Available online at www.ifejost.org

IFE JOURNAL OF SCIENCE AND TECHNOLOGY

Vol 3. No. 1 (2019) 60–73

A Graphical User Interface for Path Planning of Mobile Robot

Z Haruna[a]*, M. B. Abdurrazaq[a], A. Umar[a], U. Musa[b]
[a]Department of Computer Engineering, Ahmadu Bello University, Zaria, Nigeria
[a]Department of Electrical Engineering, Ahmadu Bello University, Zaria, Nigeria
Correponding author: hzaharuddeen@abu.edu.ng
Tel.: +2347036895532

Abstract

This paper presents a graphical user interface (GUI) for path planning of mobile robot using a modified bat algorithm in a 2D environment containing rectangular static obstacles. The modified bat algorithm was used as the controller that guided the mobile robot based on the echolocation behaviour of bats. This enables the mobile robot to move from a source position to a target destination without colliding with the obstacles. A GUI based path planning simulation environment was developed to display the optimal collision free path generated by the controller. With the GUI software, it is easier for a user to implement path planning of mobile robot as user has opportunity to define environment variables without having knowledge of programming. The simulation result obtained shows that the modified bat algorithm path planning controller generates a collision free optimal path with minimum elapsed time. This indicates that the software is a good technique for implementation of mobile robot path planning in different scenarios of obstacles and maps size.

Keywords: *path planning; graphic user interface; modified bat algorithm; path planning controller; static obstacles*

Introduction

Mobile robots have applications ranging from the field of automated surveillance, transportation, medicine, military operation, drug design, computer animation and hazardous environment (Raja & Pugazhenthi, 2012). Thus, for mobile robots to be able to execute different tasks in these environments with minimum or no human intervention, mobile robots need to be more autonomous and intelligent. One of the most important and critical aspects of an autonomous mobile robot is path planning.

Path planning is the determination of a collision-free optimal path for a mobile robot to move from a source location to a target destination in an environment containing obstacles. However, there are several paths for a mobile robot to reach its target destination, but the optimal path is computed according to some criteria. These criteria include: shortest distance, shortest time or minimum energy consumption. The shortest distance and time are the most commonly adopted criteria (Abbas & Ali, 2014). Thus, for designing a fast and efficient procedure for navigation, path planning is an essential aspect.

The path planning problem consists of finding a sequence of moves for rearranging the robot in an environment, from a source location to a given target destination, and the mobile robot must avoid collision with the obstacles in the environment (Yun *et al.*, 2010). Mobile robot path planning is divided into global (offline) and local (online) path planning. In the former, prior knowledge of the environment and optimal collision-free paths are known by mobile robot in advance before commencing motion. While in the latter case, the knowledge of the environment is unknown to the mobile robot but the path planning approach uses sensor to extract information of the environment so as to generate a collision-free path for the mobile robot.

However, depending on the environment where the mobile robot is located, the path planning can be classified into static and dynamic environment (Reshamwala & Vinchurkar, 2013). Mobile robot path planning in a static environment contains only static obstacles in the environment; while mobile robot path planning in a dynamic environment contains both static and dynamic obstacles in the environment.

Various approaches have been developed in literature to address the path planning problem. These approaches are classified into classical and metaheuristics-based approaches. The classical approach such as cell decomposition, visibility graphs, potential field and heuristics approach suffers from the problem of local minima convergence and high computational cost. This leads researchers to the development of the metaheuristics-based approach for addressing path planning problem.

Meta-heuristic techniques are well-known global optimization methods that have been successfully applied in many real-world and complex optimization problems (Annicchiarico *et al.*, 2005; Gandomi *et al.*, 2013). These techniques attempt to mimic natural phenomena or social behaviour so as to generate better solutions

for optimization problem by using iterations and stochasticity (Talbi, 2009). Fig. 1 represents the classification of path planning approaches:

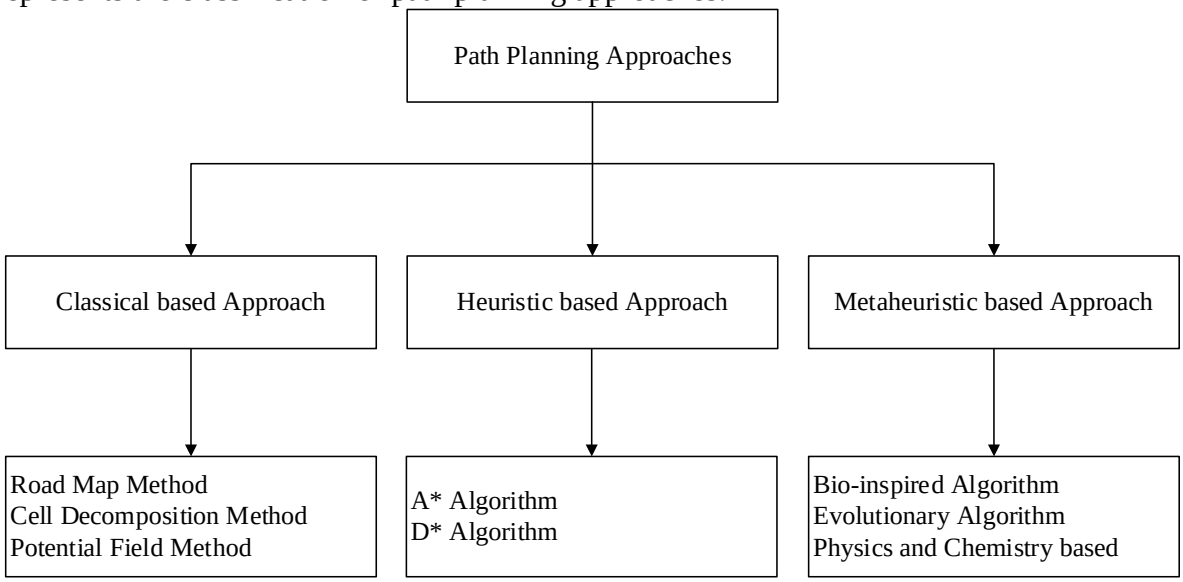

Figure 1: Classification of Path Planning Approaches

The bat algorithm is a bio-inspired metaheuristics search algorithm developed by Yang (2010) based on the echolocation behaviour of bats. The primary purpose of bats echolocation is to serve as a hunting strategy. Thus, while searching for food, bats send sound signals usually at the pulse rate of 10-20 times per second and listen for reflection sound (echo) that bounces back after striking the obstacles. For navigation, bats use the time delay between emissions and reception of sound to determine obstacle-free path. This, when applied to a mobile robot, can be used to guide its movement in an environment containing obstacles. However, the bat algorithm has a high rate of exploitation, but at times it may get trapped into some local optima, so that it may not perform exploration very well (Alihodzic & Tuba, 2014). This was the main reason why Haruna *et al.* (2017) developed a modified bat algorithm using elite opposition-based learning in order to diversify the solution search space so that the algorithm can avoid being trapped in local optima.

Therefore, this research work develops a GUI based path planning using a modified bat algorithm for easy determination of an optimal collision-free path of a mobile robot moving from a source location to a defined target destination without colliding with obstacles.

The Bat Algorithm

Yang (2010) developed a new metaheuristics search algorithm that mimics the echolocation behaviour of bats. The bat's echolocation is the process by which bats emit sound pulses at the rate of 10 to 20 pulses per second while searching for food and await the echo to bounce back from surrounding obstacles so as to generate an obstacle-free path. Thus, the bat algorithm uses the bat's echolocation to explore the solution search space efficiently during its search process. As such, bats navigate by using the time delay from emission to reflection of sound pulses. The pulse rate can be determined in the range of 0 to 1, where 0 means that there is no sound emission and 1 means that the bats are emitting sound at their maximum. In order to transform these behaviors of bats to algorithm, Yang used three idealized rules:

All bats use echolocation to sense distance, and also know the difference between food/prey and background barriers;

a. Bats fly randomly with velocity v_i at position x_i with a fixed frequency $(\mathrm{F}_{\min})$, varying wavelength λ and loudness L_0 to search for prey. They automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission $r \in [0,1]$, depending on the proximity of the target;
b. The loudness varies from a large (positive) L_0 when searching for prey to a minimum constant value $(L_{\min})$ when homing towards the prey.

The bat algorithm is implemented based on the following steps:

a. Initialize the bats parameters: the bat population, initial position (*x*), initial velocity (*v*), $\mathrm{F}_{\max}$, $\mathrm{F}_{\min}$, loudness (*L*), pulse emission rate (*r*) are defined.
b. Generate new solution by updating *F*, *v* and *x* in the equations below:

$$\mathrm{F}_i = F_{\min} + (F_{\max} - F_{\min}) * \beta(0,1) \quad (1)$$

$$v_i^n = v_i^{n-1} + (x_i^t - x_{best}) * F_i \quad (2)$$

$$x_i^n = x_i^{n-1} + v_i^n \quad (3)$$

c. Perform local search by choosing a solution among the best solution using equation (4).

$$rand > r \quad (4)$$

A local solution is then generated around the best solution using equation (5)

$$x_{new} = x_{old} + \varepsilon L^n \quad (5)$$

d. Generate new solution by flying randomly using equation (6)

$$\left(rand < L \; and \; obj\left(x_i\right) < obj\left(x_{best}\right)\right) \quad (6)$$

The modified bat algorithm

The bat algorithm is known to have high exploitation rate which leads the algorithm at times being trapped in local minimal. Hence, Haruna *et al.* (2017) developed the modified bat algorithm by incorporating elite opposition – based learning at the initialization stage of the bat algorithm so as to efficiently increase the diversity of the solution search space and at the local search space, inertia weight was introduced so as to balance exploitation and exploration during the algorithm search process.

Elite opposition based - learning is a new technique in the field of intelligence computation. Its main premise is to utilize some selected elite individuals from the current population to generate a corresponding opposite population located within the dynamic search boundaries by opposition-based learning (Zhou *et al.*, 2016). The mathematical implementation of elite opposition based learning is shown in equation (7):

$$\tilde{x} = \delta(\mathrm{p}_i + q_i) - x_e \quad (7)$$

Where $i = 1, 2, ..., \mathrm{m}$ and $j = 1, 2, ..., \mathrm{D}$, m is the population size, D is the dimension of x, $\delta \in U\left(0,1\right)$, (p_j, q_j) is the dynamic decision bound of the j^{th} decision variable and can be calculated as:

$$p_j = \min\left(x_{i,j}\right) \text{ and } q_j = \max\left(x_{i,j}\right) \quad (8)$$

The inertia weight chosen for this research is a linear value of iteration weight. The choice of this inertia weight is to ensure that the solution search space always minimizes as the algorithm moves towards the termination criteria during exploitation. Equation (9) is the linear value of inertia weight employed in the research.

$$\lambda = \left(\frac{itr_{\max} - itr}{itr_{\max}}\right)^m \quad (9)$$

The inertia weight in equation (9) is then introduced into the local search equation of the bat algorithm as shown in equation (10):

$$x_{new} = x_{old} + \varepsilon\lambda L^n \quad (10)$$

The flow chart of the modified bat algorithm is given in Fig. 2 (Haruna *et al.*, 2017)

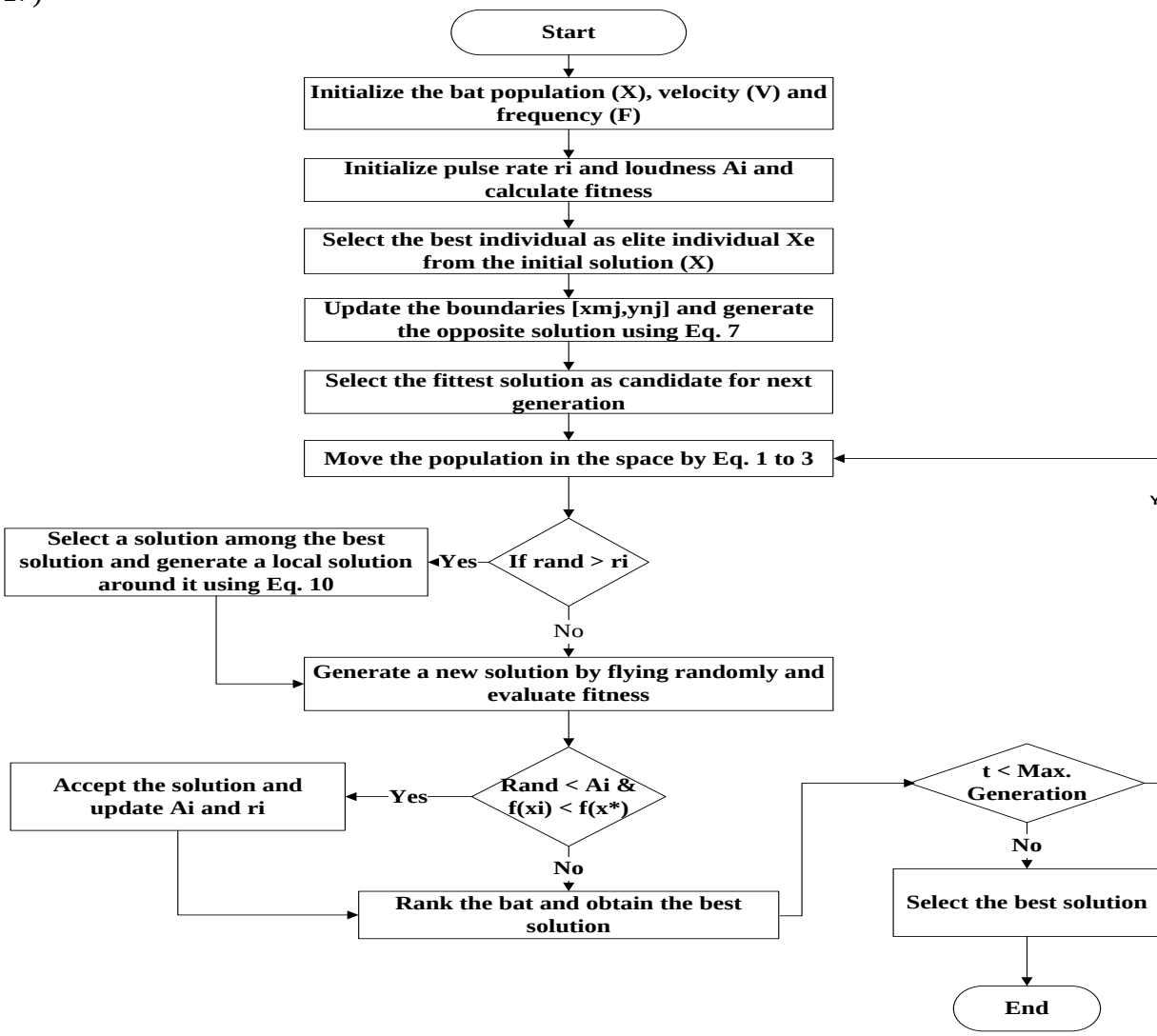

Fig. 2. Flow Chart of the Modified Bat Algorithm (Haruna *et al.*, 2017)

Path planning problem

The path planning problem is to determine an optimal collision-free path for a mobile robot to move from source position with coordinate (X_1, Y_1) to a target

destination with coordinate (X_2, Y_2) in static environment populated with obstacles. The optimality of the path is for the mobile robot to navigate to the target in a minimum time via a shortest path without colliding with the obstacles in the environment. Fig. 3 presents the developed simulation environment:

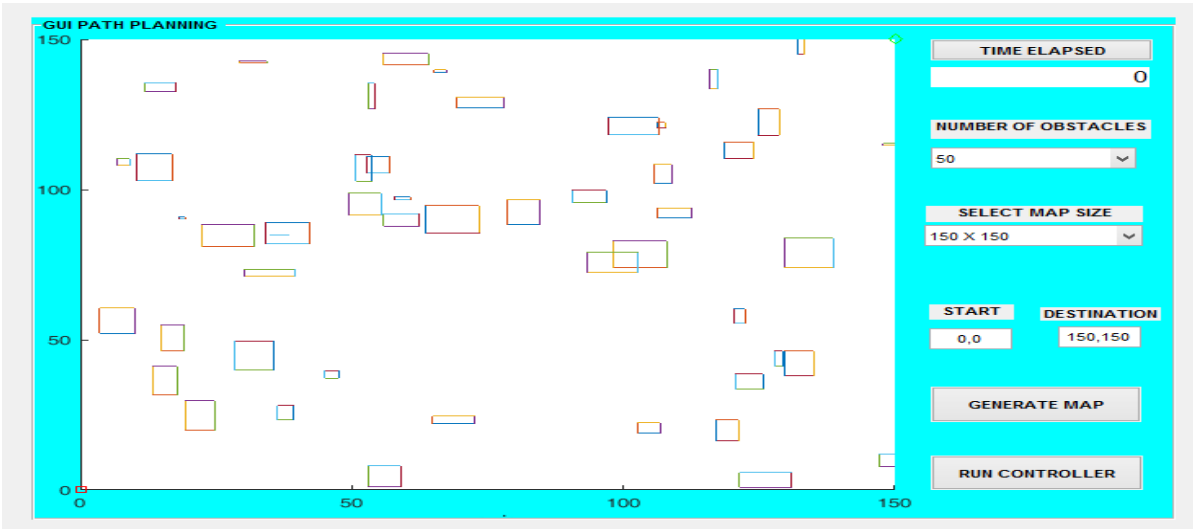

Fig. 3. GUI Based Simulation Environment Configuration

From Fig. 3, the mobile robot source position is represented using a red square with coordinate (0,0) and a target destination represented using a green diamond with coordinate (150,150) in a static environment containing 50 number of rectangular obstacles. The mission of the mobile robot is to reach the target destination from the source location via a collision free optimal path in a minimum time as possible using the developed modified bat algorithm path planning controller.

Materials and Methods

A graphical User Interface (GUI) is a graphical display in one or more windows containing components which enable a user to execute interactive tasks. GUI is visually presented to a user as a "front-end" of a software application (Patrick & Thomas, 2003). GUI components consist of a figure window containing menus, toolbars, radio buttons, push buttons, static text, sliders and list boxes which a user can manipulate interactively with the mouse and the key boards. The two main steps in creating a GUI are design of layout and writing callback functions that perform the desired operations when the user selects different features (Hunt *et al.*, 2004). The developed GUI path planning software is shown in Fig. 4:

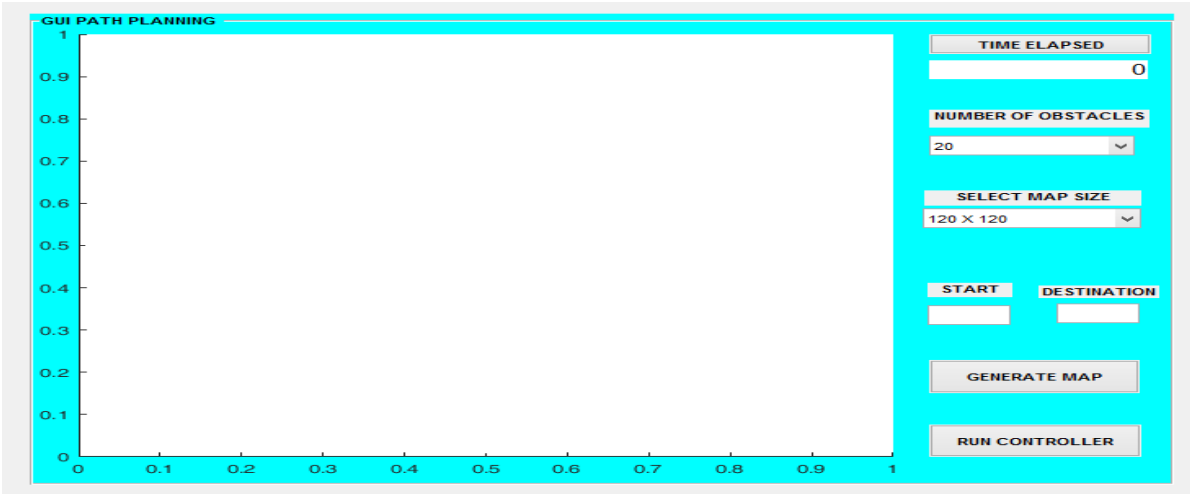

Fig. 4. Snippet of the GUI Based Path Planning Simulation Environment

From Fig. 4, the time elapsed button displays the time taken by the mobile robot to move from the source position to the target destination through a collision free optimal path. A popup menu is used for selecting the number of obstacles which contains three categories i.e. 20, 30 or 50 obstacles. Three categories of map size are defined in the select map size button for the user to choose. These include 100×100, 120×120 and 150×150 environment dimension. The software allows the user to select freely the start position and target destination. The generate map button generates the map of the environment based on the inputted parameters while the controller button guides the mobile robot so as to reach its target destination via a collision free optimal path.

The controller designed is based on the structure of the modified bat algorithm developed by (Haruna *et al.*, 2017). The objective function of the algorithm is then designed based on a distance formula in equation (11) so as to determine the distance of the mobile robot from its current position to its target destination. The modified bat algorithm (controller) optimized this objective function given in equation (11) so as to obtain a minimum path collision free path for the mobile robot.

$$dist = \sqrt{\left(X_2 - X_1\right)^2 + \left(Y_2 - Y_1\right)^2} \quad (11)$$

The direction of movement of the mobile robot is computed using equation (12)

$$direction = \tan^{-1}\left(\frac{Y_2 - Y_1}{X_2 - X_1}\right) \quad (12)$$

The mobile robot incremental movement along the horizontal and vertical direction within the obstacles environment is computed using equations (13) and (14).

$$X_{new} = X_1 + step * \cos(\text{direction}) \quad (13)$$

$$Y_{new} = Y_1 + step * \sin(\text{direction}) \quad (14)$$

The simulation of the mobile in the developed software by updating its coordinates is presented in Fig. 5:

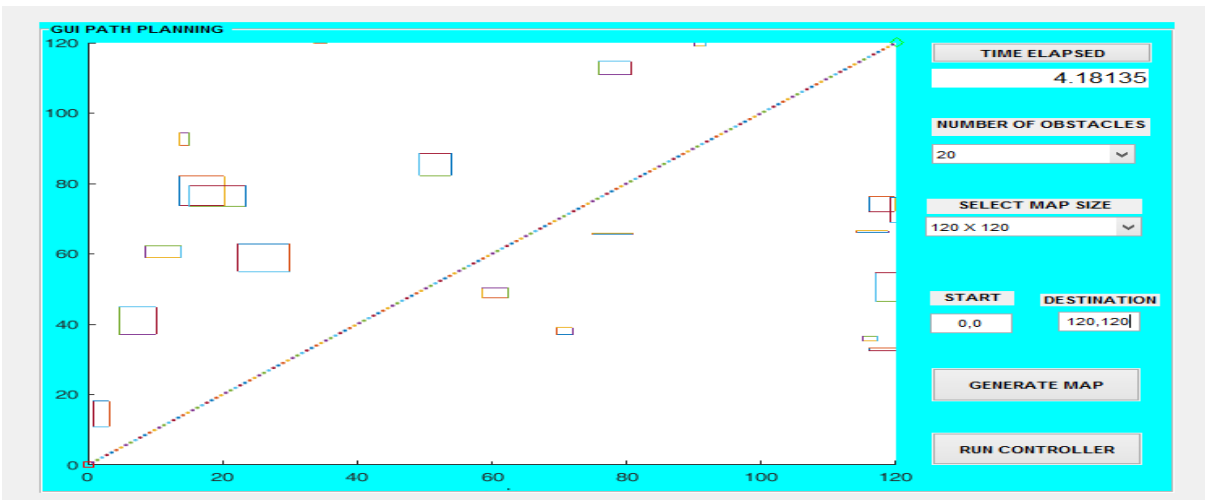

Figure 5: Mobile Robot Reaching Target without Detecting Obstacles in its Path

From Fig. 4, as the mobile robot moves towards the target from the source location without detecting obstacles along its path, the mobile robot continues to move based on equations (13) and (14) until the target destination is reached.

However, if the mobile robot detects an obstacle(s) as it moves towards the target destination during the search for optimal path, the modified bat algorithm path planning controller guides the mobile robot based on the information gathered from the obstacles environment while sending and receiving sound signal. This is shown in Fig. 6:

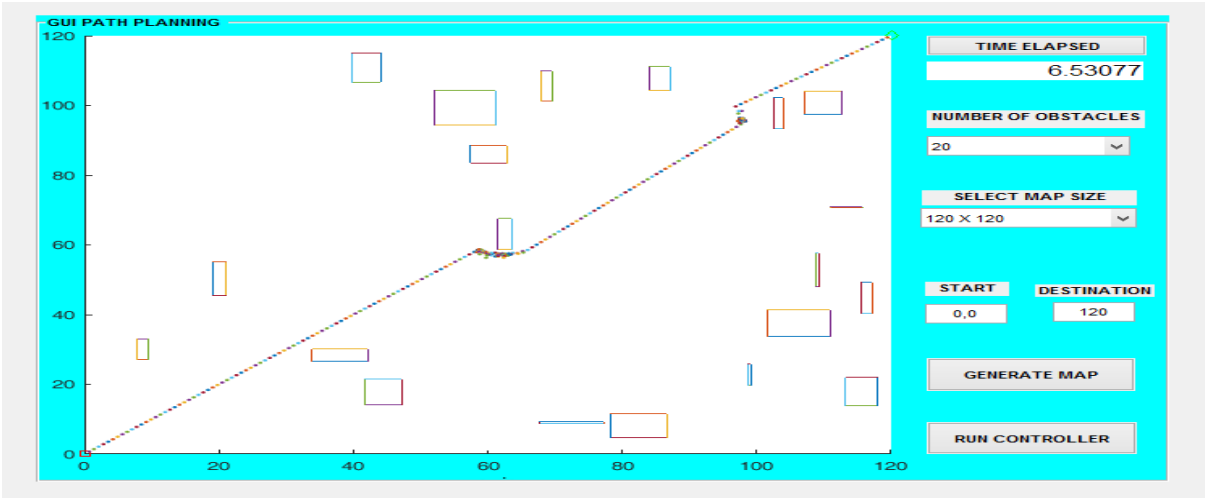

Figure 6: Mobile Robot Reaching Target while Detecting Obstacles in its Path

Results and Discussion

After development of the GUI-based path planning simulation software using MATLAB R2015a simulation environment, the modified bat algorithm path planning controller is then used to determine a collision free optimal path after generating the path planning map that contains the static obstacles, source position of the mobile robot and the target destination. The simulation results are presented in Figs 7, 8 and 9.

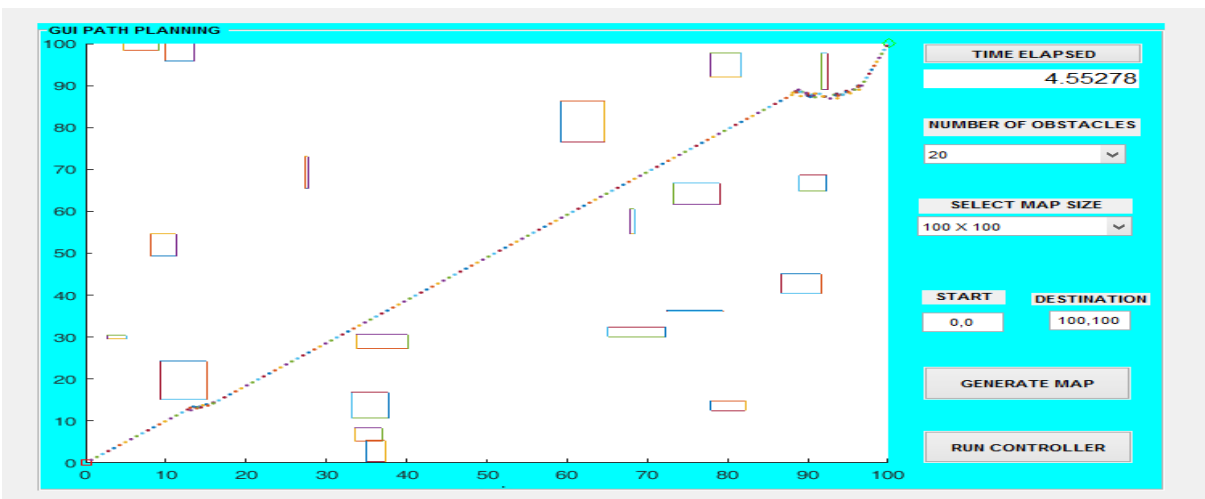

Figure 7: Mobile Robot Reaching Target in 100*100 Environment Containing 20 Obstacles

From Fig. 7, it is evident that the path followed by the mobile robot in reaching the target destination is a collision free optimal path. However, the time taken by the mobile robot in reaching the target from the source location is recorded as 4.5528 *seconds*.

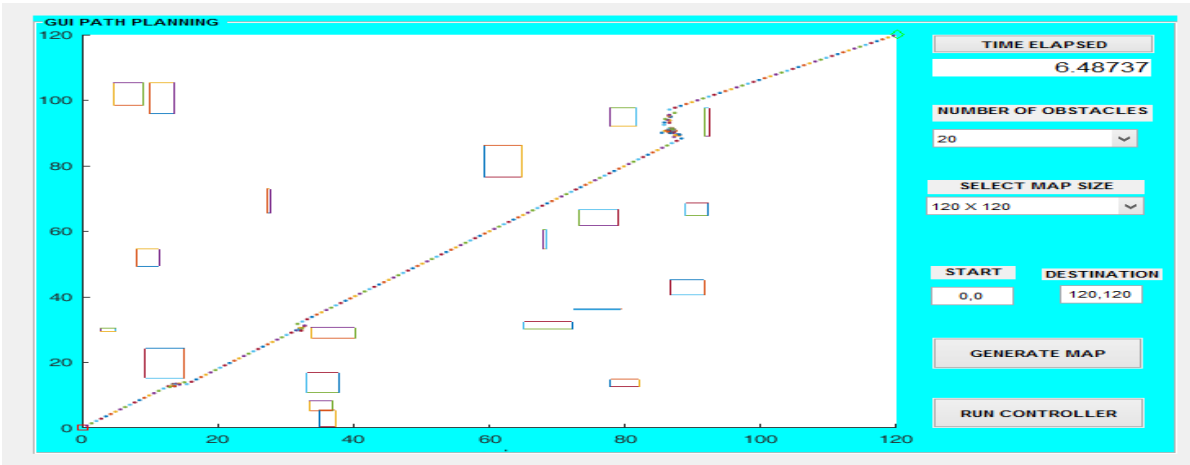

Figure 8: Mobile Robot Reaching Target in 120*120 Environment Containing 20 Obstacles

From Fig. 8, it is obvious that the path followed by the mobile robot in reaching the target destination is also a collision free optimal path. However, the time taken by the mobile robot in reaching the target destination from the source location is recorded as 6.4874 *seconds*. When compared to the time recorded in 100*100 environment, the time taken increases due to the increase in the distance between the source and the target destination.

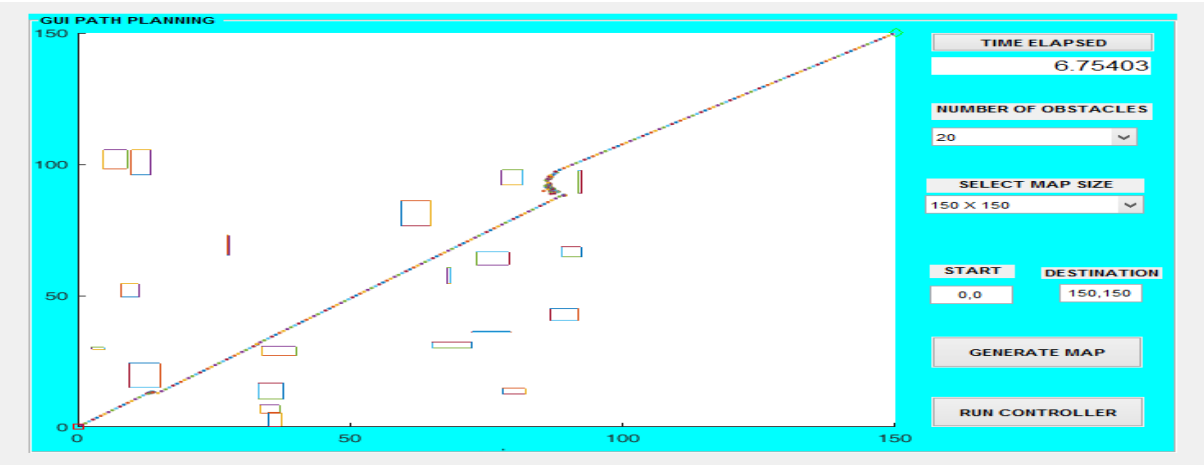

Figure 9: Mobile Robot Reaching Target in 150*150 Environment containing 20 Obstacles

From Fig. 9, it can be seen that the path followed by the mobile robot in reaching the target destination is also a collision free optimal path. However, the time taken by the mobile robot in reaching the target destination from the source location is recorded as 6.7540 *seconds*. When compared to the time recorded in 100*100 and 120*120 environments, the time taken increases due to the increases in the distances between the source and the target destination.

However, the same scenario was employed using 10 and 50 number of obstacles in the three different environments with dimensions 100*100, 120*120 and 150*150. Table 1 below shows the record of the time elapsed by the mobile robot in reaching the target destination.

Table 1: Summary of Time Elapsed for Mobile Robot to Reach the Target Destination

Number of Obstacles	Time Elapsed 100*100	Time Elapsed 120*120	Time Elapsed 150*150
10	4.2528	5.0616	5.8010
20	4.5528	6.4874	6.7540
50	7.4374	9.8904	14.2237

From the results in Table 1, it can be seen that the time elapsed by the mobile robot in reaching the target destinations increases across the rows and the columns. This is an evidence that across the rows, the distance from the source position to the target destination increases as the map size (environment dimension) increases while across the column, the complexity of the environment

increases as the number of the obstacles also increases. This showed that the modified bat algorithm path planning based controller is efficient in determining a collision free optimal path with minimum elapsed time.

However, to compare the performance of the developed path planning controller using modified bat algorithm with methods reported in literature, such as standard bat algorithm and particle swarm optimization algorithm (PSO), Table 2 was obtained.

Table 2: Summary of Time Elapsed for Mobile Robot to Reach the Target Destination using Different Algorithms

Algorithm	Time Elapsed 100*100	Time Elapsed 120*120	Time Elapsed 150*150
Modified bat algorithm	4.5528	6.4874	6.7540
Bat algorithm	9.5609	13.2716	15.6031
PSO algorithm	14.1137	19.4732	22.3571

Table 2 is the result obtained when the developed controller was implemented using three different methods (modified bat algorithm, standard bat algorithm and PSO) in an environment containing 20 number of obstacles. The results obtained show that the developed controller using the modified bat algorithm was able to obtain the best time for the mobile robot to reach its target. Gigras and Vasishth, (2015) also compare bat algorithm with PSO for mobile robot path planning and the result obtained shows that bat algorithm is more effective for mobile robot path planning.

Conclusions

A Graphical User Interface based path planning of mobile robot has been developed and a modified bat algorithm controller was used to determine a collision free optimal path for the mobile robot to move from the source location to the target destination in an environment containing different number of obstacles. With the GUI software, it is easier for a user to implement path planning of mobile robot as the user has opportunity to define environment variables without having knowledge of programming. The performance of the developed software was determined by generating three different maps with map size of 100*100, 120*120 and 150*150 each respectively containing different number of obstacles of sizes 10, 20 and 50 so as to increase the complexity of the environment in order to determine the effectiveness of the modified bat algorithm path planning controller.

The simulation result obtained shows that the modified bat algorithm path planning controller generates a collision free optimal path for larger areas or obstacles without significant increase in delay. This indicates that the software is a good technique for implementation of mobile robot path planning in different scenarios of obstacles and maps size.

References

Abbas, N. H., & Ali, F. M. (2014). Path Planning of an Autonomous Mobile Robot using Directed Artificial Bee Colony Algorithm. *International Journal of Computer Applications, 96*(11): 11 – 16.

Alihodzic, A., & Tuba, M. (2014). *Improved hybridized bat algorithm for global numerical optimization.* Paper presented at the Computer Modelling and Simulation (UKSim), 2014 UKSim-AMSS 16th International Conference on.

Annicchiarico, W., Periaux, J., Cerrolaza, M., & Winter, G. (2005). *Evolutionary algorithms and intelligent tools in engineering optimization*: Wit Pr/Computational Mechanics.

Gandomi, A. H., Yang, X.-S., Talatahari, S., & Alavi, A. H. (2013). *Metaheuristic applications in structures and infrastructures*: 32 Jamestown Road, London NWI 7BY: Elsevier.

Gigras, Y., & Vasishth, O. (2015). Comparison of BAT with PSO for Path Planning Problems. *International Journal of Engineering Development and Research,* 3(2): 590 - 595.

Haruna, Z., Mu'azu, M. B., Abubilal, K. A., & Tijani, S. A. (2017). *Development of a modified bat algorithm using elite opposition—Based learning.* Paper presented at the Electro-Technology for National Development (NIGERCON), 2017 IEEE 3rd International Conference on.

Hunt, B. R., Lipsman, R. L., Coombes, R., Osborn, J. E., & Hall, G. V. M. P. (2004). A Guide to MATLAB® for Beginners and Experienced Users.

Patrick, M., & Thomas, H. O. (2003). Graphics and GUIs with MATLAB: Chapman & Hall CRC Press Company, London.

Raja, P., & Pugazhenthi, S. (2012). Optimal path planning of mobile robots: A review. *International Journal of Physical Sciences, 7*(9): 1314-1320.

Reshamwala, A., & Vinchurkar, D. P. (2013). Robot path planning using an ant colony optimization approach: A survey. *Int. J. Adv. Res. Artificial Intelligence, 2*: 65-71.

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation* (Vol. 74): John Wiley & Sons.

Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm *Nature inspired cooperative strategies for optimization (NICSO 2010)* (pp. 65-74): Springer.

Yun, S. C., Ganapathy, V., & Chong, L. O. (2010). *Improved genetic algorithms based optimum path planning for mobile robot.* Paper presented at the Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on.

Zhou, Y., Wang, R., & Luo, Q. (2016). Elite opposition-based flower pollination algorithm. *Neurocomputing, 188*: 294-310.