EAI Endorsed Transactions
on AI and Robotics

Research Article EAI.EU

Evolutionary Computation Based Real-time Robot Arm Path-planning Using Beetle Antennae Search

Ameer Tamoor Khan[1], Xinwei Cao[2], Zhan Li[3], Shuai Li[2,*]

[1]Department of Computing, The Hong Kong Polytechnic University, Hong Kong
[2]Department of Electronic and Electrical Engineering, Swansea University, UK
[3]Department of Computer Science, Swansea University, UK

Abstract

This paper presents a model-free real-time kinematic tracking controller for a redundant manipulator. Redundant manipulators are common in industrial applications because of the flexibility and dexterity they get from redundant joints. However, at the same time, the modeling of these systems becomes quite challenging, even for simple tasks like trajectory tracking. Some classical approaches are being used to tackle the issue, including a numerical approximation of the Jacobian and pseudo-inverse of the Jacobian matrix. These approaches have their limitations as they require exact parameters for the modeling of the manipulator; they are not immune to position error accumulation with time and put the manipulator way off the target position. Swarm-based meta-heuristic algorithms have given a new direction to the solution of the redundancy resolution problem. However, they are computationally intensive, formulated in discrete-time, and better suited for offline computation rather than real-time. We proposed a novel continuous-time Zeroing Neural Network with Beetle Antennae Search (ZNNBAS). The ZNNBAS algorithm can solve the quadratic optimization problem for redundancy resolution in real-time. To test its performance, we applied it on 7-DOF redundant manipulator with two trajectories to follow: character “M" and hypotrochoid. The manipulator was able to trace the reference trajectories with minimal tracking errors.

Received on 22 November 2021; accepted on 07 January 2022; published on 18 January 2022

Keywords: Redundant Manipulator, Neural Network, Beetle Antennae Search, Kinematic Tracking Controller.

doi:10.4108/airo.v1i.6

1. Introduction

The path planning of the robotics systems has always been an immense challenge for the researchers, especially when it involves multi-Degree Of Freedom (DOF) robots to perform daily chores like; picking, dropping, assembling, carrying, and placing objects [1–3]. Most of them are redundant manipulators as they possess more DOF than required for specific tasks. For example, it is widely recognized that six degrees of freedom are needed for end-effector motion trajectory. A robot with seven or more joints is known as a redundant manipulator for this task [4–7]. This so-called redundancy gives manipulator dexterity and flexibility in performing tasks [8–10]. The three major problems that redundant manipulators inherit are as follows:

- Kinematic model of these manipulators is a challenging task as they can have different configurations where each leads to a new kinematic model.
- No closed-form solution exists to the problem, which involves manipulators with more than six-joints [11, 12].
- These manipulators can have multiple configuration space states that results in the same workspace state.

The redundancy of these manipulators can be exploited to optimize the energy consumption and obstacle avoidance in the workspace state [13–16].

*Corresponding author. Email: Shuaili@ieee.org

In kinematic control, researchers have investigated different methods to come around the problem of redundancy resolution for the manipulators with known kinematic model [17–20]. The classical approach is the use of Jacobian matrix pseudo-inverse (JMPI) for the solution of redundancy resolution problem [21, 22]. JMPI has its limitation as it is shown in [23] that it does not have the potential to produce repeatable results and is not immune to generating undesirable configurations. A later technique to solve the redundancy resolution problem of a robotic manipulator is to approach them as a constrained optimization problem [24–26]. These methods involve a function known as a fitness or objective function. Its purpose is to refine the angles of joints continuously; the solution to redundancy resolution is to maximize the objective value [27]. In this case, the objective function is not limited to kinematic control and can generate different random configurations within and outside the kinematic model; this optimization-driven approach changes the paradigm to solve the redundancy resolution of robotic manipulators. Inspired by this approach, [28] improved optimization technique and, instead of relying alone upon fitness function, introduced another penalty function to constrain the angles of joints within a specific range based on their mechanics. Similarly, another approach inspired by the neural network is introduced in [29, 30] to solve the redundancy resolution problem where the dual neural network is used to solve the optimization problem in real-time. To be noted, all these methods require prior knowledge of the kinematic model of the robotic manipulator, and as they operate in an open-loop, so they are not immune to Position error Accumulation (PEA). Uncertainties in manipulators like; length of the robotic arms, angles of the joints, human error, and Denavit–Hartenberg (DH) parameters can affect the performance of open-loop control of the redundant manipulator. If not, the manipulator's long-term use makes it vulnerable to several uncertainties because of friction and wearing away, which can change the initial parameters of the system.

In this paper, we have introduced a kinematic-model-free method to solve the redundancy resolution problem of the manipulator. The classical approach is a numerical approximation to compute the inverse kinematic model of the robots [31–33], but they can fail to converge to a correct solution, and sometimes out of infinite many feasible solutions, they are limited to compute one [34]. This can be disadvantageous for the manipulators in trajectory planning as a single configuration may not be feasible given the limitations of manipulator mechanics [35]. Advanced techniques include training and data-driven approach, which uses a neural network to estimate the kinematic model of redundant manipulator [36]. Based on the estimated model, the robot is being controlled, but these techniques are computationally intensive and can not work in real-time as they require an extensive data set for estimation. Some approaches focus on the approximation of the Jacobian of a manipulator and design the kinematic controller of the velocity space. This technique requires the computation of Jacobian and its pseudo-inverse along the path in real-time, which is computationally expensive.

In this paper, we introduced time-invariant Zeroing Neural Network (ZNN) [37] with Beetle Antennae Search(BAS) [38] in continuous time (ZNNBAS) to design a model-free control for a redundant robotic manipulator. ZNN has different variants, but in the case of non-linear problems, they are designed to search for optimum solutions or zeros of the problem using a gradient in a closed-loop. However, meta-heuristic algorithms such as BAS do the random search in the search space to find the optimum solution to the problem. They are known to solve the non-convex problems, which include several local-minima along with global-minima [39]. Most of the meta-heuristic algorithms work in swarm-based approach where the group of searching particles, [40, 41] they used discrete Particle Swarm Optimization (PSO) to model the kinematic control of 7-DOF robotic manipulator by employing several particles in space to search for the best joint configuration possible to move the end-effector in Cartesian space. There are mainly two problems with these systems.

- As the number of particles increases, the algorithm becomes computationally expensive.
- In discrete domain, particles jump from point to point, but they have continuous motion in reality.

Here we employed an algorithm in continuous time that mimics the beetle's nature. It is known as Beetle Antennae Search (BAS). It is widely used in several real-world applications [42–51]. This algorithm involves a single particle and overcomes the problems that discrete swarm particle optimization faces. We implemented time-invariant ZNN where instead of the gradient, we used a meta-heuristic BAS algorithm for a random search of an optimum solution in a closed loop to achieve the model-free design of redundant robotic manipulator.

It is worth mentioning here that this method neither requires the kinematic model nor needs to compute Jacobian or inverse Jacobian on every iteration. It feedbacks the fitness value of the end-effector in Cartesian space to the joints in joint space to minimize their angle error. The objective of this paper is:

- Proposed a model-free kinematic controller for redundant robotic manipulator using Zeroing neural network with meta-heuristic algorithm inspired from beetles in continuous time

(ZNNBAS) which does not require the calculation of a jacobian or inverse jacobian matrix of manipulator making them computationally intensive.

- The proposed algorithm is immune to the limitations of discrete-time swarm-based meta-heuristic algorithms.
- Theoretical analysis was done to prove the stability and convergence of the algorithm.
- Simulation results were achieved on KUKA LBR IIWA-14, a 7-DOF robotic manipulator. Two path tracking scenarios were tested to prove the efficiency of the algorithm.

The remainder of the paper is as follows: Section II discusses the problem formulation of a redundant robotic manipulator. Section III gives a detailed overview of ZNBAS in continuous time and its mathematical insight. In Section IV, we discuss the simulation scenarios, results, and discussion. In section V, we concluded aper with final remarks.

2. Problem Formulation

In this section, we will discuss in detail the kinematics of redundant manipulators and the kinematic control and trajectory tracking.

2.1. Kinematics of Redundant Manipulator

Consider a robotic manipulator; the position and orientation of its end-effector depend on the configuration of its joints. To demonstrate it consider m-DOF robotic manipulator *i.e.*, m numbers of joints, and has an n-dimensional workspace or Cartesian space. The position of the end-effector in space corresponds to the unique configuration of the joint-space. The mapping from joint-space to Cartesian space is given as

$$x(t) = f(\theta(t)) \tag{1}$$

where $x(t)$ corresponds to the position of end-effector in Cartesian space and $x(t)\ \epsilon\ \Re\mathfrak{e}^n$, similarly, $\theta(t)$ corresponds to the configuration of arms in joint-space and $\theta(t)\ \epsilon\ \Re\mathfrak{e}^m$, in case of redundant manipulator $m > n$. Here, $f(.)$ is a non-linear function, which is a non-linear transformation between joint and Cartesian space. In robotics, this is known as forward kinematics, where given the configuration of joints, one can evaluate the position of the end-effector in a workspace. However, in most of the real-world applications, forward kinematics is scarcely used as tasks are carried out in Cartesian space instead of joint-space, which is known as inverse kinematics and is given as

$$\theta(t) = f^{-1}(x(t)) \tag{2}$$

where $f^{-1}(.)$ Shows the transformation from Cartesian space to the joint-space. The above equation is the inverse of (1), for a known trajectory in Cartesian space, we can compute the trajectory for the arms in joint-space. This is possible only if we know the exact kinematics of the system, so the accuracy of any trajectory followed depends on the perfection of the model. In the case of redundant manipulators, there is no unique solution in the joint-space for a given configuration in a workspace, which means that for any configuration of end-effector in Cartesian space, there can exist an infinite number of configurations in the joint space. Thus, $f^{-1}(.)$ is not uniquely invertible, and as $f(.)$ is non-linear, so most of the time, the solution to its inverse is not possible.

An approach to compute the kinematic model of a redundant manipulator is to use the Jacobian matrix. Take the time derivative of (1)

$$\dot{x}(t) = J(\theta(t))\dot{\theta}(t) \tag{3}$$

where $\dot{x}(t)$ corresponds to the velocity of end-effector in Cartesian space and $\dot{x}(t)\ \epsilon\ \Re\mathfrak{e}^n$, similarly, $\dot{\theta}(t)$ corresponds to the velocity of arms in joint-space and $\theta(t)\ \epsilon\ \Re\mathfrak{e}^m$. $J(\theta(t))\ \epsilon\ \Re\mathfrak{e}^{nxm}$ is a Jacobian matrix and is calculated as $J(\theta(t)) = \partial f(\theta(t))/\partial\theta(t)$. The (3) is again a mapping from joint-space to Cartesian space in velocity domain. As mentioned earlier in real-world tasks include mapping from workspace to joint-space so take the inverse of the above equation

$$\dot{\theta}(t) = J^{-1}(\theta(t))\dot{x}(t) \tag{4}$$

In the case of a redundant manipulator, it would be a rectangular matrix, and there is no easy way to compute the inverse of a rectangular matrix. Researchers have come around this problem by computing the pseudo-inverse of the Jacobian matrix, but it is computationally intensive. In addition, the Jacobian is valid only in the vicinity of $\theta(t)$ e.g., $J(\theta_o)$ is valid in a given range let us say ϵ from the past configuration *i.e.*, $|\theta - \theta_o| < \epsilon$, where ϵ is very small. For the whole trajectory, the algorithm needs to compute it repeatedly, which makes it computationally very expensive.

Here we assumed that the correct kinematic mapping of the redundant manipulator is known and the calculated Jacobian generates a non-singular solution, so the pseudo inverse is computable. All these strict hard-core assumptions, which involve complex computation, make the system vulnerable to the uncertainties of the environment and system itself, calculation error, and it requires much off-line modeling of the system throughout the trajectory being followed. However, our algorithm is model-free; it does not need to know whether the manipulator is redundant for a particular task or not. It does not involve computationally expensive calculations like Jacobian and inverse Jacobian of

the system. Our algorithm computes the objective function value based on the difference between end-effector coordinates in Cartesian space and the goal position and feeds it back to joints of the manipulator to tune their angles such that they follow the trajectory.

2.2. Kinematic control and trajectory Following

Consider a robotic arm manipulator, and its task is to grab a payload with the end-effector from a given location in Cartesian space, move it through the same space, and drop it to a goal position by following a given trajectory. The kinematic control in the task is as mentioned in an (1) to evaluate the states of links in joint-space and map it in the Cartesian space such that it follows the trajectory to the end-goal. To understand this, consider a reference trajectory *i.e.*, $x_r(t)$, objective is given the states of links in joint-space *i.e.*, $\theta_r(t)$, the kinematic controller should generate the position and orientation of end-effector in Cartesian space such that it should generate a reference trajectory which is given as

$$x_r(t) = f(\theta_r(t)) \tag{5}$$

where $\theta_r(t)$ is an angle of the links in joint-space which corresponds to the position of end-effector in Cartesian space *i.e.*, $x_r(t)$. As mentioned earlier, in the case of the redundant manipulator, the controller can generate many infinite solutions in joint-space for the same position and orientation of the end-effector in Cartesian space, and it is possible that many of those solutions are not feasible with the mechanical constraints of the system, for example, angle of the joint going beyond its mechanical limit or prismatic joint extended out of its limit. Our objective is to follow the goal trajectory (5) such that the generated solutions lie within the mechanical constraints of the redundant manipulator. As we are proposing an optimization technique to solve the redundancy resolution problem so we will make an optimization problem out of this, which is

$$\min_{\theta} \; h(x_r(t), \theta(t)) = \|x_r(t) - f(\theta(t))\|_2^2 \tag{6}$$

subject to:

$$\theta_1^- < \theta_1 < \theta_1^+, \quad \theta_2^- < \theta_2 < \theta_2^+, \quad \ldots \theta_m^- < \theta_m < \theta_m^+$$

where $h(x_r(t), \theta(t))$ is an optimization function that is to minimize or zero finding, and m in θ_m represents the number of joints along with the angle constraints where θ^+ and θ^- shows the maximum and minimum limits of the joint. The problem states that the controller needs to minimize the difference between the reference trajectory $x_r(t)$ and the runtime evaluated orientation and position of the end-effector $f(\theta(t))$, all in Cartesian space. Optimization problem also includes the mechanical constraints of the joints, as in our case, we only have angular joints, so we included angle constraints only. The tuning parameter in this problem is the angles of the links in the joint space $\theta(t)$. The optimization problem is nothing more than an error function which in simple form is given as

$$e = x_r(t) - f(\theta(t))$$

after this the optimization problem becomes

$$\min_{\theta} \; h(x_r(t), \theta(t)) = (e)^t(e) \tag{7}$$

subject to:

$$\theta^- < \theta < \theta^+$$

where $\theta^- = [\theta_1^-, \theta_2^-, \ldots \theta_m^-]^t$ and $\theta^+ = [\theta_1^+, \theta_2^+, \ldots \theta_m^+]^t$. The solution to this optimization problem will generate a trajectory in joint-space $\theta^*(t)$ that will follow the reference trajectory in the same space $\theta_r(t)$ which is given as.

$$\theta_r(t) = \theta^*(t) \tag{8}$$

As we mentioned earlier, our algorithm does not rely on the kinematic model of the manipulator. Instead, it is a model-free approach to solve the redundancy resolution problem of a redundant robotic manipulator. Instead of kinematic control $f(.)$, the algorithm will only rely on the state of orientation and position of end-effector coming from sensors $\hat{x}(.)$, so now the error (7) becomes

$$e = x_r(t) - \hat{x}(\theta(t))$$

It is worth mentioning here that mostly the algorithms depends on the kinematic model to compute the coordinates of the end-effector. However, our algorithm does not rely on that and directly takes the coordinates $\hat{x}(\theta(t))$ from the manipulator based on the fed angles $\theta(t)$. Now our optimization problem becomes

$$\min_{\theta} \; h(x_r(t), \theta(t)) = \|x_r(t) - \hat{x}(\theta(t))\|_2^2 \tag{9}$$

Our algorithm ZNNBAS, with its zero-finding ability (ZNN) of non-linear function through random search (BAS), can solve the optimization problem in (7) with the objective function (9).

3. Control and Algorithm Design

In this session, we will discuss the algorithm formulation of ZNNBAS and then detail theoretical analysis on the stability and convergence of ZNNBAS.

3.1. Algorithm Formulation

Consider a 7-DOF redundant manipulator required to track a given trajectory, which means that it should minimize the objective value function given in (9), in such a way that all the constraints of the redundant manipulator angles meet as shown in (7). Our proposed

algorithm involves a meta-heuristic algorithm which mimics the searching nature of Beetle insect (BAS) [38], as it senses the presence of food through its antennae. The closed-loop zero-finding nature of ZNN along with the random search of a single beetle in continuous time makes it a perfect combination to optimize the 7-DOF redundant robotic manipulator such that it follows the given reference trajectory. Consider the manipulator at time t is $\theta(t)$, ZNNBAS will start with a random direction based on its antennae sensation towards the optimum solution. The randomly generated direction vector by ZNNBAS is $\vec{\mathbf{b}}$, so now the joint configuration of the 7-DOF redundant manipulator becomes

$$\theta^{'}(t) = \beta(\theta(t) + \lambda\vec{\mathbf{b}}) \tag{10}$$

where $\beta(.)$ is a projection of joint-angles within the constraint space of joint-angles, as we mentioned earlier that we have to respect the angle limitations the manipulator as well (7). Any angle which transgress this constraint will be brought within its limit, the limiting mechanism or the formulation of $\beta(.)$ projection is given as

$$\beta(\theta_i) = \begin{cases} \theta_i^- & \theta_i < \theta_i^- \\ \theta_i & \theta_i^- < \theta_i < \theta_i^+ \\ \theta_i^+ & \theta_i > \theta_i^+ \end{cases} \tag{11}$$

This projection function limits the joint-angles with the mechanical constraints of the manipulator. The λ in euq.10 is a scaling factor of the direction vector, it could be a scalar number λ_{const} or a vector $\lambda = [\lambda_1, \lambda_2, ...\lambda_m]^t$ depends on application of how a particular joint should proceeds in joint-space.
The robotic manipulator will move the seven joints based on their respective angles and eventually the end-effector will move to its new position $\hat{x}(t)$. Here ZNNBAS algorithm will compute the error of reference trajectory with the newly computed coordinates of the end-effector as shown below

$$h(\theta^{'}(t)) = h(x_r(t), \theta^{'}(t)) = \|x_r(t) - \hat{x}(\theta^{'}(t))\|_2^2 \tag{12}$$

This is the objective function value we mentioned in detail in (9). As the algorithm started from a random point in search space, the value of the objective function (12) will be high, which ZNNBAS will minimize with a shorter period and will bring close to zero.

ZNNBAS not only minimizes the objective function but its difference with previous values as well. ZNNBAS stores a few previous states of the objective function and then later t check the difference of the current objective function value with the α old value of the objective function as it is shown below

$$\Delta h = h(\theta^{'}(t)) - h(\theta^{'}(t) - \alpha) \tag{13}$$

To limit the difference of objective function Δh, ZNNBAS used $\text{sign}(h(\theta^{'}(t)) - h(\theta^{'}(t) - \alpha))$, it will limit the difference between two limits, in our case, we keep it $\text{sign}_{min} = -1$ and $\text{sign}_{max} = 1$. The (13) is a scalar value, but to update the angles of the joints we need m x 1 vector, equivalent to the number of joints of the manipulator. For that we will multiple it with input angles and the new updated angles for all the joints will become

$$\dot{\theta}_{new}(t) = -k(\theta^{'}(t) - \theta^{'}(t - \alpha)) \\ \text{sign}(h(\theta^{'}(t)) - h(\theta^{'}(t) - \alpha)) \tag{14}$$

The bit of mathematical manipulation ended-up as a differential equation, which means that we will integrate all the previous values of the closed-loop system and the resultant will give us the new-updated angles for all the joints

$$\theta_{new}(t) = \int_0^t -k(\theta^{'}(\tau) - \theta^{'}(\tau - \alpha)) \\ \text{sign}(h(\theta^{'}(\tau)) - h(\theta^{'}(\tau) - \alpha))d\tau \tag{15}$$

To further understand its working let's see the schematic of ZNNBAS to implement the control of redundant manipulator is shown in Fig.1. It can

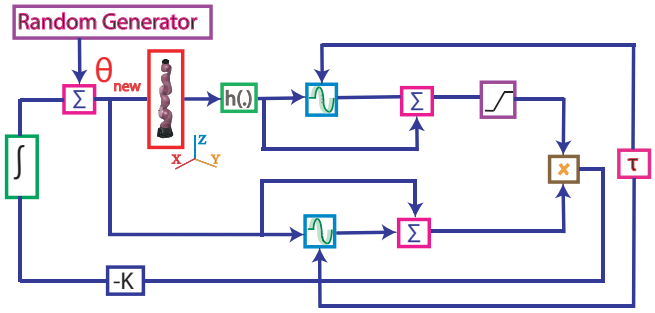

Figure 1. The schematics of the ZNNBAS to solve the redundancy resolution problem of redundant manipulator explained in Section.3.1.

be seen that *Random_Generator* generates m random signals based on the number of links of the redundant manipulator. Those generated angles are then added to all the integrated states of the closed-loop system. The resultant is then split into two, one for the input delay box and the other for the manipulator. In the case of a manipulator, the joints of the robotic system rotate as per those input angles. As a result, the end-effector moves in the workspace. The manipulator output the xyz coordinates of the robotic manipulator to the object function $h(.)$, which computes the error between input coordinates and the reference trajectory. The output is then added to delayed signals of $h(.)$. The resultant is then limited using sign function and then multiplied with an input signal to generate m x 1 vector. The

product is then integrated back with the previous state, and this closed-loop system keeps running, and eventually, the objective function error narrows down to zero (almost), and a redundant manipulator starts chasing the reference trajectory. We implemented it in MATLAB-SIMULINK.

Remark 1: In the implementation of ZNNBAS, although we kept the λ given in (10) constant, but a modification could be introduced by scaling its value as per the amount of error generated by the objective function value. As in the initial stages, the manipulator is far from the reference trajectory, producing a larger error, so the manipulator should take big steps, and when it gets closer to trajectory and start chasing it, it should take smaller steps, which is given as

$$\lambda = c_1 \sqrt{h(x_r(t), \theta(t))} \tag{16}$$

where c_1 is a constant and $h(x_r(t))$ is an optimization function in (9).

3.2. Theoretical Analysis

Here we will do the theoretical analysis of the ZNNBAS to prove that it is stable and converges to the optimal solution. For that, there are mainly two theorems to analyze one is related to the convergence of the joint angles, and the second is related to the convergence of the coordinates of the end-effector.

Definition 1: For the chasing of the reference trajectory, the controller of the manipulator should show the convergence in the join-space of the manipulator. The objective function $h(.)$ should decrease monotonically in time t. The ZNNBAS is said to be stable if

$$h(x_r(t_2), \theta(t_2)) < h(x_r(t_1), \theta(t_1)) \quad t_1 < t_2 \tag{17}$$

where $x_r(t)$ is a reference trajectory, the above equation means that objective function is a monotonically decreasing function of time.

Theorem 1: *For tracking controller of the redundant manipulator, where robotic manipulator is in joint-state of $\theta(t)$ with the end-effector coordinates $\hat{x}(t)$. The trajectory generated by ZNNBAS in continuous time to track $x_r(t)$ is stable.*

proof: The controller for redundant manipulator ZNNBAS is not stable if the object function $h(.)$ does not converge if it does not follow (17). Consider the redundant manipulator at time t_1 has a joint-state $\theta(t_1)$ with the objective function $h(x_r(t_1), \theta(t_1))$, and later in time instance t_2 the objective function becomes $h(x_r(t_2), \theta(t_2))$ where $t_1 < t_2$. The ZNNBAS will only update the joint-state of redundant manipulator if the objective function value $h(.)$ decreases monotonically which is given as

$$h(x_r(t_2), \theta(t_2)) < h(x_r(t_1), \theta(t_1)) \quad t_1 < t_2$$

otherwise, it will bring the manipulator back to its previous joint-state, and will wait until the objective function value further converge.

Definition 2: For the chasing of the reference trajectory, the controller of the manipulator should converge in the Cartesian space as well. It means that as time approaches to infinity $t \to \infty$, the end-effector trajectory must trace the reference trajectory.

$$\hat{x}(t) \to x_r(t) \quad t \to \infty \tag{18}$$

where $\hat{x}(t)$ is the trajectory generated by the manipulator and $x_r(t)$ is a reference trajectory.

Algorithm 1 Zeroing Neural Network with Beetle Antennae Search (ZNNBAS)

Input: Write and objective function $f(\boldsymbol{y(t)})$ where
$\boldsymbol{y(t)} = [y_1(t), y_2(t), y_3(t), ...y_n(t)]$
Initialize:
$\lambda \leftarrow 0.1$ %Scaling factor
k $\leftarrow -10$ %Gain
sign $\leftarrow [-1, 1]$ %Activation
$\alpha \leftarrow 10$ %Delay
$t \leftarrow 0$ %Initial time
$t_{end} \leftarrow T$ %Final time
Output:
$\theta(t)_{best} = 0$
$h(.)_{best} = h(\theta(t)_{best})$
while $\{t < t_{end}\}$
Generate random direction vector $\mathbf{b} \in [-1, 1]$,
and calculate $\theta(t)$ using (10).
Compute objective function value $h(\theta(t))$.
    if $(mod(t, (t - t_{end})) == \alpha)$
        $h(\theta(t - \beta)) = h(\theta(t))$
        $\theta(t - \beta) = \theta(t)$
    end if
    if $(h(\theta(t)) < h(.)_{best})$
        $h(.)_{best} = h(\boldsymbol{\theta(t)})$
        $\theta(t)_{best} = \theta(t)$
    end if
Update $\theta(t)$ using (15).
end while

4. Simulation Methodology, Result, and Discussion

In this section, we presented a simulation methodology, which includes IIWA-14 redundant robotic manipulator tracking the character "M" and hypotrochoid (star) using ZNNBAS.

4.1. Simulation Methodology

In order to test the performance of our proposed ZNNBAS algorithm, we used a redundant manipulator IIWA-14 provided by the MATLAB Robotics Toolbox. The robotic model is very close to the real-world

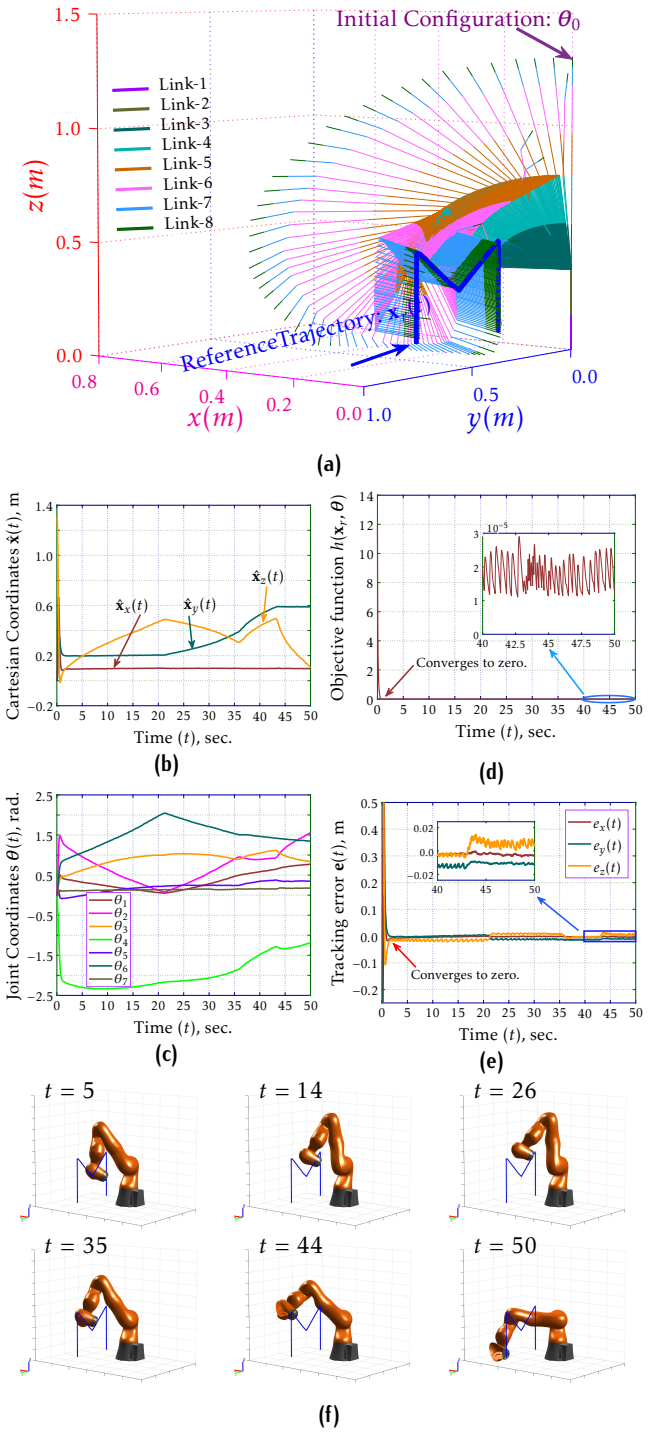

Figure 2. Simulation results for 7-DOF IIWA-4 tracking using ZNNBAS. (a) shows a robotic system tracking the character "M" trajectory, starting from the home configuration. (b) shows the XYZ coordinates of end-effector, (c) shows the angles of all seven-joints in joint-space, (d) shows the objective function value how fast it converges to zero, (e) shows the error between XYZ coordinates of end-effector and reference trajectory, and (f) shows the robot following character "M" trajectory.

IIWA-14 manipulator and includes 7-DOF; we used it as a testbed to measure the efficiency of our algorithm. We used two-tracking paths as a benchmark to prove the validity of ZNNBAS. In our simulation we tested the algorithm by tracking two paths using 7-DOF redundant robotic manipulator; the first one is Character "M" and the second one is Hypotrochoid or star. For the character "M" trajectory the equation is given below

$$x(t) = \begin{cases} \vec{c_0} + \frac{t-t_0}{t_1-t_0}(\vec{c_1} - \vec{c_0}) & t_0 < t < t_1 \\ \vec{c_1} + \frac{t-t_1}{t_2-t_1}(\vec{c_2} - \vec{c_1}) & t_1 < t < t_2 \\ \vec{c_2} + \frac{t-t_2}{t_3-t_2}(\vec{c_3} - \vec{c_2}) & t_2 < t < t_3 \\ \vec{c_3} + \frac{t-t_3}{t_4-t_3}(\vec{c_4} - \vec{c_2}) & t_3 < t < t_4 \end{cases} \tag{19}$$

where $x(t)\ \epsilon\ \Re\mathfrak{c}^3$, and $\vec{c} = [\vec{c_0}, \vec{c_1}, \vec{c_2}, \vec{c_3}, \vec{c_4}]$ represents the vertices of the character "M" such that $x(t_0) = \vec{c_0}$, $x(t_1) = \vec{c_1}$, $x(t_2) = \vec{c_2}$, $x(t_3) = \vec{c_3}$, and $x(t_4) = \vec{c_4}$. As the "M" trajectory has five vertices so the trajectory we used has following vertices: $\vec{c_0} = [0.1, 0.2, 0.1]$, $\vec{c_1} = [0.1, 0.2, 0.5]$, $\vec{c_2} = [0.1, 0.4, 0.3]$, $\vec{c_3} = [0.1, 0.6, 0.5]$, and $\vec{c_4} = [0.1, 0.6, 0.1]$. Our character "M" trajectory is defined in $y - z$ plane at $x = 0.1$.

Similarly, the other tracking trajectory is hypotrochoid (star) which is a roulette, made of two circles. It is trace by a single point joined with a smaller circle of radius r, which circle around inside the larger circle of radius R. The distance from the point to the center of the smaller circle is denoted as d. The parametric equations for hypotrochoid (star) are as follows

$$x(t) = \begin{bmatrix} c_x \\ c_y + (R-r)\cos(t)a_y + d\cos(\frac{R-r}{r}t)b_y \\ c_z + (R-r)\sin(t)a_z - d\sin(\frac{R-r}{r}t)b_z \end{bmatrix} \tag{20}$$

where $\vec{\mathbf{c}} = [\vec{\mathbf{c_x}}, \vec{\mathbf{c_y}}, \vec{\mathbf{c_z}}]^t$ is the center of the hypotrochoid (star), $\vec{\mathbf{a}} = [\vec{\mathbf{a_x}}, \vec{\mathbf{a_y}}, \vec{\mathbf{a_z}}]^t$, and $\vec{\mathbf{b}} = [\vec{\mathbf{b_x}}, \vec{\mathbf{b_y}}, \vec{\mathbf{b_z}}]^t$ are two perpendicular vector which defines the plane for 3D hypotrochoid (star). The t is an angle between the range of $0 < t < 2\pi LCM(r, R)/R$. In our case, $\vec{\mathbf{c}} = [0.1, 0.3, 0.2]^t$, $\vec{\mathbf{a}} = [0, 1, 0]^t$, and $\vec{\mathbf{b}} = [0, 0, 1]^t$. Our hypotrochoid (star) trajectory is defined in $y - z$ plane at $x = 0.1$.

Although we have tried two trajectories to validate the performance of our proposed Continuous Time Beetle Antennae Search algorithm (ZNNBAS), it will work for any other tracking path as well.

The tracking of the Character "M" trajectory is shown in Fig.2. Fig.2a shows the 3D tracking of IIWA-14 redundant manipulator by our proposed algorithm ZNNBAS. In the beginning, it shows the base or home position of the manipulator; then, by randomly searching in the workspace, it finally reaches the tracing point of the character "M" and starts following it. At first, the error between the reference trajectory and the

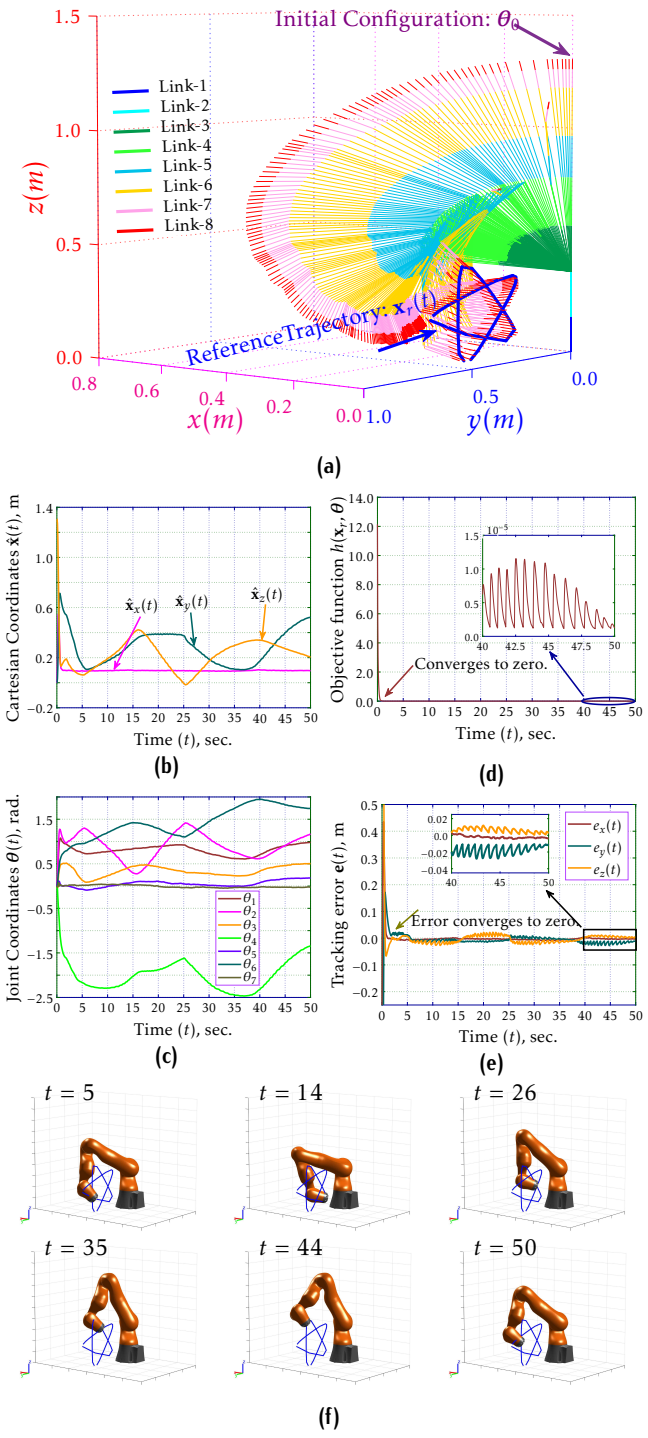

Figure 3. Simulation results for 7-DOF IIWA-4 tracking a hypotrochoi (star) trajectory using ZNNBAS. (a) shows robotic system tracking the hypotrochoid (star) trajectory, starting from home configuration. (b) shows the XYZ coordinates of end-effector, (c) shows the angles of all seven-joints in joint-space, (d) shows the objective function value how fast it converges to zero, (e) shows the error between XYZ coordinates of end-effector and reference trajectory, and (f) shows the robot following hypotrochoi (star) trajectory.

manipulator's trajectory is large, which narrows down as the manipulator starts chasing it, and this is all done by a single beetle particle working in continuous time. The Fig.2b shows the XYZ coordinates of the end-effector. It can be seen that character "M" is in $y-z$ axes as the y and z coordinates of the end-effector move in the plane, and the x-coordinate remains constant. Similarly, Fig.2c shows the angles of the redundant manipulator throughout the tracking. The Fig.2d shows the objective function value, and as it is seen that simulation lasts for $50sec$, but even less than $10sec$ the error approaches zero, and during the time from $40sec$ to $50sec$ the error went down to 10^{-5}, which shows how efficiently and precisely ZNNBAS optimizes the problem without showing any spikes of noise or undesirable behavior of manipulator. Fig.2e shows the error in each coordinate, and as it can be seen, that error approached zero in no time.

For the hypotrochoid (star) trajectory, the results are shown in Fig.3. The Fig.3a shows the 3D tracking of the redundant manipulator, which starts from the base or home position, which is erect and then moves through space randomly to trace the reference track, and eventually, it did track the hypotrochoid (star). This shows the power of a single beetle in continuous time and how efficiently and accurately it tracks down complex trajectories like a star. Fig.3b and Fig.3c show the coordinates of the end-effector and the angles of all the seven links of the IIWA-14 redundant manipulator. Similarly, Fig.3d shows the objective function value, and in a simulation of $50sec$, it approaches 0 in less than $3sec$. Likewise, Fig.3e shows the error in xyz coordinates of the end-effector, and it can be seen that they approach zero in no time.

The major drawback or limitation of ZNNBAS is the correct determination of α, which is the core hyper-parameter to determine the performance of the algorithm. Likewise, the other hyper-parameters are manually set. In future, we plan to work on these limitations to make ZNNBAS more robust and efficient.

5. Conclusion

The objective of this paper was to address the redundancy resolution problem of the redundant manipulator used in numerous industrial applications and their kinematic control design. We proposed a model-free control for redundant manipulators using Zeroing Neural Network with Beetle Antennae Search (ZNN) in continuous time. ZNN alone used a gradient-based method to search for an optimal solution that may limit it to local-minima, but with the random search of BAS, it can avoid local-minima and converge to a global solution. The redundancy resolution problem is tackled through Particle Swarm Optimization (PSO) algorithm, but they

are computationally intensive and do not mimic the nature of the insect army in general as they are discrete. ZNNBAS has overcome these challenges as it involves a single particle to solve the redundancy resolution problem of a redundant robotic manipulator in continuous time. We theoretically prove the stability and convergence of ZNNBAS, and a test case, we used 7-DOF IIWA-4 redundant manipulator provided by MATLAB and successfully tracked two trajectories *i.e.*, character "M", and Hypothyroid (star).

References

[1] R. U. Ayres and S. M. Miller, "Robotics: Applications and social implications," 1983.

[2] R. A. El-laithy, J. Huang, and M. Yeh, "Study on the use of microsoft kinect for robotics applications," in *Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium*, pp. 1280–1288, IEEE, 2012.

[3] D. Wu, Y. He, X. Luo, and M. Zhou, "A latent factor analysis-based approach to online sparse streaming feature selection," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2021.

[4] G. S. Chirikjian and J. W. Burdick, "A modal approach to hyper-redundant manipulator kinematics," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 3, pp. 343–354, 1994.

[5] J. Wang, Y. Li, and X. Zhao, "Inverse kinematics and control of a 7-dof redundant manipulator based on the closed-loop algorithm," *International Journal of Advanced Robotic Systems*, vol. 7, no. 4, p. 37, 2010.

[6] M. Benzaoui, H. Chekireb, M. Tadjine, and A. Boulkroune, "Trajectory tracking with obstacle avoidance of redundant manipulator based on fuzzy inference systems," *Neurocomputing*, vol. 196, pp. 23–30, 2016.

[7] M. Shang, Y. Yuan, X. Luo, and M. Zhou, "An α-β-divergence-generalized recommender for highly accurate predictions of missing user preferences," *IEEE Transactions on Cybernetics*, 2021.

[8] D. Liang, Y. Song, T. Sun, and X. Jin, "Rigid-flexible coupling dynamic modeling and investigation of a redundantly actuated parallel manipulator with multiple actuation modes," *Journal of Sound and Vibration*, vol. 403, pp. 129–151, 2017.

[9] G. Chen, D. Liu, Y. Wang, Q. Jia, and X. Liu, "Contact force minimization for space flexible manipulators based on effective mass," *Journal of Guidance, Control, and Dynamics*, pp. 1–8, 2019.

[10] Z. Li, S. Li, and X. Luo, "An overview of calibration technology of industrial robots," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 1, p. 23, 2021.

[11] N. Kofinas, E. Orfanoudakis, and M. G. Lagoudakis, "Complete analytical forward and inverse kinematics for the nao humanoid robot," *Journal of Intelligent & Robotic Systems*, vol. 77, no. 2, pp. 251–264, 2015.

[12] G. Tevatia and S. Schaal, "Inverse kinematics for humanoid robots," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 1, pp. 294–299, IEEE, 2000.

[13] A. M. Zanchettin, L. Bascetta, and P. Rocco, "Acceptability of robotic manipulators in shared working environments through human-like redundancy resolution," *Applied ergonomics*, vol. 44, no. 6, pp. 982–989, 2013.

[14] A. M. Zanchettin, L. Bascetta, and P. Rocco, "Achieving humanlike motion: Resolving redundancy for anthropomorphic industrial manipulators," *IEEE Robotics & Automation Magazine*, vol. 20, no. 4, pp. 131–138, 2013.

[15] J. Hollerbach and K. Suh, "Redundancy resolution of manipulators through torque optimization," *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 308–316, 1987.

[16] B. Wang and J. Fang, "A hybrid type admm for multi-block separable convex programming," in *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 1–6, IEEE, 2018.

[17] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, pp. 398–410, 1997.

[18] Y. Zhang, J. Wang, and Y. Xia, "A dual neural network for redundancy resolution of kinematically redundant manipulators subject to joint limits and joint velocity limits," *IEEE transactions on neural networks*, vol. 14, no. 3, pp. 658–667, 2003.

[19] A. G. Ruiz, J. C. Santos, J. Croes, W. Desmet, and M. M. da Silva, "On redundancy resolution and energy consumption of kinematically redundant planar parallel manipulators," *Robotica*, vol. 36, no. 6, pp. 809–821, 2018.

[20] J. Nurmi and J. Mattila, "Global energy-optimised redundancy resolution in hydraulic manipulators using dynamic programming," *Automation in Construction*, vol. 73, pp. 120–134, 2017.

[21] D. Chen, Y. Zhang, and S. Li, "Tracking control of robot manipulators with unknown models: A jacobian-matrix-adaption method," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3044–3053, 2017.

[22] S. Li, Y. Zhang, and L. Jin, "Kinematic control of redundant manipulators using neural networks," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2243–2254, 2016.

[23] Y. Wan, Y. Kou, and X. Liang, "Closed-loop inverse kinematic analysis of redundant manipulators with joint limits," in *International Conference on Mechanical Design*, pp. 1241–1255, Springer, 2017.

[24] L. Jin, S. Li, H. M. La, and X. Luo, "Manipulability optimization of redundant manipulators using dynamic neural networks," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 6, pp. 4710–4720, 2017.

[25] B. Wang, H. Jiang, J. Fang, and H. Duan, "A proximal admm for decentralized composite optimization," *IEEE Signal Processing Letters*, vol. 25, no. 8, pp. 1121–1125, 2018.

[26] B. Wang, J. Fang, H. Duan, and H. Li, "Proximal-free admm for decentralized composite optimization via graph simplification," *arXiv preprint arXiv:1809.01346*, 2018.

[27] M. M. Elshabasy, K. T. Mohamed, and A. A. Ata, "Power optimization of planar redundant manipulator moving along constrained-end trajectory using hybrid techniques," *Alexandria Engineering Journal*, vol. 56, no. 4, pp. 439–447, 2017.

[28] S. Li, M. Zhou, and X. Luo, "Modified primal-dual neural networks for motion control of redundant manipulators with dynamic rejection of harmonic noises," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 10, pp. 4791–4801, 2017.

[29] L. Jin, S. Li, X. Luo, Y. Li, and B. Qin, "Neural dynamics for cooperative control of redundant robot manipulators," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 3812–3821, 2018.

[30] Y. Zhang and J. Wang, "Obstacle avoidance for kinematically redundant manipulators using a dual neural network," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 752–759, 2004.

[31] M. V. Weghe, D. Ferguson, and S. S. Srinivasa, "Randomized path planning for redundant manipulators without inverse kinematics," in *2007 7th IEEE-RAS International Conference on Humanoid Robots*, pp. 477–482, IEEE, 2007.

[32] S. Li, H. Wang, and M. U. Rafique, "A novel recurrent neural network for manipulator control with improved noise tolerance," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 5, pp. 1908–1918, 2017.

[33] Z. Li and S. Li, "Neural network model-based control for manipulator: An autoencoder perspective," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[34] A. Freddi, S. Longhi, A. Monteriù, and D. Ortenzi, "Redundancy analysis of cooperative dual-arm manipulators," *International Journal of Advanced Robotic Systems*, vol. 13, no. 5, p. 1729881416657754, 2016.

[35] J. Zhao, T. Xu, Q. Fang, Y. Xie, and Y. Zhu, "A synthetic inverse kinematic algorithm for 7-dof redundant manipulator," in *2018 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pp. 112–117, IEEE, 2018.

[36] C. Yang, X. Wang, L. Cheng, and H. Ma, "Neural-learning-based telerobot control with guaranteed performance," *IEEE transactions on cybernetics*, vol. 47, no. 10, pp. 3148–3159, 2016.

[37] L. Jin, S. Li, B. Liao, and Z. Zhang, "Zeroing neural networks: A survey," *Neurocomputing*, vol. 267, pp. 597–604, 2017.

[38] X. Jiang and S. Li, "Bas: beetle antennae search algorithm for optimization problems," *arXiv preprint arXiv:1710.10724*, 2017.

[39] V. K. Kamboj, S. Bath, and J. Dhillon, "Solution of non-convex economic load dispatch problem using grey wolf optimizer," *Neural Computing and Applications*, vol. 27, no. 5, pp. 1301–1316, 2016.

[40] S. Suresh and S. Lal, "Multilevel thresholding based on chaotic darwinian particle swarm optimization for segmentation of satellite images," *Applied Soft Computing*, vol. 55, pp. 503–522, 2017.

[41] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Computing and Applications*, vol. 27, no. 4, pp. 1053–1073, 2016.

[42] A. T. Khan, S. Li, and X. Cao, "Control framework for cooperative robots in smart home using bio-inspired neural network," *Measurement*, vol. 167, p. 108253, 2021.

[43] A. T. Khan and S. Li, "Human guided cooperative robotic agents in smart home using beetle antennae search," *Science China Information Sciences*, 2021.

[44] A. T. Khan, X. Cao, S. Li, B. Hu, and V. N. Katsikis, "Quantum beetle antennae search: a novel technique for the constrained portfolio optimization problem," *Science China Information Sciences*, vol. 64, no. 5, pp. 1–14, 2021.

[45] A. T. Khan, S. Li, and X. Zhou, "Trajectory optimization of 5-link biped robot using beetle antennae search," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2021.

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

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

[48] A. T. Khan, X. Cao, S. Li, V. N. Katsikis, I. Brajevic, and P. S. Stanimirovic, "Fraud detection in publicly traded us firms using beetle antennae search: A machine learning approach," *Expert Systems with Applications*, p. 116148, 2021.

[49] A. T. Khan and S. Li, "Smart surgical control under rcm constraint using bio-inspired network," *Neurocomputing*, 2021.

[50] A. H. Khan, X. Cao, S. Li, V. N. Katsikis, and L. Liao, "Bas-adam: an adam based approach to improve the performance of beetle antennae search optimizer," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 2, pp. 461–471, 2020.

[51] A. H. Khan, X. Cao, V. N. Katsikis, P. Stanimirović, I. Brajević, S. Li, S. Kadry, and Y. Nam, "Optimal portfolio management for engineering problems using nonconvex cardinality constraint: a computing perspective," *IEEE Access*, vol. 8, pp. 57437–57450, 2020.