by second equation (also ref Fig.6). Final linear error (F) and angular errors(αf) are calculated. In order to adjust values in order minimise error the Linear velocity (lvel) and angualr velocity(rotvel) proportional to the corresponding error values are supplied to the robot. The values are multiplied with a tuning coefficients whose values are assigned after testing over a set of values.

6.3 Software and Hardware Requirements

All the development work is done in C++ language, OS: Ubuntu 16.04.

Amigobot is a mobile robot with differential wheel drive. A part of the bigger family of Mobile Robots, it can be used out of the box without any requirement for hardware assembling making it easy to use for programmers. It has eight sensors for obstacle sensing and comes with a Software Development Kit having support for C++, Python, Matlab, Java. we prefer to operate AmigoBot from the computer because from computer to access the robot's functionalities is fast and easy while

working with high-level software on a familiar host computer.

6.3.1 The Development Kit

ARIA, an object-oriented, robot applications programming interface for MobileRobots written in C++ language. It allows the programmers to access the robot's functionalities at a high level and hence not requiring low level knowledge about the circuitory. A host machine is required to establish connection with the amigobot in order to communicate and supply instructions for motion and other tasks.

MobileSim – A part of the Pioneer SDK for simulating environments and MobileRobots.
It allows the programmer to work with different robot models and analyse the behaviour for various ARIA programs.

SONARNL is a C++ library for the purpose of navigation in a known environment or positioning of robot. The sonar sensors are used for the purpose of figuring the location using sensor data.

Mapper3 - An application to create maps to be used along with MobileSim for simulating various environments.

Evolving Objects (EO)

An Evolutionary Computation Framework offers EOlib which is an open source , ANSI-C++ library with class support for various evolutionary algorithms including Genetic algorithm. The implementation for any EO framework is highly customizable, the library offers control over a large variety of parameters like relative rates for crossovers and mutation. Multiple flavours of GA are present such as SGA(simple genetic algorithm). It contains tutorials for understanding the usage better.

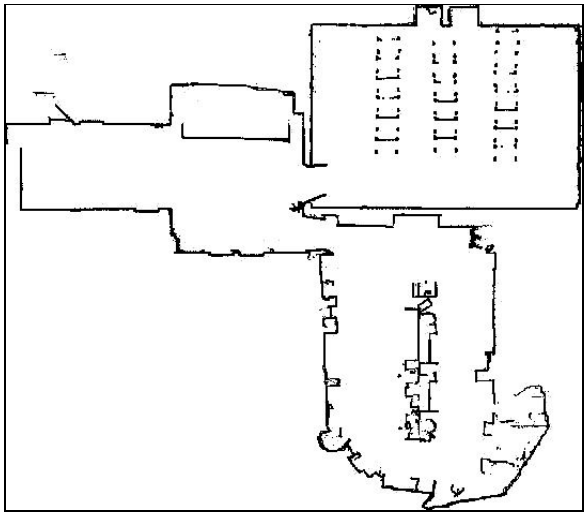


Fig.7 : Map of Robita lab, IIITA generated using SONARNL

7. RESULTS

The fig. 8-9 shows three scenarios with varying source and goal points. Outcome in all of them is a near optimal path. The Genetic Algorithm converges with optimal or near optimum fitness values. From this we can say that our choice of genetic algorithm for optimisation has been justified.

The plot shown in figure 11, is for scenario 2 between generation count and fitness value. We can see that fitness value drops significantly with initial generations but eventually becomes steady and converges.
Another plot in figure 11 shows that the evaluation count increases with generation count. Hence it has a direct proportional relationship.

Table No .5 Results for scenarios

Scenario	Best Fitness value	Last generation number
1.	434.59	100
2.	478.47	120
3.	351.656	50

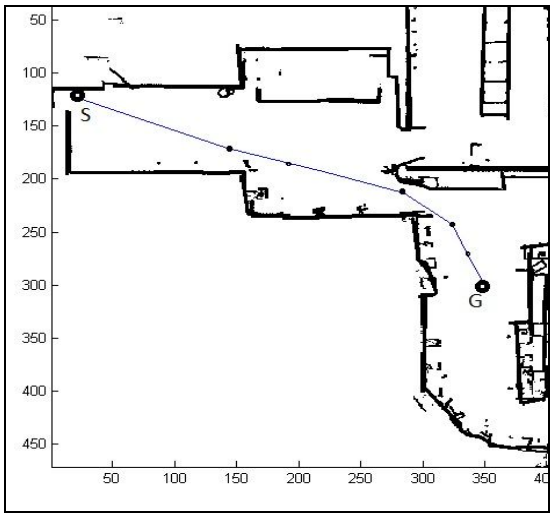


Fig.8 Scenario 1

Scenario 1 start (125,25) and goal (300,350)

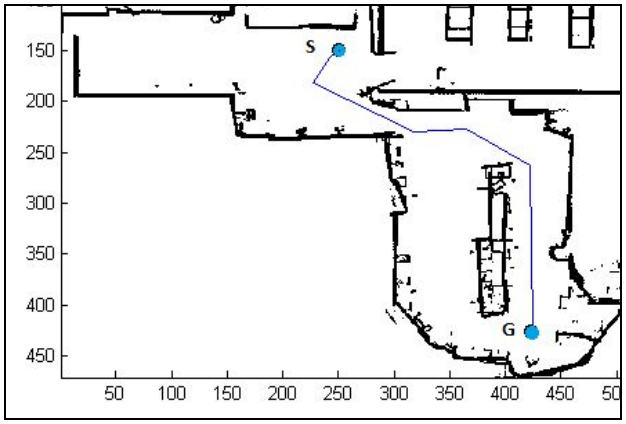


Fig.9 : Scenario 2

Scenario 2 start (125,250) and goal(425,425)

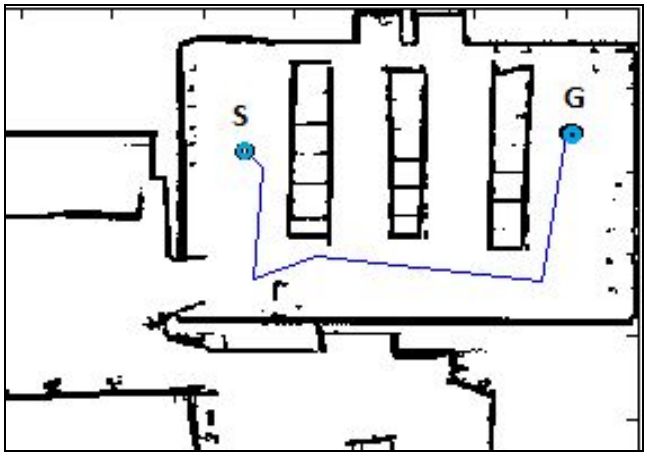


Fig. 10 : Scenario 3

Scenario 3 start (90,325) and goal (75,500)

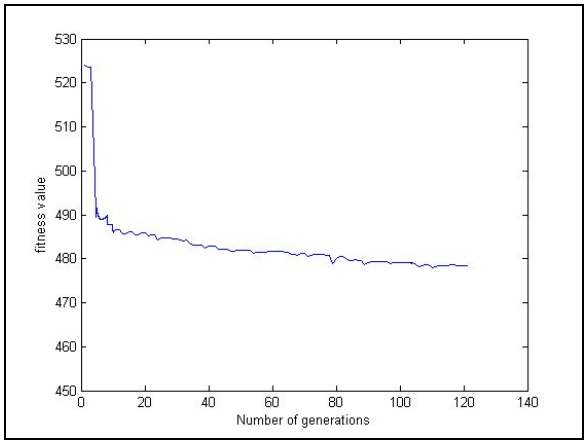


Fig 11 : Number of generations(x) vs Fitness value(y) for scenario 2

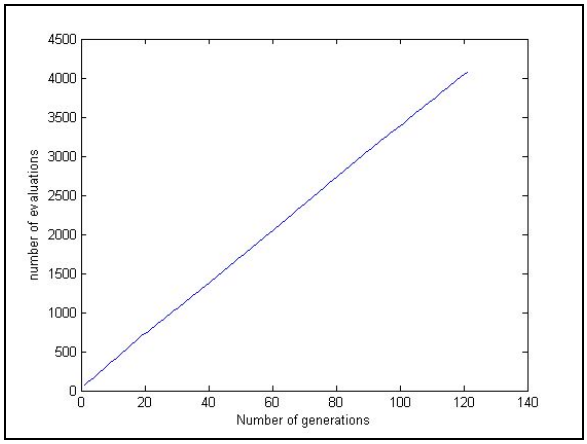


Fig 12 : Number of generations(x) vs evaluation count(y) for scenario 2

8. CONCLUSION

This report discusses a method to apply genetic algorithm for the motion planning for mobile robot. We first consider the problem of motion planning without any non-holonomic constraints. Next we work towards achieving the non-holonomic constraints most importantly the steering angle constraint we solved this using B-spline curve. After applying this approach the path is optimal as well as smooth.

In the environment in which there is set of already known obstacles, the Genetic Algorithm has proved to find the near-optimal path effectively within 100-150 generations. Upon increasing the no. of generation we can move towards achieving better solution till convergence occurs. The path we found by the algorithm, satisfies the optimisation criteria like the cost of path

, smoothness to greatest extent. We can conclude by saying that, motion planning of robots is one of the In-depth research area. In future we shall extend this methodology in dynamic Environment Scenario to prove its effectiveness and correctness.

9. REFERENCES

[1] Control for Mobile Robots Christopher Batten Maslab IAP Robotics Course January 7, 2005 https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-186-mobile-autonomous-systems-laboratory-january-iap-2005/lecture-notes/control.pdf

[2] Cezary Z. Janikow & Zabignew Micalniewicz "An experimental comparison of Binary and floating point representation in genetic algorithms" Pg 4-6

[3] Bhushan Mahajan & Punam Marbate "Literature Review on Path planning in Dynamic Environment" International Journal of Computer Science and Network, Vol 2, Issue 1, 2013 115 ISSN (Online)

[4] Pid Controller(Webpages) https://www.robotix.in/tutorial/avr/pid/

[5] AmigoBot(Webpages) https://www.generationrobots.com/media/AmigoGuide.pdf

[6] B-Spline Curve ,Equation (Webpages) https://en.wikipedia.org/wiki/B-spline http://soliton.ae.gatech.edu/people/jcraig/classes/ae4375/notes/b-splines-04.pdf

[7] Waghoo Parvez , Sonal Dhar , "Path planning of robot in static environment using Genetic Algorithm (GA) Technique" International Journal of Advances in Engineering & Technology, July 2013

[8] Maria Angelova & Tania Pencheva "Tuning Genetic Algorithm Parameters to Improve Convergence Time" International Journal of Chemical Engineering Volume 2011 (2011), Article ID 646917, 7 pages

[9] Intelligent Planning for Mobile Robotics: Algorithmic Approaches Chapter 4 Evolutionary Robotics 1 (pages 77-99)

[10] Toolika Arora & Yogita Gigras & Vijay Arora "Robotic Path Planning using Genetic Algorithm in Dynamic Environment"Volume 89 – No 11, March 2014

[11] 1,Er. Waghoo Parvez , 2,Er. Sonal Dhar "Path Planning Optimization Using Genetic Algorithm – A Literature