EAI Endorsed Transactions on AI and Robotics Research Article **EAI.EU**

Bio-inspired BAS: Run-time Path-planning And The Control of Differential Mobile Robot

Mubashir Usman Ijaz[1], Ameer Tamoor Khan[2], Shuai Li[3] *

[1]Department of Mechanical Engineering, Pakistan Institute of Engineering and Applied Sciences, Pakistan
[2]Department of Computing, The Hong Kong Polytechnic University, Hong Kong
[3]Department of Electronic and Electrical Engineering, Swansea University, UK

Abstract

Trajectory tracking and obstacle avoidance lies at the heart of autonomous navigation for mobile robots. In this paper, a control architecture for trajectory tracking while avoiding obstacles and controller tuning is proposed for a differential drive mobile robot (DMR). The framework of optimization algorithm is inspired by the food search behavior of beetles using their antennae. Path planning and controller tuning still remain computationally demanding tasks despite of the proposed algorithms existing today. Here, we propose a meta-heuristic optimization algorithm to solve these two problems by choosing appropriate objective functions. Our bio inspired approach unifies path planning and controller tuning problems by minimizing the respective cost functions and solving the optimization problem efficiently. Trajectory tracking problem is based on the difference of the current and next pose of the robot while obstacle avoidance is achieved on the principle of maximizing the minimum distance between the robot and obstacle in the path of the robot. The proposed architecture is simulated in V-REP environment using MATLAB. Simulation results have verified that beetle antennae search can successfully plan and track the reference path by tuning the PID controller efficiently.

Received on 26 April 2022; accepted on 02 June 2022; published on 10 June 2022

Keywords: Meta-heuristic optimization, Trajectory Tracking, Path planning, Obstacle avoidance, Bio-inspired algorithm

doi:10.4108/airo.v1i.656

1. Introduction

While performing navigation in previously unseen environments, robotic agents must localize themselves by perceiving their surroundings, plan a safe trajectory to reach the desired goal and control their motion along the way[1]. Path planning includes finding a shortest and an obstacle free path while motion control involves setting the joint target velocities so that the robot follows the desired path. Path planning is an intricate task for autonomous navigation of mobile robots[2] and becomes more challenging in unstructured environments with no prior knowledge. Therefore, path planning has been a hot spot in many fields including intelligent robotics research and a considerable number of research issues have been studied in this area[3–6]. In recent times various path planning algorithms have been proposed such as dijkstra, probabilistic road maps, A*, rapidly exploring random trees (RRT), genetic algorithms, and particle swarm optimization to name a few[7–10]. Despite of many proposed solutions and their eminent advantages, advanced discoveries reveal that these methods have several inherent weaknesses[11][12]. Main hurdles remain computational complexity, adaptability, time constraint, and convergence on local optima[13]. Since obstacle avoidance is quite challenging task, many approaches are proposed by different researchers to solve this problem[14]. For example, although Bug1 algorithm[15] is simplest and easy to implement however it spends excessive amount of time around the obstacle. Another approach developed by Mr. Khatib called Artificial Potential field in which robot is considered as a point and it gets repelled from obstacles[16]. The main problem with this method is that the robot can get trapped in local optima. In addition to these

*Corresponding author. Email: shuaili@ieee.org obstacle avoidance algorithms some other algorithms can be found in[17–21]. Mobile robot path planning approaches can be classified in two broad categories based on the availability of prior information. These include global planning and local planning[22, 23]. Global path planning works by planning a trajectory in an environment where position and the vertices of the obstacles are known. In contrast, local path planning is based on the current information of the robot and priori information of the obstacles is not available. A recent study undertaken by Nitin and Chinmay discusses various classes of path planning algorithms along with their variants[24]. In addition to trajectory planning, motion of the robot is required to be controlled to ensure it follows the desired path[25–27]. Various control techniques have been proposed by the scientists over the years, such as PID, optimal and model predictive control. In this paper, PID controller is adopted which is a widely used across many other industrial applications[28–30]. For proper functioning optimal parameters of the controller are needed and thus it must be tuned[31–33]. Conventionally, Zeigler and Nichols method [34] works well for wide range of applications however, it's prone to overshoot and changes in system or environment dynamics and therefore needs to be retuned. With advanced computational techniques, new methods for parameter tuning have been proposed for optimal performance[35–37]. Some of them include Genetic Algorithms (GA)[38], fuzzy systems[39], Eagle Perching Optimizer[40], Ant Colony Optimization (ACO)[41], artificial neural networks (ANN)[42], Particle Swarm Optimization (PSO)[43], bio-inspired neural networks[44] and artificial immune systems. Some of these algorithms such as PSO and ACO are nature inspired evolutionary algorithms which are based on the principle of survival of the best solutions found by communicating personal best with global best positions[45, 46]. Russel and Kennedy, in 1995, developed particle swarm optimization algorithm which mimicked the behavior of swarm populations[47]. Another such algorithm inspired by egg laying behavior of cuckoo called Cuckoo Search (CS) was developed by Yang in 2009[48]. Not all algorithms perform well with all classes of optimization problems. BAS has successfully been tested for fraud detection in trade, control of surgical robots, distributed control of robots and redundant manipulators[49–52]. In this paper we attempt to address two different constraint optimization problems[53].

- Planning an obstacle free shortest path for a differential robot to reach the goal.

- The robot should track the time varying reference trajectory based on kinematic model by tuning a PID controller.

- Non-Holonomic constraints and rotational speed limit of mechanical joints are observed.

To solve these problems, we have proposed a nature inspired beetle antennae search (BAS) algorithm. BAS is a metaheuristic optimization algorithm inspired by the food search behavior of beetles[54]. There are several advantages of using BAS over other previously mentioned algorithms such as Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO)[13].

- Algorithms such as PSO, GA, and ACO use swarm of particles. These swarms explores the search space and either one of the particle, as in PSO, or all of them, as in GA, move toward the global minimum by communicating. Whereas, in contrast, BAS is a single particle algorithm.

- As BAS is a single particle algorithm and requires less parameters to be adjusted, therefore it is computationally efficient, robust, immune to local minima and easy to implement.

- Time and space complexity of swarm algorithms is dependent on the population size and problem dimension whereas the complexity of BAS is polynomial in time [55].

- Comparison with other algorithms such as PSO, and its real-world applications such as portfolio optimization[56–58] and control of UAV for obstacle avoidance [59] shows that BAS is the state-of-the-art bio-inspired algorithm.

Our algorithm solves the two different optimization problems simultaneously which corroborates the adaptability and robustness of our algorithm for different optimization problems. Our main contributions in this paper are as follows.

- Our proposed algorithm successfully plans a smooth, obstacle free trajectory to reach the goal and controls the robot motion to track the reference trajectory.

- Objective function we used minimizes the tracking error while the penalty function used in our algorithm rewards the optimizer, rather than acting passively, for avoiding obstacle by maximizing the minimum distance between the robot and the obstacle.

- We tested our algorithm on Pioneer-P3Dx platform provided in the V-rep environment and simulation shows the successful implementation as the robot is able to reach the goal while tracking the reference trajectory.

The following table gives the abbreviations and nomenclature adopted throughout the paper:

DMR	Differential Mobile robot
BAS	Beetle Antennae Search
PSO	Particle Swarm Optimization
NP	Non-deterministic polynomial time

1.1. Kinematics and unicycle model of the DMR

A differential drive mobile robot consists of two wheels which are independently driven of each other and are mounted on the common axis which is perpendicular to the orientation of the robot. It is worth mentioning that here we have considered only kinematic model and actual inputs, i.e force and torque, to the wheels are ignored to avoid dynamics of the system [60]. For this, it is assumed that motors are powerful enough to realize V and ω instantaneously. The position of robot can be controlled by changing the velocities of the two wheels. When one of the wheels turns slower than the other, DMR turns in that direction. The robot model which treats the robot as rigid body and relates left and right wheel velocities to the position and orientation is given by the following equations.

$$x^{'} = \frac{R}{2}\,(v_r + v_l)\,cos\,(\theta) \tag{1}$$

$$y^{'} = \frac{R}{2}\,(v_r + v_l)\,sin\,(\theta) \tag{2}$$

$$\theta = \frac{R}{L}\,(v_r - v_l) \tag{3}$$

The figure 1 shows the kinematic model of differential drive mobile robot.

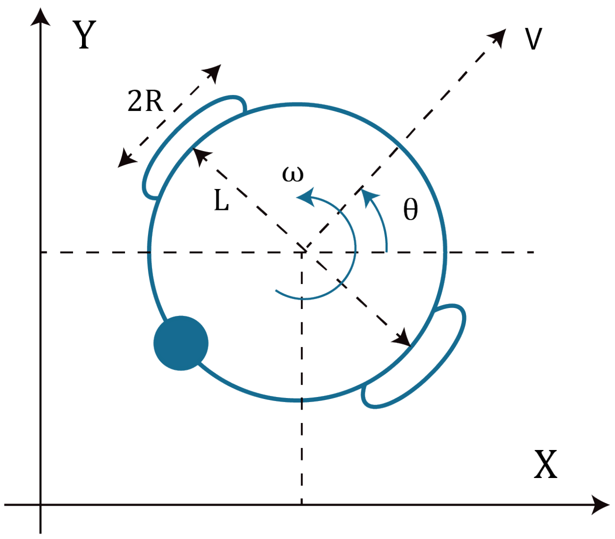

Figure 1. Kinematic model of the DMR

Odometry is used to determine the robot's state in the environment i.e pose of the robot[61]. Wheel encoders keep track of the revolutions and give the information of how far each wheel has travelled for a short time scale. When a robot changes its pose, its new position and orientation (x', y', θ) are needed to be determined from its previous pose as measured from the center of the robot. Let's consider the left and right wheels of the robot have moved a distance of d_l and d_r from its initial pose, then the distance moved by the center d_c of robot can be computed as:

$$d_c = \frac{d_l + d_r}{2} \tag{4}$$

$$\Delta\theta = \frac{d_r - d_l}{2} \tag{5}$$

$$\Delta\theta^{'} = \theta - \Delta\theta \tag{6}$$

$$x^{'} = x + d_c\,cos(\theta) \tag{7}$$

$$y^{'} = y + d_c\,sin(\theta) \tag{8}$$

Here we have adopted unicycle model of the robot for simplicity which is based on the particle motion and takes into account only forward velocity and direction of motion of our robot. Here velocity and orientation are related to the position of the robot in world frame as:

$$\begin{bmatrix} x^{'} \\ y^{'} \\ \theta^{'} \end{bmatrix} = \begin{bmatrix} cos(\theta(t)) & 0 \\ sin(\theta(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V(t) \\ \omega(t) \end{bmatrix} \tag{9}$$

Jacobian matrix defines the relation between forward and rotational velocities of the unicycle model to the left and right wheel velocities of the actual robot. If **p** is the pose of the robot and **q** vector contains the wheel velocities than unicycle velocity vector is given by:

$$V_u = \frac{\partial p}{\partial q}\frac{\partial q}{\partial t} = J\frac{\partial q}{\partial t} \tag{10}$$

In the above equation J represents the Jacobian matrix and V_u contains forward, lateral, and rotational velocities of the unicycle model. This equation describes the relationship between forward velocity V and rotational velocities. From this equation it is evident that the robot's forward velocity V and rotational velocity ω are equal to the product of Jacobian matrix with left and right wheel's velocities. To meet the non-holonomic constraint of the robot, there is no motion in lateral direction of the wheels. Thus, forward and rotational velocities are given as:

$$V = \frac{R}{2}(v_l + v_r) \tag{11}$$

$$\omega = \frac{R}{L}(v_r - v_l) \tag{12}$$

Two components of the robot velocity given by the above equations in body attached frame are related to left and right wheel velocities by J matrix as:

$$\begin{bmatrix} V \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{R}{2} & \frac{R}{2} \\ \frac{R}{L} & \frac{-R}{L} \end{bmatrix} \begin{bmatrix} v_r \\ v_l \end{bmatrix} \tag{13}$$

Hence

$$J = \begin{bmatrix} \frac{R}{2} & \frac{R}{2} \\ \frac{R}{L} & -\frac{R}{L} \end{bmatrix} \tag{14}$$

Also, we have a maximum rotational speed $v_m ax$ that can be achieved from mechanical actuators of the DMR. The speed commands to left and right motor should be less than this value to avoid any mechanical failure. This can be written as:

$$v_r < v_{max}, \quad v_l < v_{max} \tag{15}$$

Wheel radius and distance between the wheels of the DMR is given as:

Parameters	Value
R	$2cm$
L	$33cm$

The remainder of the paper is structured as follows. Section-2 explains the kinematics of the robotic platform and goes on to formulate the objective functions for the optimization problems. In section-3 we discuss our proposed nature inspired algorithm along with its implementation to solve the problem. In section 4, we employed our algorithm to simulate it on the test platform and discussed the results. Finally in section-5 we conclude our article with final remarks.

2. Problem formulation

Next, we will discuss the trajectory tracking, obstacle avoidance and controller tuning problem with their objective functions.

2.1. Trajectory Tracking

To reach the desired goal, DMR must follow a reference trajectory as calculated by beetle antennae search algorithm. For the robot to follow a desired path in its environment, its cartesian space coordinates should change as per the reference trajectory. If the pose of the robot in global co-ordinates is given by the vector $\mathbf{p}(t) = [x, y, \theta]^T$ and reference trajectory is given by $\mathbf{p_{ref}}(t) = [x, y, \theta]^T$, than the error between the actual and reference pose is calculated as $h_t(t) = \mathbf{p_{ref}(t)} - \mathbf{p(t)}$. As the states of the robot evolves with time, the trajectory tracking problem is formulated as an optimization problem with the goal to minimize the error distance between the actual pose and reference trajectory. Thus, the optimization problem is formulated as:

$$min\ \mathbf{h_t}(\mathbf{p(t)}, \mathbf{p_{ref}(t)}) \tag{16}$$

While tracking the reference trajectory, objective function $\mathbf{h_t}(.)$ minimizes the error, which is given as:

$$\mathbf{h_t}(\mathbf{p(t)}, \mathbf{p_{ref}(t)}) = \|\mathbf{p_{ref}(t)} - \mathbf{p(t)}\|_2 \tag{17}$$

2.2. Obstacle avoidance

Often robots have to perform in an unseen environment which contain static and moving objects. Avoiding these obstacles while maintaining an optimal path is a crucial task for safe operation of the robot in its environment. The trajectory tracking problem does not incorporate obstacle avoidance and thus it needs to be addressed separately. Obstacle avoidance problem is based on the principle of maximizing the minimum distance between the robot and the obstacle[51]. This leaves us with another optimization problem which is modelled by the following equation:

$$min\ \mathbf{h_o}(Obs, \mathbf{p(t)}) \tag{18}$$

In the above equation Obs is the obstacle defined by $Obs \in \mathbb{R}^{1\times2}$ and robot position is given by $\mathbf{p}$. Here we have adopted an efficient approach of Gilbert-Johnson-Keerthi (GJK) algorithm which calculates the minimum distance between the obstacle and robot and is given as:

$$dis(Obs, \mathbf{p(t)}) = GJK(Obs, \mathbf{p(t)}) \tag{19}$$

GJK algorithm works by computing the minimum distance between the closest vertices of the arbitrary shaped convex objects. It can be easily extended to handle concave shapes formed by combining different convex polygons. Now we are able to formulate our optimization problem which is based on maximizing the minimum distance between the DMR and the obstacle, concretely:

$$\mathbf{h_o}\,(Obs, \mathbf{p(t)}) = \frac{1}{min\ dis(Obs, \mathbf{p(t)})} \tag{20}$$

The denominator of the above equation calculates the minimum distance, and its reciprocal gives the maximum value. Here DMR is considered as a point object, and this assumption leads to an additional constraint which requires us to incorporate a distance greater than the base radius b of the robot.

$$GJK(Obs, \mathbf{p(t)}) > b \tag{21}$$

Thus, the final optimization problem for the obstacle avoidance is a combination of maximizing of the minimum distance and base radius of the robot and is now defined as:

$$\mathbf{h_o} = \frac{1}{min\ dis(Obs, \mathbf{p(t)})} s.t\ GJK(Obs, \mathbf{p(t)}) > b \tag{22}$$

Now we are in a position to concretely formulate our optimization problem.

2.3. Overall objective function

In the previous sections, we have separately defined our objective functions for tracking a reference trajectory and avoiding obstacles. Now before we move on to our overall objective function we have joint angle limits constraint associated with the mechanical actuators of the robot. Motor joints are only capable to work between upper and lower limits of certain joint angles. These limits are dependent on the design of actuators and motor joints must operate between these limits. This leaves us with joint angles limits constraint and it can be stated as:

$$\theta^- < \theta(t) < \theta^+ \tag{23}$$

Now we are in a position to combine these objective functions to formulate our overall optimization problem with objective function $\mathbf{h(.)}$ which is given as:

$$min\ \mathbf{h}\ (\mathbf{h_t(t)}, \mathbf{h_o(t)}) \tag{24}$$

Essentially, our goal is to minimize the following objective function while constraints $GJK(Obs, \mathbf{p(t)}) > b$ and $\theta^- < \theta(t) < \theta^+$ are satisfied:

$$min\ \mathbf{h}\ (\mathbf{h_t(t)}, \mathbf{h_o(t)}) = \beta_1 \|\mathbf{p_{ref}(t)} - \mathbf{p(t)}\|_2 + \frac{\beta_2}{dis(Obs, \mathbf{p(t)})} \tag{25}$$

In the above objective function β_1 and β_2 are the respective weights that influence the impact for trajectory tracking and obstacle avoidance outputs. Values of the β_1 and β_2 constants are paramount to the proper functioning of the algorithm. With that, we have now our fully developed optimization problem. The optimization algorithm presented next will be able to generate an optimal obstacle free path by minimizing this objective function.

2.4. Controller loss function

Manual tuning of the PID controller is painstaking and a time-consuming task. A widely used Ziegler–Nichols's method can sometimes result in larger overshoots and is not an optimal approach. With accelerating computational power in recent times, several evolutionary algorithms such as particle swarm optimization and genetic algorithms have been used to achieve optimal controller performance. Here we will use Beetle Antennae Search, an intelligent metaheuristic optimization algorithm to find the controller gains. Controller loss function, also referred as performance index, determines how far the error is from the set point and thus gives a quantitative measure of how well the controller is tuned. Here are four commonly used performance indices [43].

$$IAE = \int_0^\infty |e(t)|\, dt \tag{26}$$

$$ITAE = \int_0^\infty t|e(t)|\, dt \tag{27}$$

$$ISE = \int_0^\infty (e(t))^2\ dt \tag{28}$$

$$ITSE = \int_0^\infty t\,(e(t))^2\ dt \tag{29}$$

Integral absolute error simply sums up the areas which are above and below the set point. In case of penalizing the errors at later stage IAE is multiplied with time so that its integral becomes larger with time. In this paper square of the error is integrated over time to punish the larger errors. Integral time squared error ($ITSE$) function is used to penalize both larger errors and errors at the later stage. With ISE performance index, BAS tries to seek gains for proportional, integral, and derivative terms so that the loss function is minimized. Thus, using this performance index, PID gains are adjusted to reach the desired velocity.

3. Proposed Algorithm

In this section, we will formulate the framework of BAS and then we use it for solving the path planning problem for DMR. Finally, we use BAS to tune PID controller automatically. Traditional optimization algorithms such as gradient descent work well for continuous functions only and are also computationally expensive. Also, these algorithms can converge on a local minimum and therefore can't be used for non-convex problems. Here we will use Beetle Antennae Search algorithm (BAS) to solve our optimization problem. BAS is a nature inspired metaheuristic optimization algorithm based on the food search behavior of the beetles. Unlike particle swarm optimization that uses swarms of particles to converge to the global optimum BAS stems from the foraging behavior of the beetles. By using its two antennae beetles sense the smell intensity and move in the direction of higher intensity which helps them locate their food. Inspired by this incredible sense of smell of beetles, BAS works by randomly initializing a search direction and calculating the difference of objective function value. At this stage, it's decided in which direction value of the objective function improves and a new direction vector is estimated. This process is repeated until the goal is reached. The main loop of the algorithm terminates if the goal is reached, or the maximum allowed number of iterations are reached. Next we present a step by step procedure of solving an optimization problem using BAS.

Algorithm 1 Beetle Antennae Search

```
1: Input: Given an Objective function $min\ f(x)$
2: Initialize:
3: $T \leftarrow 100$;          % Number of iterations
4: $l \leftarrow 0.3$;           % Step-size
5: $\delta \leftarrow 0.99$;          % Step controlling factor
6: $d_o \leftarrow 0.9$;           %Antennae length
7: Initial random vector $\vec{b}$
8: Output: $\vec{x_{best}}, \vec{f_{best}}$
9: For $t = 1 \leftarrow T$
10:     Generate random vector $\vec{b} = [0, 1]$
11:     Compute the objective function $f(x)$.
12:     if $f(x) < f_{best}$
13:         $f_{best} = f(x)$
14:         $x_{best} = x$
15:     end if
16:     Move in the estimated direction
17:     Return $\vec{x_{best}}, \vec{f_{best}}$
18: end For
```

3.1. Path Planning

Other than perception and motion control problems, path planning is of paramount importance, and it has been a point of discussion for quite a while in academia particularly in the field of autonomous navigation of mobile robots. Since path planning is a non-deterministic polynomial time (NP) hard problem and its complexity depends on the degrees of freedom of the system. A good path planning algorithm must ensure the following four conditions. If exists, it should always be able to find an optimal path in static environment. The algorithm should be expandable to find path in dynamic environments. Additionally, it should minimize storage requirements and thus be computationally inexpensive. In this section, we will use Beetle Antennae Search algorithm to make the robot reach its goal without colliding into obstacles and reduce computational burden on CPU.

A path generated must meet the conditions of completeness and optimality to successfully solve the path planning problem. By completeness it's meant that if there exists a path between two points than the algorithm must be able to generate a solution. Optimality constraint ensures that the path generated should have minimum distance while avoiding any obstacles. Obviously, path doesn't need to be optimal if any one of the objective constraints are compromised.

As with any optimization algorithm, hyper-parameter tuning remains imperative for successful implementation of BAS. Here we tried several values for step size l, step controlling factor δ, antenae length d_o and number of iterations T before using them. The optimization of these hyper-parameters is a separate optimization problem which is not the main focus here. Initial values for the robot's X and Y co-ordinates are randomly initialized inside the 2D space $S(S = \mathbb{R}^2)$ and obstacles lie within this space S such that $Obs \in S$. It is also assumed that the obstacle and goal position remain fixed during the search. The goal position is provided to the algorithm as $X_g \in \mathbb{R}^{1\times2}$ and BAS starts to explore the space. We have modelled the DMR and obstacles as a point objects with a safety radius which is an added constraint to our overall objective function as shown in the previous section. With the current and goal positions known, BAS samples through the workspace S and generates shortest trajectory along with ensuring the obstacle avoidance condition is satisfied.

On each iteration, BAS performs random sampling, and left and right direction vectors are estimated to find the objective function value at that point. In each next step, an if condition is checked to find if the current estimated vector decreases the value of the objective function, such that $f < f_{min}$. Values of x_{best} vector are only updated when if condition is satisfied. Finally main loop is terminated either if the goal point is reached or if the maximum number of iterations are reached. The vector x_{best} represents the set of optimal or near optimal waypoints and is given as input to the controller.

3.2. Controller Tuning

PID controllers are extensively employed throughout various industries and therefore choosing optimal gains for proportional, integral, and derivative terms remains a point of interest. Conventional methods of PID tuning are either iterative or require algorithms that are computationally expensive and consequently the significance of controller tuning using simpler algorithm can't be condoned. Here, controller receives the error between the desired and current heading direction of the robot. This error is transformed to rotational velocity required to decrease the error as the output signal of the controller. The error is only computed if the distance of the robot is greater than the threshold distance value d_g.

$$\omega = K_p\, e + K_i \int_0^t e\, dt \tag{30}$$

For the robot to follow the desired trajectory, K_p and K_i controller gains must be chosen carefully. Here Beetle Antennae Search algorithm is used to search for these gains using the ISE performance index as an objective function. The range of the PID parameters is set between 0 to 100 and BAS yields the values for proportional and integral gains that minimize the cost function. The update rule for x_{best} variable remains the same as in case of path planning which ensures the value of output variable is only updated if the x_{new} is better than the previous value and is kept same if it increases the value of objective function. Next, we present the BAS implementation for path planning and controller tuning for our robotic platform.

Algorithm 2 BAS Path Planning and Controller Tuning

```
1:  Input: Given the Objective functions
2:          $\mathbf{h}\ (\mathbf{h_t(t)}, \mathbf{h_o(t)}),\ \int_0^\infty (e(t))^2\ dt$
2:  Initialize:
3:  $T \leftarrow 100$;        % Number of iterations
4:  $l \leftarrow 0.3$;         % Step-size
5:  $\delta \leftarrow 0.99$;       % Step controlling factor
6:  $d_o \leftarrow 0.9$;        %Antennae length
7:  Start position $\mathbf{p} = \mathbf{x}$
8:  Goal position $\mathbf{p_g}$
9:  Obstacle defined by $Obs$
8:  Output: $\vec{x_{best}}, \vec{f_{best}}$
9:  For $t = 1 \leftarrow T$
10:     Generate random vector $\vec{b} = [0,1]$
11:     Direction Vector $x^r, x^l = x \mp l{\times}\vec{b}$
12 :     $x_{new} = x + d_o sign(f(x_l) - f(x_r))\vec{b}$
13 :     if $f(x) < f_{best}$
13 :         $f_{best} = f(x)$
14:         $x_{best} = x$
15:     end if
16:     Move in the estimated direction
17:     Return $\vec{x_{best}}, \vec{f_{best}}$ where $\vec{x_{best}} = \vec{p_{ref}}$
18: end For
19: For $k = 1 \leftarrow size(x_{best})$
20:     if $d < d_g$
21:         break
22:     else
23:         Calculate desired direction $\theta_d$
24:         Heading direction error e
25:         Generate random PID gains
26:         For $j = 1 \leftarrow T$
27:             Generate random vector $\vec{b} = [0,1]$
28:             Direction Vector $x^r, x^l = x \mp l{\times}\vec{b}$
29 :             $x_{new} = x + d_o sign(f(x_l) - f(x_r))\vec{b}$
30 :             if $f(x) < f_{best}$
31 :                 $f_{best} = f(x)$
32:                 $x_{best} = x$
33:             end if
34:             Return $\vec{x_{best}}, \vec{f_{best}}$ where $\vec{x_{best}} = K_p, K_i$
35:         end For
36:     Calculate $\omega$
37:     Calculate $[v_r, v_l]$
38: end For
```

4. Simulation and Results

We used our algorithm, written in MATLAB, on Pioneer-P3Dx platform provided in the virtual robotic simulator V-REP environment. V-REP provides the virtual environment along with handling forward and inverse kinematics of the robot using "ODE" physics engine. These settings offer real world constraints to test our algorithm. Remote API functions are used for communication between MATLAB and the V-REP environment. The hyperparameters, antennae length, step size, number of iterations and step controlling factor are defined in the MATLAB. Schematically, figure 2 represents the integration of MATLAB with V-REP along with different components of the simulation below.

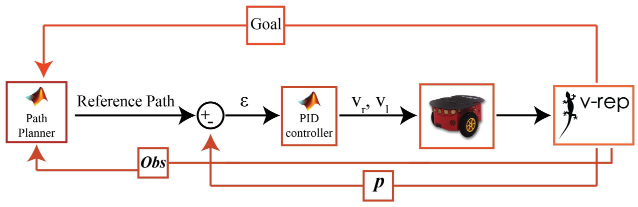

Figure 2. Schematics of path planning and control framework

The novelty of this work is the adaptability of our algorithm to solve two different optimization problems with bare minimum change of objective functions. It is evident that PID tuning, and path planning are two separate problems with different constraints and our proposed algorithm successfully solves these different optimization problems. In case of path planning, first we attempted to achieve the go-to-goal objective without any obstacle along the way and after that obstacle was introduced in the robot's path to solve for the complete path planning problem. Automatic tuning of PID controller is achieved while simultaneously solving the path planning problem. Here we will discuss results for both approaches in which Pioneer-P3Dx navigates towards the goal. The following figure 3 shows the simulation of Pioneer-P3Dx while navigating towards the goal in the V-rep environment.

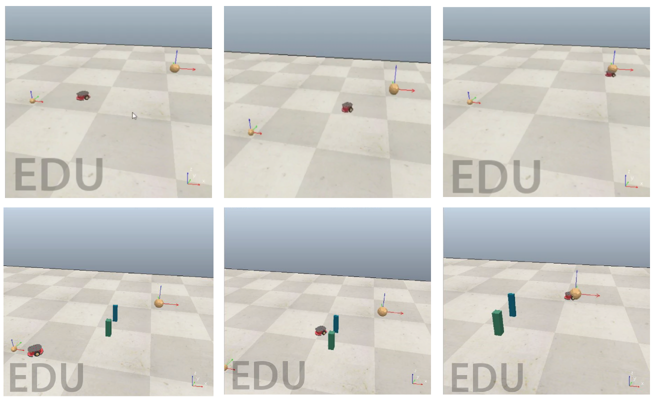

Figure 3. Trajectory tracking in VREP environment

In path planning part, a set of waypoints is generated corresponding to the optimal path for driving the Pioneer-P3Dx to its goal position. First go-to-goal objective is achieved without any obstacles in the way. Here the start and goal positions are given by $\mathbf{p} = [0, 0]$ and $\mathbf{p_g} = [5, 5]$. These waypoints for the planned path are plotted in excel which are depicted by the figure X. After that, an obstacle is introduced along the way and then the path planning problem is solved for the same start and goal positions. The following graphs in figure 4 shows the way points generated by the BAS from origin to the goal position in case of when there is no obstacle along the way and when an obstacle is introduced respectively.

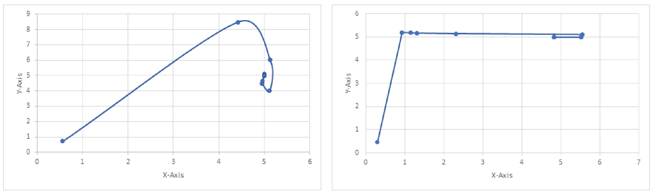

Figure 4. Solution of Go to Goal optimization in an obstacle free [Left] and with obstacles along the way [Right]

In case of controller tuning, at each iteration, the desired and current heading direction is calculated based on the way points generated and the current position as retrieved from V-rep. PI controller adjusts the heading direction by varying the angular velocity ω of the tires. Since the linear velocity is kept constant therefore the robot keeps moving towards the target until the distance between the robot and goal is less than d_g. The conventional tuning methods such as Ziegler-Nichols's method result in undesirable settling times along with larger overshoots. The below table shows the controller gains obtained with manual tuning along with optimized BAS-PI parameters.

5. Conclusion

This work presents a solution to path planning and controller tuning problems for a differential mobile robot. Our method formulates two optimization problems by carefully choosing objective functions for trajectory tracking, obstacle avoidance and PID controller. A commonly used bio-inspired algorithms such as Particle Swarm Optimization uses a swarm of particles which are time consuming and computationally expensive. We used a metaheuristic algorithm called Beetle Antennae Search based on random search method to solve these two optimization problems. We tested our approach to reach a goal by avoiding obstacles and simulated on Pioneer-P3Dx platform provided in V-rep environment. Our approach successfully plans a path and optimized the controller to track the trajectory.

However, in this paper, it is assumed that the obstacles and environment are static with respect to the robot. Thus it can be further extended to solve the path planning problem in dynamic environments. Another direction could be to solve multi-robot path planning problem by minimizing the arrival time of the robots. To further reduce the computational work, fidelity of the optimization can be intelligently controlled in future.

References

[1] Khan, A.T., Li, S., Kadry, S. and Nam, Y. (2020) Control framework for trajectory planning of soft manipulator using optimized rrt algorithm. *IEEE Access* **8**: 171730–171743.

[2] Khan, A.T., Li, S., Chen, D. and Li, Y. (2020) Open-source projects for autonomous robotics and systems: A survey. *Filomat* **34**(15): 4953–4966.

[3] Chen, Z., Walters, J., Xiao, G. and Li, S. (2022) An enhanced gru model with application to manipulator trajectory tracking. *EAI Endorsed Transactions on AI and Robotics* **1**: 1–11.

[4] Karur, K., Sharma, N., Dharmatti, C. and Siegel, J.E. (2021) A survey of path planning algorithms for mobile robots. *Vehicles* **3**(3): 448–468. doi:10.3390/vehicles3030027, URL https://www.mdpi.com/2624-8921/3/3/27.

[5] Costa, M.M. and Silva, M.F. (2019) A survey on path planning algorithms for mobile robots. In *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)* (IEEE): 1–7.

[6] Khan, A.T., Li, S. and Zhou, X. (2021) Trajectory optimization of 5-link biped robot using beetle antennae search. *IEEE Transactions on Circuits and Systems II: Express Briefs* **68**(10): 3276–3280.

[7] Jordan, M. and Perez, A. (2013) Optimal bidirectional rapidly-exploring random trees .

[8] LaValle, S.M. *et al.* (1998) Rapidly-exploring random trees: A new tool for path planning .

[9] Moshayedi, A.J., Abbasi, A., Liao, L. and Li, S. (2019) Path planning and trajectroy tracking of a mobile robot using bio-inspired optimization algorithms and pid control. In *2019 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)* (IEEE): 1–6.

[10] Wong, C.C., Wang, H.Y., Li, S.A. and Cheng, C.T. (2007) Fuzzy controller designed by ga for two-wheeled mobile robots. *International Journal of Fuzzy Systems* **9**(1).

[11] Gasparetto, A., Boscariol, P., Lanzutti, A. and Vidoni, R. (2015) Path planning and trajectory planning algorithms: A general overview. *Motion and operation planning of robotic systems* : 3–27.

[12] Peralta, F., Arzamendia, M., Gregor, D., Reina, D.G. and Toral, S. (2020) A comparison of local path planning techniques of autonomous surface vehicles for monitoring applications: The ypacarai lake case-study. *Sensors* **20**(5). doi:10.3390/s20051488, URL https://www.mdpi.com/1424-8220/20/5/1488.

[13] Khan, A.T., Li, S. and Li, Z. (2021) Obstacle avoidance and model-free tracking control for home automation using bio-inspired approach. *Advanced Control for Applications: Engineering and Industrial Systems* : e63.

[14] Khan, A.T., Cao, X. and Li, S. (2022) Dual beetle antennae search system for optimal planning and robust control of 5-link biped robots. *Journal of Computational Science* : 101556.

[15] Ng, J. and Bräunl, T. (2007) Performance comparison of bug navigation algorithms. *Journal of Intelligent and Robotic Systems* **50**(1): 73–84.

[16] Khatib, O. (1985) Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation* (IEEE), **2**: 500–505.

[17] Korkmaz, M. and Durdu, A. (2018) Comparison of optimal path planning algorithms. In *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)* (IEEE): 255–258.

[18] Niu, H., Lu, Y., Savvaris, A. and Tsourdos, A. (2016) Efficient path planning algorithms for unmanned surface vehicle. *IFAC-PapersOnLine* **49**(23): 121–126.

[19] Sung, I., Choi, B. and Nielsen, P. (2021) On the training of a neural network for online path planning with offline path planning algorithms. *International Journal of Information Management* **57**: 102142.

[20] Khan, A.T., Cao, X., Li, Z. and Li, S. (2022) Evolutionary computation based real-time robot arm path-planning using beetle antennae search. *EAI Endorsed Transactions on AI and Robotics* **1**: 1–10.

[21] Geng, N., Gong, D. and Zhang, Y. (2014) Pso-based robot path planning for multisurvivor rescue in limited survival time. *Mathematical Problems in Engineering* **2014**.

[22] Buniyamin, N., Ngah, W.W., Sariff, N., Mohamad, Z. *et al.* (2011) A simple local path planning algorithm for autonomous mobile robots. *International journal of systems applications, Engineering & development* **5**(2): 151–159.

[23] Warren, C.W. (1989) Global path planning using artificial potential fields. In *1989 IEEE International Conference on Robotics and Automation* (IEEE Computer Society): 316–317.

[24] Karur, K., Sharma, N., Dharmatti, C. and Siegel, J.E. (2021) A survey of path planning algorithms for mobile robots. *Vehicles* **3**(3): 448–468.

[25] Padhy, P.K., Sasaki, T., Nakamura, S. and Hashimoto, H. (2010) Modeling and position control of mobile robot. In *2010 11th IEEE International Workshop on Advanced Motion Control (AMC)* (IEEE): 100–105.

[26] Liao, B., Li, J., Li, S. and Li, Z. (2022) Briefly revisit kinematic control of redundant manipulators via constrained optimization. *EAI Endorsed Transactions on AI and Robotics* **1**: 1–7.

[27] Khan, A.R., Khan, A.T., Salik, M. and Bakhsh, S. (2021) An optimally configured hp-gru model using hyperband for the control of wall following robot. *International Journal of Robotics and Control Systems* **1**(1): 66–74.

[28] Yu, W. and Rosen, J. (2013) Neural pid control of robot manipulators with application to an upper limb exoskeleton. *IEEE Transactions on cybernetics* **43**(2): 673–684.

[29] Xu, Q., Kan, J., Chen, S. and Yan, S. (2014) Fuzzy pid based trajectory tracking control of mobile robot and its simulation in simulink. *International journal of control and automation* **7**(8): 233–244.

[30] Taysom, B.S., Sorensen, C.D. and Hedengren, J.D. (2017) A comparison of model predictive control and pid temperature control in friction stir welding. *Journal of Manufacturing Processes* **29**: 232–241.

[31] Meshram, P. and Kanojiya, R.G. (2012) Tuning of pid controller using ziegler-nichols method for speed control of dc motor. In *IEEE-international conference on advances in engineering, science and management (ICAESM-2012)* (IEEE): 117–122.

[32] Chopra, V., Singla, S.K. and Dewan, L. (2014) Comparative analysis of tuning a pid controller using intelligent methods. *ACTA Polytechnica hungarica* **11**(8): 235–249.

[33] Akkara, S. and Jarin, T. (2022) Pi controller based switching reluctance motor drives using smart bacterial foraging algorithm. *EAI Endorsed Transactions on AI and Robotics* **1**: 1–8.

[34] Åström, K.J. and Hägglund, T. (2004) Revisiting the ziegler–nichols step response method for pid control. *Journal of process control* **14**(6): 635–650.

[35] Tan, W., Liu, J., Chen, T. and Marquez, H.J. (2006) Comparison of some well-known pid tuning formulas. *Computers & chemical engineering* **30**(9): 1416–1423.

[36] Borase, R.P., Maghade, D., Sondkar, S. and Pawar, S. (2021) A review of pid control, tuning methods and applications. *International Journal of Dynamics and Control* **9**(2): 818–827.

[37] Kuyvenhoven, N. (2002) Pid tuning methods an automatic pid tuning study with mathcad. *Calvin college ENGR* **315**.

[38] Jayachitra, A. and Vinodha, R. (2014) Genetic algorithm based pid controller tuning approach for continuous stirred tank reactor. *Advances in Artificial Intelligence (16877470)* .

[39] Bansal, H.O., Sharma, R. and Shreeraman, P. (2012) Pid controller tuning techniques: a review. *Journal of control engineering and technology* **2**(4): 168–176.

[40] Khan, A.T., Senior, S.L., Stanimirovic, P.S. and Zhang, Y. (2018) Model-free optimization using eagle perching optimizer. *arXiv preprint arXiv:1807.02754* .

[41] Zhang, S., Pu, J., Si, Y. and Sun, L. (2021) Path planning for mobile robot using an enhanced ant colony optimization and path geometric optimization. *International Journal of Advanced Robotic Systems* **18**(3): 17298814211019222.

[42] Kumar, R., Srivastava, S. and Gupta, J. (2016) Artificial neural network based pid controller for online control of dynamical systems. In *2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)* (IEEE): 1–6.

[43] Solihin, M.I., Tack, L.F. and Kean, M.L. (2011) Tuning of pid controller using particle swarm optimization (pso). In *Proceeding of the international conference on advanced science, engineering and information technology*, **1**: 458–461.

[44] Mourtas, S., Katsikis, V. and Kasimis, C. (2022) Feedback control systems stabilization using a bio-inspired neural network. *EAI Endorsed Transactions on AI and Robotics* **1**: 1–13.

[45] Yu, X. and Gen, M. (2010) *Introduction to evolutionary algorithms* (Springer Science & Business Media).

[46] Deng, W., Zhao, H., Yang, X., Xiong, J., Sun, M. and Li, B. (2017) Study on an improved adaptive pso algorithm for solving multi-objective gate assignment. *Applied Soft Computing* **59**: 288–302.

[47] Kennedy, J. and Eberhart, R. (1995) Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks* (IEEE), **4**: 1942–1948.

[48] Yang, X.S. and Deb, S. (2010) Engineering optimisation by cuckoo search. *International Journal of Mathematical Modelling and Numerical Optimisation* **1**(4): 330–343.

[49] Khan, A.T., Cao, X., Li, S., Katsikis, V.N., Brajevic, I. and Stanimirovic, P.S. (2022) Fraud detection in publicly traded us firms using beetle antennae search: A machine learning approach. *Expert Systems with Applications* **191**: 116148.

[50] Khan, A.T. and Li, S. (2022) Smart surgical control under rcm constraint using bio-inspired network. *Neurocomputing* **470**: 121–129.

[51] Khan, A.T., Li, S. and Cao, X. (2022) Human guided cooperative robotic agents in smart home using beetle antennae search. *Science China Information Sciences* **65**(2): 1–17.

[52] Khan, A.H., Li, S. and Luo, X. (2019) Obstacle avoidance and tracking control of redundant robotic manipulator: An rnn-based metaheuristic approach. *IEEE transactions on industrial informatics* **16**(7): 4670–4680.

[53] Khan, A.T., Cao, X., Li, Z. and Li, S. (2021) Enhanced beetle antennae search with zeroing neural network for online solution of constrained optimization. *Neurocomputing* **447**: 294–306.

[54] Wang, J. and Chen, H. (2018) Bsas: Beetle swarm antennae search algorithm for optimization problems. *arXiv preprint arXiv:1807.10470* .

[55] Khan, A.T., Li, S. and Cao, X. (2021) Control framework for cooperative robots in smart home using bio-inspired neural network. *Measurement* **167**: 108253.

[56] Khan, A.T., Cao, X., Li, S., Hu, B. and Katsikis, V.N. (2021) Quantum beetle antennae search: a novel technique for the constrained portfolio optimization problem. *Science China Information Sciences* **64**(5): 1–14.

[57] Khan, A.T., Cao, X., Brajevic, I., Stanimirovic, P.S., Katsikis, V.N. and Li, S. (2022) Non-linear activated beetle antennae search: A novel technique for non-convex tax-aware portfolio optimization problem. *Expert Systems with Applications* **197**: 116631.

[58] Khan, A.H., Cao, X., Katsikis, V.N., Stanimirović, P., Brajević, I., Li, S., Kadry, S. *et al.* (2020) Optimal portfolio management for engineering problems using non-convex cardinality constraint: A computing perspective. *IEEE Access* **8**: 57437–57450.

[59] Wu, Q., Shen, X., Jin, Y., Chen, Z., Li, S., Khan, A.H. and Chen, D. (2019) Intelligent beetle antennae search for uav sensing and avoidance of obstacles. *Sensors* **19**(8): 1758.

[60] Moshayedi, A.J., Roy, A.S., Sambo, S.K., Zhong, Y. and Liao, L. (2022) Review on: The service robot mathematical model. *EAI Endorsed Transactions on AI and Robotics* **1**: 1–19.

[61] Chong, K.S. and Kleeman, L. (1997) Accurate odometry and error modelling for a mobile robot. In *Proceedings of International Conference on Robotics and Automation* (IEEE), **4**: 2783–2788.