Using Social Behavior of Beetles to Establish a Computational Model for Operational Control

Ameer Hamza Khan, Xinwei Cao, Shuai Li, and Chunbo Luo

***Abstract*—In this paper, we computationally model the social behavior of beetles and apply it to the tracking control of manipulators. The beetles demonstrate excellent skills to forage food in a previously unknown environment by merely using their olfactory senses. The goal of the beetle is to search the region with the maximum smell. Therefore, the actions of the beetle can be characterized as an optimization algorithm. This paper mathematically models this behavior in the form of a Recurrent Neural Network (RNN) with a temporal-feedback connection. We apply the formulated RNN controller for the redundancy resolution and tracking control of the redundant manipulators with an unknown kinematic model. Most of the industrial robots have redundant manipulators, and kinematic trajectory tracking is a fundamental problem for any industrial task. The behavior of the beetle allows us to formulate a position-level controller without relying on the manipulation of the Jacobian matrix. It is in contrast with the conventional velocity-level controllers, which require an accurate kinematic model of the manipulator and calculation of pseudo-inverse of Jacobian, a computationally expensive task. The proposed algorithm, called Beetle Antennae Olfactory Recurrent Neural Network (BAORNN) algorithm; is capable of driving the manipulator by only using the feedback from the position and orientation sensors. The stability and convergence of the proposed algorithm are theoretically proved, and simulations results using a 7-DOF industrial robotic arm, KUKA LBR IIWA14, are presented to demonstrate the performance of the proposed algorithm.**

I. Introduction

The social behavior of living beings in their common habitats serves as a great source of inspiration for the development of nature-inspired metaheuristic algorithms. Several nature-inspired optimization algorithms have been proposed in literature [1]–[4]. The process of evolution and natural over millions of years have optimized the behaviors of these living beings to carry out their tasks efficiently. Behavior of several animals have been formulated into an optimization algorithm e.g. ants [5], honey bees [6], fireflies [7], cuckoo [8] and bats [9]. However, most of these algorithms rely on the swarming behaviors of these animals and therefore require a large number of search particles. However, the natural behaviors of beetles are peculiar, such that they do not work in a swarm to search for food. An individual beetle alone is capable of foraging food based on its olfactory sense. By comparing the intensity of smell at two antennae locations, the beetle can create a map of smell intensity and find an optimal direction leading toward the food source. The food foraging behavior of the beetle is shown in Fig. 1. Jiang and Li [10] mathematically modeled this behavior into the BAS algorithm. In this paper, further building on their work, we propose an RNN architecture to model the beetle behavior and apply it to solve a real-world problem; tracking control of redundant manipulators with an unknown model.

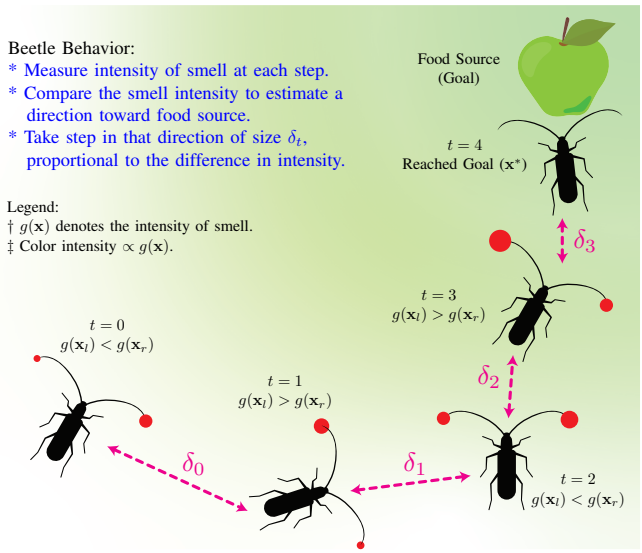


Fig. 1: Illustration of the beetle's food foraging behavior. At each step, the beetle stops and measure the smell intensity using its antennae. Then by comparing the smell intensity, it decides an optimal direction toward the food source.

Most of the real-world systems are highly nonlinear, and their real-time functioning requires efficiently solving a time-varying non-convex optimization problem. One example of such systems is redundant manipulators. With the advances in mechatronics, control theory and computing systems, using a multi-Degree Of Freedom (DOF) robots to perform common tasks, e.g. picking, holding, packing, assembling, and transporting have gained a lot of attention in academia as well as in industry [11]–[19]. These robots usually have more DOF then required by the underlying task; for example, robotic arms have more than three joints when the task requires tracking a path in 3D space [20], [21]. Kinematic modeling of such a robot is a challenging task because redundant manipulators can be connected in several different configurations, each leading to an entirely different kinematic model. In fact for most robotic arms with redundant joints, a closed-form inverse kinematic solution model does not even exist [22], [23]. Additionally, the redundancy provided by these extra joints allow the robotic arm to have multiple solutions in

A. H. Khan is with Department of Computing, Hong Kong Polytechnic University (email: ameer.h.khan@comp.polyu.edu.hk).
X. Cao is with Shanghai University, China (email: xinweicao@shu.edu.cn).
S. Li is with Swansea University, Swansea, UK (email: shuaili@ieee.org).
C. Luo is with Department of Computer Science, University of Exeter, Exeter, EX4 4RN, UK. (email: c.luo@exeter.ac.uk).
X. Cao is the corresponding author.

joint space for the same problem in task space. This extra freedom available in joint space is exploited to optimize the performance of robot and achieve secondary design objectives, such as minimize energy consumption, optimize joint-torques and obstacle avoidance [24]–[28].

Redundancy resolution for robots with a known kinematic model is a fundamental and well-studied problem in kinematic control [21], [26], [29]–[33]. For a robot with redundant manipulators, if a task is given in cartesian space, correspondingly an infinite configurations exist in joint space to perform the same task. Choosing an optimal configuration presents a challenging problem. Traditionally, for a robot with a known kinematic model, a general solution to redundancy resolution problem involves Jacobian-matrix-pseudo-inverse (JMPI) [34]–[36]. However, it was shown in [37] that the method based on JMPI does not produce repeatable results and can potentially lead to undesirable joint configuration. Later approaches for model-based kinematic control of a robot with redundant manipulators formulate redundancy resolution as a constrained optimization problem [24]–[26], [38]. These approaches use a fitness function, also called an objective function, which assigns a fitness value to each configuration in joint space. The solution to redundancy resolution is to find a configuration with maximum fitness value. The optimization-driven approach to redundancy resolution provides a new perspective toward the robot control problem because the objective function does not necessarily have to be restricted, just to the kinematic control [39]–[42]. Inspired from this approach, [43] introduced additional penalty terms in the objective function to constraint the joints inside a mechanically optimal range. Similarly, [29], [44], [45] proposed a dual Recurrent Neural Network (RNN) for solving the optimization problem in real-time. A local optimization algorithm for model-based redundancy resolution of serial and parallel manipulators was proposed in [46], however the solution is only locally optimal and and create large approximation errors. It must be noted that all of the above mentioned algorithms are highly dependent on the apriori known kinematic model of the robot and suffer from Position Error Accumulation (PEA) because of their open-loop nature. Uncertainty in mechanical parameters of the robot, e.g., length of links and location of joints, or the Denavit–Hartenberg (DH) parameters introduce large errors in the designed controllers. Even, when the mechanical parameter of the robot are exactly known apriori, the effect of long-term use, e.g., friction and wearing may introduce uncertainty in the initially known mechanical parameters of the manipulator.

For a robot with an unknown kinematic model (which is investigated in this paper), the problem of redundancy resolution remains a challenging issue. Conventional methods to solve this problem focus on numerically estimating an inverse kinematic model of a robot using training and data-driven approaches, such as neural networks [23], [47]–[49]. The estimated model is then used to control the robot directly. However, model estimation and training of the neural network is a computationally expensive task, which requires a large amount of training dataset and cannot be accomplished in real-time. Some other algorithms [20], [50], [51] focus on estimating the Jacobian of the robot and design a kinematic controller in velocity space. Similarly, approaches based on adaptive control and barrier-Lyapunov-function [52]–[58] have been proposed. Although the methods mentioned above are capable of solving the problem of redundancy resolution for a robot with an unknown kinematic model, however, these require estimation of Jacobian matrix and calculation of its pseudo-inverse at runtime for waypoint along the path of the robot. Calculation of matrix pseudo-inverse is a computationally expensive task, which reduces the real-time viability of these algorithms.

In this paper, we present a novel metaheuristic approach; Beetle Antennae Olfactory Recurrent Neural Network, called BAORNN from henceforth, to solve the problem of redundancy resolution for model-free kinematic control of robotic arms in real-time. Here we take advantage of the fact that the redundancy resolution problem is essentially equivalent to searching the joint space to find an optimal configuration corresponding to a task in cartesian space. Metaheuristic optimization algorithm are well-known [1], [2], [59]–[62] for their efficiency to search non-convex spaces for optimal solution of nonlinear objective functions [63], [64]. Here we leverage this property to search the joint space of our robotic arm at runtime to find an optimal configuration. It must be noted that metaheuristic optimization algorithms have not only been a focus of research interest but have found several applications in real-world scenarios [65], [66]. For example, several works have been proposed to study the problem of inverse-kinematics using metaheuristic optimization [28], [67]. The BAORNN algorithm, proposed in this paper, is based on Beetle Antennae Olfactory (BAO) algorithm [10], [68], which is a nature-inspired metaheuristic optimization algorithm inspired by the food foraging behavior of beetles. BAO algorithm is chosen because unlike other metaheuristic algorithms; it makes use of only a single search particle to search the optimum of an objective function, which make it suitable for model-free optimization and parameter tuning of real-world systems. In this paper, the BAO algorithm is modeled as an RNN to solve the redundancy resolution optimization problem in real-time and applied for the kinematic control of a robotic arm. The BAORNN algorithm starts from an initial configuration of the robotic arm joints and efficiently searches the joint space to minimize the tracking error.

It should be noted that unlike previous works, the proposed method neither need the forward or inverse kinematic model of the robotic arm nor the Jacobian and only relies on the feedback provided by the cartesian position and orientation sensor. This is in contrast to the traditional methods which use an analytic or estimated kinematic model of the robot. Since the proposed algorithm is Jacobian-free, therefore it does not require the computation of matrix pseudo-inverse, which make it very computationally efficient. We present the formulation of the BAO algorithm for our robotic arm as a RNN and theoretically prove its stability and convergence. The main highlights of this paper are:

1) We proposed a model-free controller for the kinematic control of a redundant joint robotic arm which does

not require estimation of the kinematic model or Jacobian matrix and does not involve the calculation of pseudo-inverse of Jacobian matrix.
2) For a given task in cartesian space, the proposed controller leverage the efficiency of the metaheuristic algorithm to efficiently search the joint spaces to find an optimal joint configuration.
3) Theoretical analysis is done to prove the stability and convergence of the proposed algorithm.
4) Simulation results using, KUKA LBR IIWA-14, a popular 7-DOF industrial robotic arm, proves the efficacy of the proposed approach.

The remainder of this paper is organized as follows: Section II presents the problem formulation of the redundancy resolution of a robotic arm with multiple manipulators. In Section III, the details BAORNN algorithm are laid down and theoretically shown that the algorithm produces a globally convergent solution. Section IV presents the simulation methodology, results, and discuss its importance. Section V concludes the paper.

II. Problem Formulation

In this section, we present the mathematical formulation of redundancy resolution problem for a general redundant robotic manipulators. Firstly, a brief introduction about the kinematics model is provided, and the shortcoming of model-based kinematic control is outlined. Next, the robotic task execution is formulated as a quadratic optimization problem.

A. Redundant Manipulator Kinematics

Given a redundant joint robotic arm, the position and orientation of end-effector are uniquely defined by the configuration of its joints. Consider a robotic arm with m-DOF robotic arm having m joints and operating in an n-dimensional cartesian space. The following mapping from joint space coordinates to the cartesian coordinates is surjective

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

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the vector representing coordinates of end-effector in cartesian space and $\boldsymbol{\theta}(t) \in \mathbb{R}^m$ are the coordinates in joint space. Note $m > n$ for a redundant joint robotic arm. The mapping function $f(.)$ is nonlinear and represents the forward kinematics of the robotic arm. Forward mapping function $f(.)$ known for a given robotic arm. However, the task for a robotic arm is usually defined in the cartesian coordinates instead of the joint coordinates;, therefore, the forward kinematics is of little interest. Based on 1, we can define the following inverse kinematics model

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

where $f^{-1}(.)$ represents the mapping from the cartesian space to the joint space and is inverse of the mapping defined in (1). If the inverse mapping is known, then for a given task in cartesian space the corresponding trajectory in the joint space can be easily calculated using (2) and robot's joint can be directly controlled using the calculated trajectory. Unfortunately, for a redundant robotic arm, the forward mapping $f(.)$ is surjective only and not one-to-one i.e., there exist an infinite number of coordinates in joint space, which are mapped to the same coordinate in the cartesian space. Therefore, function $f(.)$ is not uniquely invertible. Furthermore, $f(.)$ is a nonlinear , and its inverse cannot be expressed analytically for most cases.

The above discussion pertains to the formulation of the controller at position-level; however, the velocity-level control is an alternate approach for approximating the inverse kinematics of the robotic arm involving the Jacobian matrix. Taking time derivative of the (1) yields

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

where $\dot{\mathbf{x}}(t) \in \mathbb{R}^n$ and $\dot{\boldsymbol{\theta}}(t) \in \mathbb{R}^m$ are the velocity of robotic arm in cartesian and joint space respectively. $J(\boldsymbol{\theta}(t)) \in \mathbb{R}^{n\times m}$ is the Jacobian matrix and calculated as $J(\boldsymbol{\theta}(t)) = \partial f(\boldsymbol{\theta}(t))/\partial \boldsymbol{\theta}(\boldsymbol{t})$, i.e., $J_{ij} = \partial f_i/\partial \boldsymbol{\theta_j}$, where J_{ij} represent the element in i^{th} row and j^{th} column of the Jacobian matrix. By analysing the equation (3), it can be seen that it represent a surjective affine mapping from $\dot{\boldsymbol{\theta}}(t)$ to $\dot{\mathbf{x}}(t)$ in the close neighbourhood of a specified joint space coordinates and therefore a velocity-level inverse kinematic model can be formulated as follow

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

However, for a redundant manipulator robotic arm the Jacobian $J(\boldsymbol{\theta}(t))$ is a rectangular matrix; therefore the above relation requires calculating the pseudo-inverse of a matrix which is a computationally intensive process. Furthermore, a specific value of Jacobian, e.g. $J(\boldsymbol{\theta_0})$ is only valid in the close neighbourhood of $\boldsymbol{\theta_0}$ i.e. $||\boldsymbol{\theta} - \boldsymbol{\theta_0}||_2 < \epsilon$ for an arbitrary small constant ϵ and needs to be re-evaluated several times for the entire trajectory of the robotic arm. The recalculation of Jacobian and the subsequent matrix inversion make this process unsuitable for commonly available embedded processors.

The above discussion assumes that the forward kinematics mapping $f(.)$ is precisely known, its Jacobian can be calculated and generate a non-singular matrix so that the pseudo-inverse can also be calculated. All of these assumptions suffer for a high degree of uncertainty, modeling errors, and requires intensive offline model estimations to provide desirable results at runtime. In contrast, our algorithm is formulated on the assumption that we do not have any information about the kinematic model of the robotic arm and only have access to the sensor data measuring the position and orientation of the end-effector in cartesian space. By using the measured cartesian coordinates in feedback, our control algorithm can accurately track any arbitrary reference trajectory in a model-free manner.

B. Kinematic Trajectory Tracking

Consider the task of moving a payload with a robotic arm. The task requires that the robotic arm grabs the object using end-effector, move the payload along a desired trajectory in cartesian space and release the object at the destination. The goal of the kinematic control is to find an optimal trajectory in

the joint space to trace the desired trajectory in cartesian space. Consider a reference path $\mathbf{x}_r(t)$ in cartesian coordinates which we want the end-effector to trace, then the corresponding trajectory in joint space must satisfy the following relation

$$\mathbf{x}_r(t) = f(\boldsymbol{\theta_r}(t)), \tag{5}$$

where $\boldsymbol{\theta_r}(t)$ is a trajectory in the joint space corresponding to $\mathbf{x}_r(t)$. Note that for a redundant manipulator, an infinite number of trajectories $\boldsymbol{\theta_r}(t)$ satisfy this relation for a given trajectory $\mathbf{x}_r(t)$, many of which might not satisfy the mechanical limits of the joints, for example, extending a prismatic joint beyond its maximum length. Our goal is to find a trajectory in joint space which satisfies the (5) while respecting the constraints on the mechanically optimal range of joint motion. Without loss of generality, we can model the optimal trajectory tracking as the following constrained quadratic optimization problem

$$\min_{\boldsymbol{\theta}} \quad g(\mathbf{x}_r(t), \boldsymbol{\theta}(t)) = \frac{1}{2}||\mathbf{x}_r(t) - f(\boldsymbol{\theta}(t))||_2^2 \tag{6}$$

subject to:

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

where $g(\mathbf{x}_r(t), \boldsymbol{\theta}(t))$ denotes the objective function, m denotes the total number of joints, θ_k^- and θ_k^+ $(k \in \{1, 2, ..., m\})$ represents the minumum and maximum position limits of the k^{th} joint. To simplify the notation lets define tracking error as

$$\mathbf{e} = \mathbf{x}_r(t) - f(\boldsymbol{\theta}(t))$$

and the optimization problem (6) can be concisely written as

$$\min_{\boldsymbol{\theta}} \quad g(\mathbf{x}_r(t), \boldsymbol{\theta}(t)) = \mathbf{e}^T\mathbf{e} \tag{7}$$

subject to:

$$\boldsymbol{\theta}^- < \boldsymbol{\theta} < \boldsymbol{\theta}^+.$$

where $\boldsymbol{\theta}^- = [\theta_1^-, \theta_2^-, ..., \theta_m^-]^T$ and $\boldsymbol{\theta}^+ = [\theta_1^+, \theta_2^+, ..., \theta_m^+]^T$. The solution $\boldsymbol{\theta}^*$ of optimization problem (7) gives us the required trajectory in the joint space, i.e.,

$$\boldsymbol{\theta}_r = \boldsymbol{\theta}^*.$$

As already mentioned in Section II-A, our algorithm is formulated on the assumption that the kinematic model $f(.)$ of the robotic arm is unknown, and we only have access to position and orientation data from the sensor. Therefore in our case, we define tracking error $\mathbf{e}$ as

$$\mathbf{e} = \mathbf{x}_r(t) - \hat{\mathbf{x}}(\boldsymbol{\theta}(t))$$

where $\hat{\mathbf{x}}(\boldsymbol{\theta}(t))$ denotes the real cartesian coordinate of the end-effector coming directly from the sensor. It is worth pointing out that most the traditional methods rely on the forward kinematic model $f(.)$ to estimate the current cartesian coordinates instead of sensor data. Based on this definition, the final objective function for our algorithm becomes

$$g(\mathbf{x}_r(t), \boldsymbol{\theta}(t)) = ||\mathbf{x}_r(t) - \hat{\mathbf{x}}(\boldsymbol{\theta}(t))||_2^2. \tag{8}$$

The BAORNN algorithm solves the constrained optimization problem (7) with objective function given in (8).

Although numerical methods are available to solve the proposed optimization problem on point to point basis, at each time sample, for the entire trajectory of the robotic arm, however, the computation cost for such a strategy is inversely proportional to the sampling interval. For small time intervals, such a strategy have huge computation cost. However, in this paper, we propose the BAORNN algorithm, which uses metaheuristic information from past time steps to solve the optimization problem at runtime using recurrent neural networks while simultaneously moving the robotic arm.

Remark 1. *Although, only the kinematic model of the manipulator is used in the formulation of the objective function (8). However, manipulator controllers based on the kinematic model have been a subject of study in several recent works [20], [38], [69], [70]. Several industrial manipulators [71]–[74] have also found applications of kinematic control.*

Remark 2. *Although the constrained optimization problem (7) only enforce tracking the reference trajectory $\mathbf{x}_r(t)$ while respecting the joint constraints, the proposed optimization-driven framework is quite general. It can easily be extended to include other optimization objectives by following the approaches of [26], [40], [43]. As an example, the following optimization problem incorporates reference trajectory tracking as well as velocity minimization (which in effect minimize power consumption) with additional mechanical constraints on the velocities of the joints*

$$\min_{\boldsymbol{\theta}} \quad \mathbf{e}^T\mathbf{e} + \dot{\boldsymbol{\theta}}^T\dot{\boldsymbol{\theta}} \tag{9}$$

subject to:

$$\boldsymbol{\theta}^- < \boldsymbol{\theta} < \boldsymbol{\theta}^+,$$
$$\dot{\boldsymbol{\theta}}^- < \dot{\boldsymbol{\theta}} < \dot{\boldsymbol{\theta}}^+.$$

Similarly, obstacle avoidance can be incorporated as maximization of the distance between robot and objects present in the surrounding environment by following the approach of [25]. However, in this paper, we will consider the objective function defined in (7).

III. Control System Design

In this section, a recurrent neural network architecture is developed, and the BAORNN algorithm is formulated to solve the optimization problem (7) at runtime while concurrently moving the robotic arm along the reference trajectory. It is followed by a theoretical analysis of the convergence of the proposed algorithm.

A. Algorithm Formulation

Consider a m-DOF redundant robotic arm required to track a reference trajectory. Our goal is to minimize the objective function given in (8) while respecting the constraints given in (7). BAORNN algorithm mimics the behavior of a beetle to probe the joint space using its two antennae. Suppose, the initial coordinates of the robot in joint space is $\boldsymbol{\theta}_0$. At time-step t_k, if the joint-angles of manipulator is $\boldsymbol{\theta}_k$. The BAORNN algorithm works by generating a random direction vector $\vec{\mathbf{b}} \in$

Algorithm 1: BAORNN algorithm

Input: A m-DOF robotic arm with unknown kinematic model attached with position and orientation sennsors, a reference trajectory $\mathbf{x}_r(t) \in \mathbb{R}^n$ for end-effector and an objective function $g(\mathbf{x}_r(t), \boldsymbol{\theta}(t))$ quantifying the quality of tracking performance. Additionally, the values of hyper-parameters: c_1 and c_2 defined in (16).

Output: An optimal trajectory $\boldsymbol{\theta}_r(t)$ in joint space.

$\boldsymbol{\theta}_0 \leftarrow$ Initial joint coordinates

$t_{stop} \leftarrow$ Last time instant in $\mathbf{x}_r(t)$

while $t < t_{stop}$ **do**

- Generate a random direction vectors, $\vec{\mathbf{b}} \in \mathbb{R}^m$ in joint space.
- Calculate the location of left and right beetle antennae, $\boldsymbol{\theta}_l$ and $\boldsymbol{\theta}_r$ respectively, using (10).
- Project the calculated points inside the constrained range using (11).
- Move the robotic arm to both configurations and measure the location and orientation of end-effector, $\hat{\mathbf{x}}(\boldsymbol{\theta}(t))$ using pose sensor.
- Calculate the value of objective function at both locations using (12).
- Calculate the new location, $\boldsymbol{\theta}_{new}$, using (13), and project on the constrained set.
- Calculate the value of objective function at new location using the feedback from pose sensor, as given in (14).
- Update the location of beetle in joint space using (15).
- Move the robotic arm to the new joint configuration calculated in the previous step.
- $t \leftarrow t + 1$

end

$\mathbb{R}^m$ corresponding to the direction of the beetle antenna. Based on the generated direction vector, the algorithm calculates the end-point of both beetle antennae and project them inside the constrained space to respect the limit of joint motion as follow,

$$\boldsymbol{\theta}_L = \mathcal{P}(\boldsymbol{\theta}_k + \lambda_k \vec{\mathbf{b}}), \qquad \boldsymbol{\theta}_R = \mathcal{P}(\boldsymbol{\theta}_k - \lambda_k \vec{\mathbf{b}}), \tag{10}$$

where $\mathcal{P}(.)$ is the projection function and project the points to the constrained space defined by the objective function, $\boldsymbol{\theta}_L$ and $\boldsymbol{\theta}_R$ denotes the projected location of left and right beetle antennae respectively. λ_k is an hyper-parameter in BAORNN algorithm and denotes the length of the beetle antenna. Strategy for choosing an optimal value for λ are discussed later. Although there are several ways to define a projection function, in this paper we chose a numerically efficient projection function which is commonly used, i.e., the saturation function. The saturation function is defined as follow

$$\mathcal{P}(\theta) = \begin{cases} \theta^- & \text{if} \quad \theta < \theta^- \\ \theta & \text{if} \quad \theta^- < \theta < \theta^+ \\ \theta^+ & \text{if} \quad \theta^+ < \theta \end{cases}, \tag{11}$$

and the function is sequentially applied to each joint angle.

The robotic arm moves the joints to both of the calculated configuration, $\boldsymbol{\theta}_L$ and $\boldsymbol{\theta}_R$, successively and calculate the value of the objective function using the following relation

$$\begin{aligned} g_L &= g(\mathbf{x}_r(t), \boldsymbol{\theta}_L) = ||\mathbf{x}_r(t) - \hat{\mathbf{x}}(\boldsymbol{\theta}_L)||_2^2 \\ g_R &= g(\mathbf{x}_r(t), \boldsymbol{\theta}_R) = ||\mathbf{x}_r(t) - \hat{\mathbf{x}}(\boldsymbol{\theta}_R)||_2^2, \end{aligned} \tag{12}$$

where g_L and g_R are the values of the objective function at left and right beetle antenna locations, respectively. $\mathbf{x}_r(t)$ is the cartesian coordinate on the reference trajectory at current time instant t and $\hat{\mathbf{x}}(\boldsymbol{\theta})$ is the value available from sensor, hence both values are known and the values of g_L and g_R can be evaluated.

BAORNN algorithm then uses the value of the objective function at both antennae location to take the next step toward a direction in which the value of the objective function is decreasing by using the following update rule

$$\boldsymbol{\theta}_{new} = \mathcal{P}(\boldsymbol{\theta}_k - \delta_k(\lambda_k)\text{sign}(g_L - g_R)\vec{\mathbf{b}}) \tag{13}$$

where $\boldsymbol{\theta}_{new}$ is the new updated location of the robotic arm in the joint space, $\text{sign}(g_L - g_R)\vec{\mathbf{b}}$ ensures that the updated location of the beetle is in the direction of the antenna with small objective function value. $\delta_k(\lambda_k)$ denotes the step-size, i.e., euclidean distance between the updated location $\boldsymbol{\theta}_{new}$ and the current location $\boldsymbol{\theta}_k$. It is a function of antennae length λ_0 and there relationship will also be discussed later. After reaching $\boldsymbol{\theta}_{new}$, the value of objective function is again re-evaluated

$$g_{new} = g(\mathbf{x}_r(t), \boldsymbol{\theta}_{new}) = ||\mathbf{x}_r(t) - \hat{\mathbf{x}}(\boldsymbol{\theta}_{new})||_2^2, \tag{14}$$

the value g_{new} is compared to the value of objective function at initial location $\boldsymbol{\theta}_k$. If the value of the objective function is decreased than the robot remain there, otherwise it moves back to $\boldsymbol{\theta}_k$

$$\boldsymbol{\theta}_{k+1} = \begin{cases} \boldsymbol{\theta}_k, & \text{if} \quad g_{new} \geq g(\mathbf{x}_r(t), \boldsymbol{\theta}_k) \\ \boldsymbol{\theta}_{new}, & \text{if} \quad g_{new} < g(\mathbf{x}_r(t), \boldsymbol{\theta}_k) \end{cases} \tag{15}$$

After moving from $\boldsymbol{\theta}_k$ to $\boldsymbol{\theta}_{k+1}$, the whole process is repeated; a new random direction vector $\vec{\mathbf{b}}$ is generated, antennae location, $\boldsymbol{\theta}_L$ and $\boldsymbol{\theta}_R$, are calculated using (10), objective funtion values, g_L and g_R, are calculated using (12) and the beetle location is update using (15). The algorithm is formally presented in Algorithm 1.

The choice of hyper-parameter λ_k and $\delta_k(\lambda_k)$, where k is the sample index, effects the convergence performance of our proposed algorithm. By empirical observations and analyzing the experimental results, we found that the following relations can provide a reasonable convergence rate which desirable dynamic response from the robot

$$\lambda_k = c_1 \sqrt{g(\mathbf{x}_r(t), \boldsymbol{\theta}_k)} \tag{16}$$

$$\delta_k(\lambda_k) = c_2 \lambda_k \tag{17}$$

where c_1 and c_2 are constants and the factor $\sqrt{g(\boldsymbol{\theta}_i)}$ control the step-size i.e. Euclidean distance between $\boldsymbol{\theta}_{i+1}$ and $\boldsymbol{\theta}_i$ in joint space. Such a choice of step-size ensures large step-size when the end-effector is far from the goal position and make

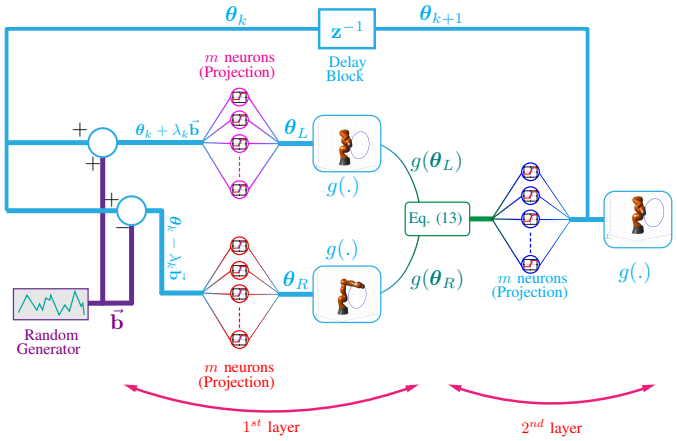


Fig. 2: The topology of the recurrent neural network of the proposed BAORNN algorithm. The diagram illustrates the working of the algorithm formulated in Section III-A.

them extremely small when reached near the goal. The small step-size near goal position reduces the overshooting of the robotic arm near the goal position. For c_1 and c_2, we found that following heuristics provide a good starting point for further tuning there values

$$c_1 \propto T_s, \quad c_2 \in [1, 3]$$

where T_s is the sampling time of the control loop running the proposed algorithm and the robotic arm.

The BAORNN algorithm presented above can be formulated as a recurrent neural network, as shown in Fig. 2. The RNN has a two-layered architecture with $3m$ neurons. The RNN has a temporal feedback connection between the output of the second layer to the input of the first layer. The block "Random" represents a normally distributed random vector generator and provide direction vector $\vec{\mathbf{b}}$ for the BAORNN algorithm. The neurons shown as small circle represent the projection operator $\mathcal{P}$, while the neurons shown as curved rectangular boxes (in cyan) represent the objective function and the output values of those neurons are calculated by using the sensor data coming from the actual robotic arm. It must be noted that the $g(.)$ blocks in layer one is evaluated sequentially based on sensor data from the robotic arm because we are running the algorithm on a real robotic arm at runtime.

By parsing the graph of RNN and counting the number of numeric operations and mathematical function evaluations, it can be seen that the algorithm have a complexity $O(m)$, i.e., the computational complexity is just a linear function of the number of joints. The algorithm involves elementary floating point operations, which can be executed very fast on embedded processors since most of them have dedicated hardware units for floating point operations.

B. Theoritical Analysis

Now we will theoretically analyze the proposed BAORNN algorithm and prove its stability and asymptotical convergence to an optimum solution. The detailed analysis on stability and convergence of BAO algorithm can be found in [68], and here we lay down the sketch of the proof.

Definition 1. *For the tracking control of a robotic arm, starting from an arbitrary initial joint configuration $\boldsymbol{\theta}_0$, the joint space trajectory $\boldsymbol{\theta}_r(t)$ generate by BAORNN algorithm is said to be stable if*

$$g(\mathbf{x}_r(t), \boldsymbol{\theta}_{k+1}) \leq g(\mathbf{x}_r(t), \boldsymbol{\theta}_k), \quad \forall \quad k \geq 0, \tag{18}$$

i.e., objective function is a monotonically decreasing function of the time. $\boldsymbol{\theta}_r(t_k)$ is written as $\boldsymbol{\theta}_k$ for simplicity of notation.

Lemma 1. *For the tracking control of a redundant joint robotic arm, starting from an initial joint configuration $\boldsymbol{\theta}_0$ and end-effector coordinates $\hat{\mathbf{x}}(\boldsymbol{\theta}_0)$, the joint space trajectory $\boldsymbol{\theta}_r(t)$ generated by BAORNN algorithm is stable.*

Proof. See Lemma 1 of [68] for details. The general sketch of the proof goes like this: starting from an initial joint configuration $(\boldsymbol{\theta}_r(t_0)) = \boldsymbol{\theta}_0$ at $t = t_0$, the BAORNN algorithm updates the location of beetle using (15). The update rule only changes the location in joint space if

$$g(\mathbf{x}_r(t), \boldsymbol{\theta}_{new}) < g(\mathbf{x}_r(t), \boldsymbol{\theta}_0)$$

i.e., there is a decrease in the value of objective function at the new location. If on the contrary, the value of objective function further increases as compared to previous time step, i.e., $g(\mathbf{x}_r(t), \boldsymbol{\theta}_{new}) > g(\mathbf{x}_r(t), \boldsymbol{\theta}_0)$, the robotic arm moves the joint back to the configuration of previous time step, i.e., $\boldsymbol{\theta}_1 = \boldsymbol{\theta}_0$. Therefore, the robot configuration at end of each timestep, $\boldsymbol{\theta}_0$, $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$, ..., $\boldsymbol{\theta}_k$, where $t_k = t_{stop}$, always results in either decrease or the same objective function value. Which shows the BAORNN algorithm produces a monotonically decreasing series for the values of objective function and thus stable. □

Definition 2. *For the tracking control of a robotic arm, starting from an arbitrary initial joint configuration $\boldsymbol{\theta}_0$, the end-effector trajectory $\hat{\mathbf{x}}(\boldsymbol{\theta}(t))$ in Cartesian space is said to be convergent to the reference trajectory $\mathbf{x}_r(t)$ if it satisfies*

$$\hat{\mathbf{x}}(\boldsymbol{\theta}(t)) \to \mathbf{x}_r(t), \quad as \quad t \to \infty. \tag{19}$$

Lemma 2. *For the tracking control of a redundant joint robotic arm, starting from an initial joint configuration $\boldsymbol{\theta}_0$ and end-effector coordinates $\hat{\mathbf{x}}(\boldsymbol{\theta}_0)$, the joint space trajectory $\boldsymbol{\theta}_r(t)$ generated by BAORNN algorithm converges the cartesian trajectory end-effector $\hat{\mathbf{x}}(\boldsymbol{\theta}_r(t))$ to the reference trajectory $\mathbf{x}_r(t)$.*

Proof. See Theorem 1 of [68] for details. The general sketch of the proof goes like this: suppose at time instant t_k, probability that the the joint configuration calculated by BAORNN $\boldsymbol{\theta}_k$ does not lie in an optimal direction is given by p_k, where $0 < p_k < 1$ by the definition of probability. The probability that $\boldsymbol{\theta}_{k+1}$ also does not lie in an optimal direction is same as the probability that none of the previous points lies in the optimal direction, i.e., given by the product of those probabilities

$$p_{k+1} = p_0 p_1 p_2 ... p_k$$

the probability that $\boldsymbol{\theta}_{k+1}$ is actually an optimal solution is

$$P_{k+1} = 1 - p_{k+1} = 1 - p_0 p_1 p_2 ... p_k$$

where P_{k+1} is the probability of $\boldsymbol{\theta}_{k+1}$ being an optimal solution at time instant t_k. Since all the values of $p_0, p_1, p_2, ..., p_k$ lies in the range $[0, 1]$, and product of values in range $[0, 1]$ diminishes as the elements of product increase, therefore,

$$\lim_{k\to\infty} P_{k+1} = \lim_{k\to\infty} (1 - p_0 p_1 p_2 ... p_k) = 1 - \lim_{k\to\infty} p_0 p_1 p_2 ... p_k = 1$$

Therefore as $k \to \infty$, the joint configuration $\boldsymbol{\theta}_k$ calculated by BAORNN converges to the optimal joint configuration. □

IV. Simulation Results & Discussion

In this section, simulation methodology and results for position tracking control are presented. KUKA LBR IIWA-14 robotic arm is used as the testbench. The IIWA-14 robotic arm has 7-DOF, i.e., $m = 7$, and all joints are revolute. The robotic arm is shown in Fig. 3. Table 3b shows mechanical information about IIWA-14.

A. Simulation Methodology

We used the model of IIWA-14 robot provided by MATLAB Robotic System Toolbox [75]. The model provides an excellent representation of the real-world IIWA-14 robotic arm and therefore act as a desirable simulation testbed to test the performance of the proposed algorithm in realistic scenarios. A visual representation of the model is shown in Fig. 3.

For testing the performance of position tracking control, we used two reference trajectory by following the method of [29]. The first reference trajectory is a rectangular path, and the second is a circular one. Following time-varying system of equation defines the rectangular trajectory used in the simulations.

$$\mathbf{x}_r^{\text{rect}}(t) = \begin{cases} \frac{t_0 - t}{t_0}\vec{\mathbf{a}} + \frac{t}{t_0}\vec{\mathbf{b}} & \text{when} \quad 0 \le t \le t_0 \\ \frac{t_1 - t}{t_1 - t_0}\vec{\mathbf{b}} + \frac{t - t_0}{t_1 - t_0}\vec{\mathbf{c}} & \text{when} \quad t_0 < t \le t_1 \\ \frac{t_2 - t}{t_2 - t_1}\vec{\mathbf{c}} + \frac{t - t_1}{t_2 - t_1}\vec{\mathbf{d}} & \text{when} \quad t_1 < t \le t_2 \\ \frac{t_3 - t}{t_3 - t_2}\vec{\mathbf{d}} + \frac{t - t_2}{t_3 - t_2}\vec{\mathbf{a}} & \text{when} \quad t_2 < t \le t_3. \end{cases} \tag{20}$$

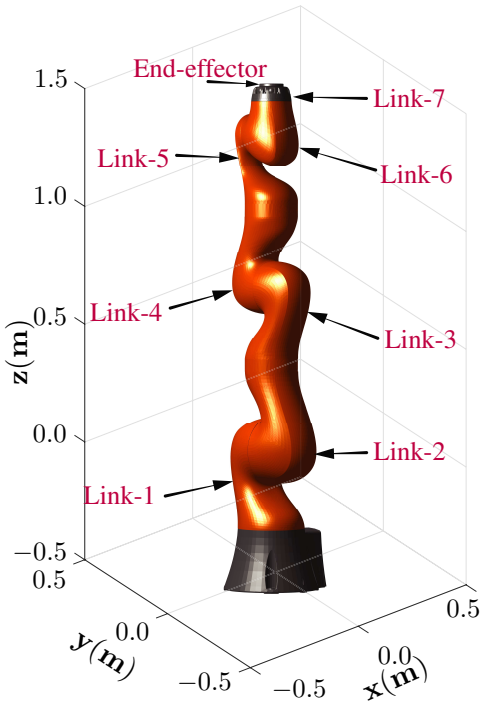
where $\mathbf{x}_r^{\text{rect}}(t) \in \mathbb{R}^3$ represent the rectangular trajectory, $\vec{\mathbf{a}}$, $\vec{\mathbf{b}}$, $\vec{\mathbf{c}}$ and $\vec{\mathbf{d}}$ denotes the vertices of the rectangular trajectory such that $\mathbf{x}_r^{\text{rect}}(0) = \vec{\mathbf{a}}$, $\mathbf{x}_r^{\text{rect}}(t_0) = \vec{\mathbf{b}}$, $\mathbf{x}_r^{\text{rect}}(t_1) = \vec{\mathbf{c}}$, $\mathbf{x}_r^{\text{rect}}(t_2) = \vec{\mathbf{d}}$, and $\mathbf{x}_r^{\text{rect}}(t_3) = \vec{\mathbf{a}}$. The rectangular trajectory spans total time duration $[0, t_3]$. For actual trajectory generation, we used the following values: $\vec{\mathbf{a}} = [-0.5\ 0.5\ 0.6]^T$, $\vec{\mathbf{b}} = [0.5\ 0.5\ 0.6]$, $\vec{\mathbf{c}} = [0.5\ 0.5\ 0.3]$, $\vec{\mathbf{d}} = [-0.5\ 0.5\ 0.3]$, and the total time duration is $[0, 30]$. These values represent a rectangular path in $x - z$ plane.

For generating the circular reference trajectory, we used the following parametric equation

$$\mathbf{x}_r^{\text{circle}}(t) = \vec{\mathbf{c}} + r\cos(2\pi t/T)\vec{\mathbf{a}} + r\sin(2\pi t/T)\vec{\mathbf{b}}. \tag{21}$$

where $\vec{\mathbf{c}}$ represents the center of the circle, $\vec{\mathbf{a}}$ and $\vec{\mathbf{b}}$ represents two perpendicular unit vectors defining the plane of the circle in 3D space. T denotes the total time duration in which robotic arm traces the complete circular trajectory. For actual trajectory generation, we used the following values: $\vec{\mathbf{a}} = [0.0\ 0.5\ 0.5]$, $\vec{\mathbf{b}} = [1\ 0\ 0]$, and $\vec{\mathbf{c}} = [0\ 1\ 0]$. These values represent a circlular path in $x - z$ plane at $y = 0.5$.

Note that the above mentioned two trajectories are chosen, because they encompass a large variety of motion required by the manipulator, i.e., straight lines and circular arcs. However, without the loss of generality, the proposed algorithm works for an arbitrarily chosen trajectory, as long as it does not violate the mechanical joint-limits of the manipulator.



(a)

Link	Motion range (degrees)	Length	Home position†
Link-1	±170	0.1575	$[0.0\ \ 0.0\ \ 0.1575]^T$
Link-2	±120	0.2025	$[0.0\ \ 0.0\ \ 0.3600]^T$
Link-3	±170	0.2045	$[0.0\ \ 0.0\ \ 0.5645]^T$
Link-4	±120	0.2155	$[0.0\ \ 0.0\ \ 0.7800]^T$
Link-5	±170	0.1845	$[0.0\ \ 0.0\ \ 0.9645]^T$
Link-6	±120	0.2155	$[0.0\ \ 0.0\ \ 1.1800]^T$
Link-7	±170	0.0810	$[0.0\ \ 0.0\ \ 1.2610]^T$
end-effector	fixed	0.0450	$[0.0\ \ 0.0\ \ 1.3060]^T$

† Link position when all joint angles are zero.

(b)

Fig. 3: KUKA LBR IIWA-14 robotic arm with 7-DOF revolute joints. (a) shows the 3D model of the robotic arm in its home configuration. (b) DH-parameters of manipulator.

B. Trajectory Tracking Results

For the rectangular trajectory tracking problem, the results are shown in Fig. 4. $c_1 = 1$ and $c_2 = 3$ were used as the values of hyperparameters in (16). During simulations, the initial configuration of the joints is assumed to be home configuration, i.e., all joint angles are zero. Fig. 4a shows the motion of the links in the robotic arm while tracking the reference trajectory (shown in blue). It shows the motion

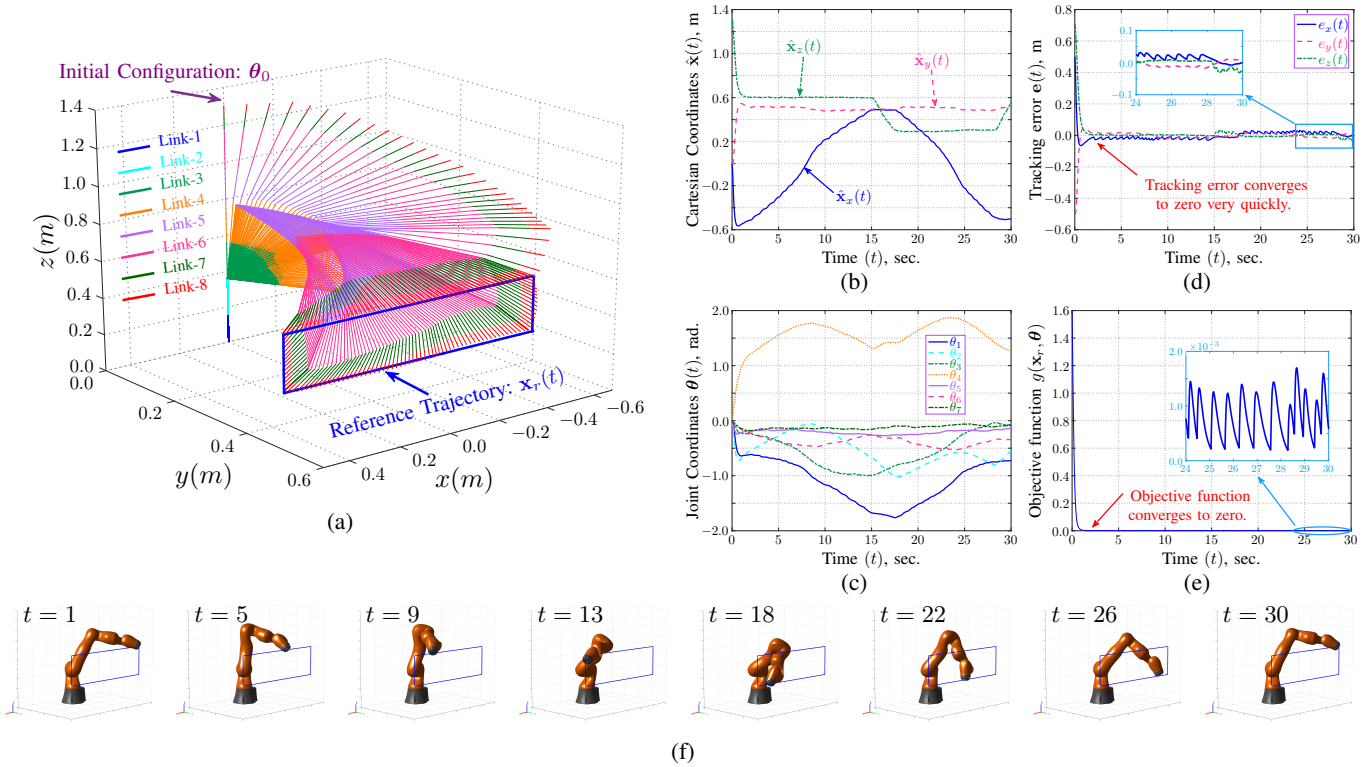


Fig. 4: Simulation results for rectangular trajectory tracking using the BAORNN algorithm. (a) Motion of the links of robotic arm while tracking the reference trajectory. (b) Profile of cartesian coordinates of the end-effector trajectory. (c) Profile of joint space trajectory of the robotic arm. (d) Profile of the position tracking error. (e) Convergence of the value of objective function, as defined in (7). (f) Simulation model of the LBE IIWA-14 robot while tracking the reference trajectory.

of the robotic arm starting from the initial configuration; therefore, the initial portion of the motion lies outside the reference trajectory. Once the end-effector reaches the reference trajectory, its remaining motion closely matches the reference signal. Fig. 4b shows the cartesian position coordinates of the end-effector while tracing the rectangular trajectory. Fig. 4c shows the motion of the joints of the robotic arm. Since all the joints are revolute, therefore it shows the rotation of each joint in radians. It is worth pointing out that the trajectories in cartesian space and joint space, as shown in Fig. 4b and 4c respectively shows a bit of unsmooth behaviour. Such behavior is characteristic for metaheuristic algorithms because of the stochastic nature; however, the resultant gain because of reduced computational cost is much greater. Fig. 4d shows the position tracking error. It can be seen that the tracking error quickly converges to a very small value. At $t = 0$, the tracking error shows a large value, $\approx [0.5 \;\; -0.5 \;\; 0.7]^T$; this happened because the robotic arm starts its motion from home configuration, which is very far from the rectangular trajectory. However, as time progresses, the end-effector finally converges to the reference trajectory, and the tracking error diminishes. This also proves that the proposed algorithm does not suffer from PEA problem; once the tracking error diminishes, it never becomes large again, except for some small ripples caused by the stochastic nature of the algorithm. Similarly, Fig. 4e shows the evolution of the value of objective function, defined in (7), with time. Similar to the tracking error, the objective function shows a very large value in the beginning, i.e., $g(\boldsymbol{\theta}_0) \approx 1.6$. However, as time progresses, the objective function value converges in the range $[0 \;\; 2 \times 10^{-3}]$ and remain inside this interval.

The tracking results for the circular trajectory are shown in Fig. 5. Similar to the case of rectangular trajectory, $c_1 = 1$ and $c_2 = 3$ were used as the values of hyperparameters in (16). Fig. 5a shows the motion of the links in the robotic arm while. Similar to the case of rectangular trajectory, the initial portion of the robotic arm motion is also shown. The end-effector trajectory finally converges to the circular trajectory. Fig. 5b shows the cartesian position coordinates of the end-effector and Fig. 5c shows joint space motion of the robotic arm joints. The position tracking error shown in Fig. 5d displays the same trend, i.e., the error is large at the beginning but diminishes with time. It again proves that the proposed algorithm does not suffer from PEA problem. The objective function profile shown in Fig. 5e also exhibit the decaying trend. Similarity in the simulation results for the circular and rectangular trajectory confirms the stability, and convergence performance of the BAORNN algorithm. Fig. 5f shows the images of the simulation models of the robotic arm while tracking the reference trajectory. These simulation results prove the stability and converge of the BAORNN algorithm. Fig. 4f shows the images of the simulation models of the robotic arm while tracking the reference trajectory.

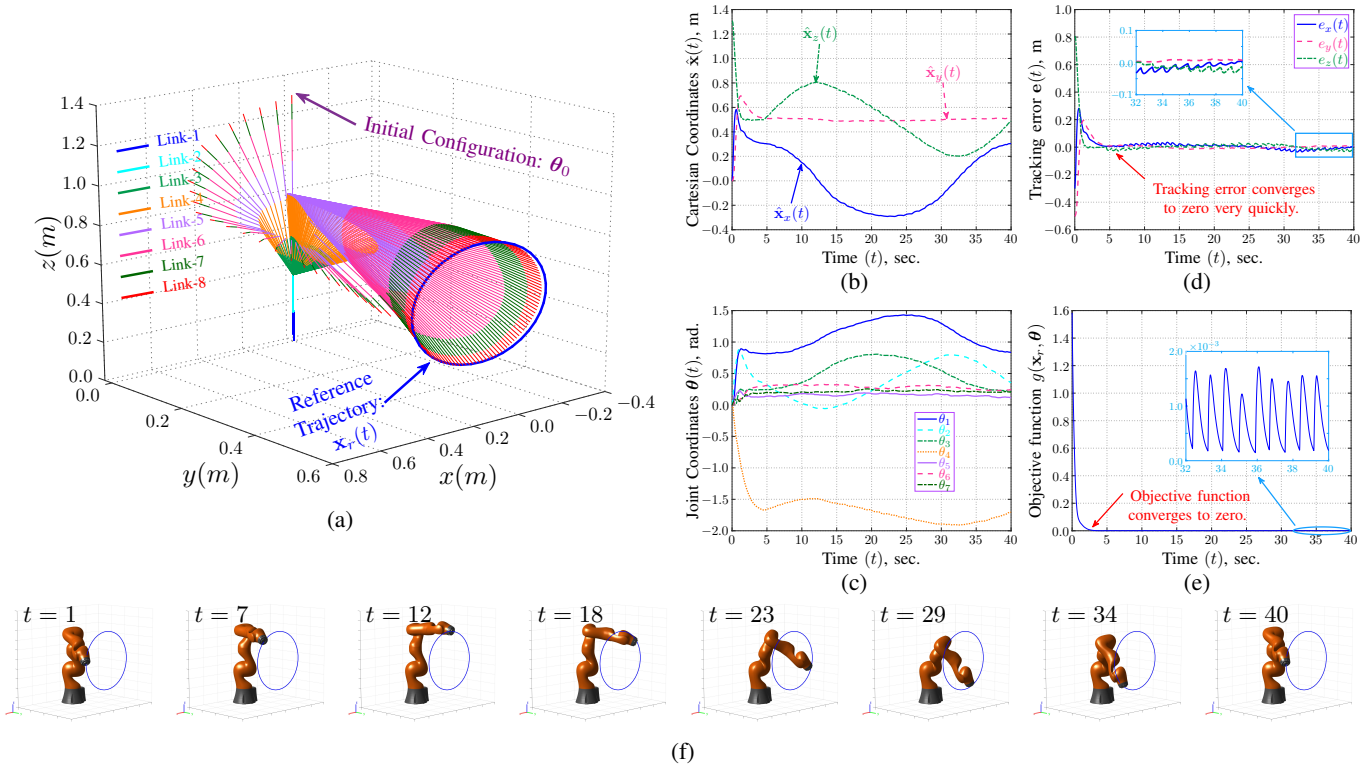


Fig. 5: Simulation results for circular trajectory tracking using the BAORNN algorithm. (a) Motion of the links of robotic arm while tracking the reference trajectory. (b) Profile of cartesian coordinates of the end-effector trajectory. (c) Profile of joint space trajectory of the robotic arm. (d) Profile of the position tracking error. (e) Convergence of the value of objective function, as defined in (7). (f) Simulation model of the LBE IIWA-14 robot while tracking the reference trajectory.

V. Conclusion

In this paper, we have addressed the problem of redundancy resolution and kinematic tracking control for redundant robotic manipulator working in an industrial environment. The proposed algorithm solves these problems in a model-free strategy which does not require the estimation of the kinematic model or the Jacobian of the robotic manipulator. The proposed BAORNN uses a RNN architecture based on BAO algorithm to solve the redundancy resolution problem and achieve kinematic tracking control in runtime. The proposed algorithm only rely on the values from the position and orientation sensor attached to the end-effector of the robot and does not require any knowledge about the mechanical parameters or physical construction of the robot, making the proposed algorithm resilient to the model-uncertainties and PEA as compared to the traditional algorithm. The proposed algorithm is also computationally efficient because it does not require pseudo-inverse of the Jacobian matrix. The stability and convergence of the proposed algorithm are theoretically proven. Simulations on a 7-DOF industrial robotic arm also prove the performance of the proposed algorithm.

VI. Future Work

A potential future research direction will include reducing the motion of the manipulator by computing the end-effector pose using the manipulator's model instead of moving the manipulator's joints. In the current version of the BAORNN algorithm, the manipulator has to move to each location, and the value of end-effector coordinates is feedback through the pose sensor. Additionally, current work only considers the tracking control as the objective of the industrial manipulator; however, an actual industrial manipulator needs to fulfill several additional objectives, e.g., obstacle avoidance. Future work will study the effect of including secondary objectives. Experimental verification of the proposed algorithm is also under consideration for the future work.

References

[1] X.-S. Yang, *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[2] J. A. Parejo, A. Ruiz-Cortés, S. Lozano, and P. Fernandez, "Metaheuristic optimization frameworks: a survey and benchmarking," *Soft Computing*, vol. 16, no. 3, pp. 527–561, 2012.

[3] I. Fister Jr, X.-S. Yang, I. Fister, J. Brest, and D. Fister, "A brief review of nature-inspired algorithms for optimization," *arXiv preprint arXiv:1307.4186*, 2013.

[4] J. Krause, J. Cordeiro, R. S. Parpinelli, and H. S. Lopes, "A survey of swarm algorithms applied to discrete optimization problems," in *Swarm Intelligence and Bio-Inspired Computation*, pp. 169–191, Elsevier, 2013.

[5] M. Dorigo and G. Di Caro, "Ant colony optimization: a new meta-heuristic," in *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, vol. 2, pp. 1470–1477, IEEE, 1999.

[6] S. Nakrani and C. Tovey, "On honey bees and dynamic server allocation in internet hosting centers," *Adaptive Behavior*, vol. 12, no. 3-4, pp. 223–240, 2004.

[7] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *International symposium on stochastic algorithms*, pp. 169–178, Springer, 2009.

[8] X.-S. Yang and S. Deb, "Engineering optimisation by cuckoo search," *arXiv preprint arXiv:1005.2908*, 2010.

[9] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pp. 65–74, Springer, 2010.

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

[11] L. Johannsmeier and S. Haddadin, "A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 41–48, 2016.

[12] Y. M. Zhao, Y. Lin, F. Xi, and S. Guo, "Calibration-based iterative learning control for path tracking of industrial robots," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 5, pp. 2921–2929, 2014.

[13] M. Tarokh and X. Zhang, "Real-time motion tracking of robot manipulators using adaptive genetic algorithms," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 3-4, pp. 697–708, 2014.

[14] Z. Zhang, A. Beck, and N. Magnenat-Thalmann, "Human-like behavior generation based on head-arms model for robot tracking external targets and body parts," *IEEE transactions on cybernetics*, vol. 45, no. 8, pp. 1390–1400, 2014.

[15] C. Yang, Y. Jiang, Z. Li, W. He, and C.-Y. Su, "Neural control of bimanual robots with guaranteed global stability and motion precision," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1162–1171, 2016.

[16] K. Tchoń, A. Ratajczak, and I. Góral, "Lagrangian jacobian inverse for nonholonomic robotic systems," *Nonlinear Dynamics*, vol. 82, no. 4, pp. 1923–1932, 2015.

[17] Y.-J. Liu and S. Tong, "Adaptive nn tracking control of uncertain nonlinear discrete-time systems with nonaffine dead-zone input," *IEEE Transactions on Cybernetics*, vol. 45, no. 3, pp. 497–505, 2014.

[18] G. Hu, N. Gans, N. Fitz-Coy, and W. Dixon, "Adaptive homography-based visual servo tracking control via a quaternion formulation," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 1, pp. 128–135, 2009.

[19] H. M. La, T. H. Dinh, N. H. Pham, Q. P. Ha, and A. Q. Pham, "Automated robotic monitoring and inspection of steel structures and bridges," *Robotica*, vol. 37, no. 5, pp. 947–967, 2019.

[20] 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.

[21] 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.

[22] 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.

[23] A. Goldenberg, B. Benhabib, and R. Fenton, "A complete generalized solution to the inverse kinematics of robots," *IEEE Journal on Robotics and Automation*, vol. 1, no. 1, pp. 14–20, 1985.

[24] 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.

[25] D. Guo and Y. Zhang, "Acceleration-level inequality-based man scheme for obstacle avoidance of redundant robot manipulators," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 12, pp. 6903–6914, 2014.

[26] F. Basile, F. Caccavale, P. Chiacchio, J. Coppola, and C. Curatella, "Task-oriented motion planning for multi-arm robotic systems," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 5, pp. 569–582, 2012.

[27] H. M. La, R. Lim, and W. Sheng, "Multirobot cooperative learning for predator avoidance," *IEEE Trans. on Control Syst. Technology*, vol. 23, no. 1, pp. 52–63, 2014.

[28] A. H. Khan, S. Li, and X. Luo, "Obstacle avoidance and tracking control of redundant robotic manipulator: An rnn based metaheuristic approach," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2019. Early Access.

[29] S. Li, S. Chen, B. Liu, Y. Li, and Y. Liang, "Decentralized kinematic control of a class of collaborative redundant manipulators via recurrent neural networks," *Neurocomputing*, vol. 91, pp. 1–10, 2012.

[30] N. C. N. Doan, P. Y. Tao, and W. Lin, "Optimal redundancy resolution for robotic arc welding using modified particle swarm optimization," in *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 554–559, IEEE, 2016.

[31] 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.

[32] C. Yang, H. Wu, Z. Li, W. He, N. Wang, and C.-Y. Su, "Mind control of a robotic arm with visual fusion technology," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 3822–3830, 2017.

[33] W. He, H. Huang, and S. S. Ge, "Adaptive neural network control of a robotic manipulator with time-varying output constraints," *IEEE transactions on cybernetics*, vol. 47, no. 10, pp. 3136–3147, 2017.

[34] B. Liao and W. Liu, "Pseudoinverse-type bi-criteria minimization scheme for redundancy resolution of robot manipulators," *Robotica*, vol. 33, no. 10, pp. 2100–2113, 2015.

[35] L. Jin and Y. Zhang, "Discrete-time zhang neural network of o (τ3) pattern for time-varying matrix pseudoinversion with application to manipulator motion generation," *Neurocomputing*, vol. 142, pp. 165–173, 2014.

[36] A. Liegeois, "Automatic supervisory control of the configuration and behaviour of multibody mechanisms," *IEEE Transactions on systems, man and cybernetics*, vol. 7, no. 12, pp. 868–871, 1977.

[37] C. A. Klein and C.-H. Huang, "Review of pseudoinverse control for use with kinematically redundant manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 2, pp. 245–250, 1983.

[38] Y. Zhang, S. Li, J. Zou, and A. H. Khan, "A passivity-based approach for kinematic control of redundant manipulators with constraints," *IEEE Transactions on Industrial Informatics*, 2019. Early Access.

[39] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*, pp. 396–404, Springer, 1986.

[40] Y. Zhang, S. S. Ge, and T. H. Lee, "A unified quadratic-programming-based dynamical system approach to joint torque optimization of physically constrained redundant manipulators," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 5, pp. 2126–2132, 2004.

[41] H. Wang and S. Kang, "Adaptive neural command filtered tracking control for flexible robotic manipulator with input dead-zone," *IEEE Access*, vol. 7, pp. 22675–22683, 2019.

[42] D. Guo and Y. Zhang, "Acceleration-level inequality-based man scheme for obstacle avoidance of redundant robot manipulators," *IEEE Trans. on Ind. Electron.*, vol. 61, no. 12, pp. 6903–6914, 2014.

[43] H. Ding and S. K. Tso, "A fully neural-network-based planning scheme for torque minimization of redundant manipulators," *IEEE Transactions on Industrial Electronics*, vol. 46, no. 1, pp. 199–206, 1999.

[44] 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.

[45] L. Xiao, S. Li, F.-J. Lin, Z. Tan, and A. H. Khan, "Zeroing neural dynamics for control design: comprehensive analysis on stability, robustness, and convergence speed," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 2605–2616, 2018.

[46] S.-H. Cha, T. Lasky, and S. Velinsky, "Kinematic redundancy resolution for serial-parallel manipulators via local optimization including joint constraints," *Mechanics based design of structures and machines*, vol. 34, no. 2, pp. 213–239, 2006.

[47] S. Tejomurtula and S. Kak, "Inverse kinematics in robotics using neural networks," *Information Sciences*, vol. 116, no. 2-4, pp. 147–164, 1999.

[48] A.-V. Duka, "Neural network based inverse kinematics solution for trajectory tracking of a robotic arm," *Procedia Technology*, vol. 12, pp. 20–27, 2014.

[49] D. Pham, M. Castellani, and A. Fahmy, "Learning the inverse kinematics of a robot manipulator using the bees algorithm," in *2008 6th IEEE International Conference on Industrial Informatics*, pp. 493–498, IEEE, 2008.

[50] B. Luo, D. Liu, T. Huang, and D. Wang, "Model-free optimal tracking control via critic-only q-learning," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 10, pp. 2134–2144, 2016.

[51] T. R. Wanasinghe, G. K. Mann, and R. G. Gosine, "A jacobian free approach for multi-robot relative localization," in *2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–6, IEEE, 2014.

[52] J. Na, X. Ren, and D. Zheng, "Adaptive control for nonlinear pure-feedback systems with high-order sliding mode observer," *IEEE transactions on neural networks and learning systems*, vol. 24, no. 3, pp. 370–382, 2013.

[53] W. He, Z. Yin, and C. Sun, "Adaptive neural network control of a marine vessel with constraints using the asymmetric barrier lyapunov function," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1641–1651, 2016.

[54] C. Yang, Y. Jiang, J. Na, Z. Li, L. Cheng, and C.-Y. Su, "Finite-time convergence adaptive fuzzy control for dual-arm robot with unknown kinematics and dynamics," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 3, pp. 574–588, 2018.

[55] J. Na, B. Jing, Y. Huang, G. Gao, and C. Zhang, "Unknown system dynamics estimator for motion control of nonlinear robotic systems," *IEEE Transactions on Industrial Electronics*, 2019.

[56] H. Wang, Y. Zou, P. X. Liu, and X. Liu, "Robust fuzzy adaptive funnel control of nonlinear systems with dynamic uncertainties," *Neurocomputing*, vol. 314, pp. 299–309, 2018.

[57] C. Yang, Y. Jiang, W. He, J. Na, Z. Li, and B. Xu, "Adaptive parameter estimation and control design for robot manipulators with finite-time convergence," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 10, pp. 8112–8123, 2018.

[58] J. Na, M. N. Mahyuddin, G. Herrmann, X. Ren, and P. Barber, "Robust adaptive finite-time parameter estimation and control for robotic systems," *International Journal of Robust and Nonlinear Control*, vol. 25, no. 16, pp. 3045–3071, 2015.

[59] C. Blum, A. Roli, and M. Sampels, *Hybrid metaheuristics: an emerging approach to optimization*, vol. 114. Springer, 2008.

[60] Z. Ren, P. Li, J. Fang, H. Li, and Q. Chen, "Sba: an efficient algorithm for address assignment in zigbee networks," *Wireless personal communications*, vol. 71, no. 1, pp. 719–734, 2013.

[61] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM computing surveys (CSUR)*, vol. 35, no. 3, pp. 268–308, 2003.

[62] J. Fang, L. Zhang, and H. Li, "Two-dimensional pattern-coupled sparse bayesian learning via generalized approximate message passing," *IEEE Transactions on Image Processing*, vol. 25, no. 6, pp. 2920–2930, 2016.

[63] J. Fang and H. Li, "Distributed estimation of gauss-markov random fields with one-bit quantized data," *IEEE Signal Processing Letters*, vol. 17, no. 5, pp. 449–452, 2010.

[64] J. Fang, Y. Shen, F. Li, H. Li, and Z. Chen, "Support knowledge-aided sparse bayesian learning for compressed sensing," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3786–3790, IEEE, 2015.

[65] Z. Zhu, Z. Zhang, W. Man, X. Tong, J. Qiu, and F. Li, "A new beetle antennae search algorithm for multi-objective energy management in microgrid," in *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 1599–1603, IEEE, 2018.

[66] Q. Wu, X. Shen, Y. Jin, Z. Chen, S. Li, A. H. Khan, and D. Chen, "Intelligent beetle antennae search for uav sensing and avoidance of obstacles," *Sensors*, vol. 19, no. 8, p. 1758, 2019.

[67] S. Dereli and R. Köker, "A meta-heuristic proposal for inverse kinematics solution of 7-dof serial robotic manipulator: quantum behaved particle swarm algorithm," *Artificial Intelligence Review*, pp. 1–16, 2019.

[68] Y. Zhang, S. Li, and B. Xu, "Convergence analysis of beetle antennae search algorithm and its applications," *arXiv preprint arXiv:1904.02397*, 2019.

[69] H. Zhang, H. Jin, Z. Liu, Y. Liu, Y. Zhu, and J. Zhao, "Real-time kinematic control for redundant manipulators in a time-varying environment: Multiple-dynamic obstacle avoidance and fast tracking of a moving object," *IEEE Trans. on Ind. Informatics*, 2019.

[70] P. Qi, C. Liu, A. Ataka, H.-K. Lam, and K. Althoefer, "Kinematic control of continuum manipulators using a fuzzy-model-based approach," *IEEE Trans. on Ind. Electron.*, vol. 63, no. 8, pp. 5022–5035, 2016.

[71] G. Wu, "Kinematic analysis and optimal design of a wall-mounted four-limb parallel schönflies-motion robot for pick-and-place operations," *Journ. of Intelligent & Robotic Syst.*, vol. 85, no. 3-4, pp. 663–677, 2017.

[72] A. Menon, R. Prakash, and L. Behera, "Adaptive critic based optimal kinematic control for a robot manipulator," in *2019 Intl. Conf. on Robot. and Autom. (ICRA)*, pp. 1316–1322, IEEE, 2019.

[73] O. Hock and J. Sedo, "Inverse kinematics using transposition method for robotic arm," in *2018 ELEKTRO*, pp. 1–5, IEEE, 2018.

[74] I. Al-Naimi, A. Taeim, and N. Alajdah, "Fully-automated parallel-kinematic robot for multitask ind. operations," in *2018 15th Intl. Multi-Conf. on Syst., Signals & Devices*, pp. 390–395, IEEE, 2018.

[75] P. I. Corke *et al.*, "A robotics toolbox for matlab," *IEEE Robotics & Automation Magazine*, vol. 3, no. 1, pp. 24–32, 1996.