Obstacle Avoidance and Tracking Control of Redundant Robotic Manipulator: An RNN based Metaheuristic Approach

Ameer Hamza Khan, *Student Member, IEEE,* Shuai Li, *Senior Member, IEEE,* and Xin Luo, *Senior Member, IEEE*

***Abstract*—This paper presents a metaheuristic-based control framework, called Beetle Antennae Olfactory Recurrent Neural Network (BAORNN), for simultaneous tracking control and obstacle avoidance of a redundant manipulator. The ability to avoid obstacles while tracking a predefined reference path is critical for any industrial manipulator. The formulated control framework unifies the tracking control and obstacle avoidance into a single constrained optimization problem by introducing a penalty term into the objective function, which actively rewards the optimizer for avoiding the obstacles. One of the significant features of the proposed framework is the way that the penalty term is formulated following a straightforward principle: maximize the minimum distance between manipulator and obstacle. The distance calculations are based on GJK (Gilbert-Johnson-Keerthi) algorithm, which calculates the distance between manipulator and obstacle by directly using their 3D-geometries. Which also implies that our algorithm works for arbitrarily shaped manipulator and obstacle. Theoretical treatment proves the stability and convergence, and simulations results using LBR IIWA 7-DOF manipulator are presented to analyze the performance of the proposed framework.**

***Index Terms*—Obstacle Avoidance, Tracking Control, RNN, Metaheuristic Optimization.**

I. Introduction

For a redundant robotic manipulator, the problem of tracking control and obstacle avoidance aims at computing an optimal control action to steer the end-effector along a required reference trajectory, while avoiding obstacles present in the environment. With the advances in robotics, the robotic manipulators have found increased research attention from academia as well as from industry [1]–[4]. Industries are interested in using the manipulators to automate the common tasks, e.g., moving, assembling, packing, and transporting the products. Accurate tracking control, along with obstacle avoidance, is a critical requirement for the industrial manipulators [5], [6]. To fulfill those requirements, redundant manipulators [7] are particularly desirable because the extra degree of freedoms (DOFs) provided by redundant joints helps in achieving secondary design objectives, such as obstacle avoidance [8]–[10]. It is well-known in the literature that the tracking control and obstacle avoidance in itself are challenging problems [11]. Unifying these two problems into single framework present an intricate technical challenge.

Several aspects of industrial manipulators have been extensively studied in the academic literature. Apart from tracking control algorithms, particular emphasis has also been placed on designing optimal task-space trajectories for the manipulator as well as analyzing the repeatability of the controller to repeatedly track the generated trajectory have been of great interest [12], [13]. For example, one of the traditional control algorithm, called Jacobian-matrix-pseudo-inverse (JMPI), was shown to have poor repeatability [14]. Jerzy *et al.* [15] proposed a systematic procedure to measure the pose repeatability of an industrial manipulator and discussed the concerning factors additional to the control algorithm, e.g., mechanical and thermal strain. Similarly, several algorithms have been proposed to increase the repeatability of the manipulator during long-term operation [16]. Other approaches to improve the repeatability of the manipulator involves the learning algorithm to estimate the kinematic model of the manipulator in real-time [17]. The learning algorithm continually adapts to variation in the system model and compensate for them in real-time. Similarly, visual Servoing based approaches have also been proposed to use computer vision algorithms in improving the control of industrial manipulators [18].

Kinematic tracking control of a redundant robotic manipulators is a well-studied problem in robotic literature [7], [10], [19]. For example, consider an industrial manipulator, assigned to move an object from one point to another by following a specified trajectory in the cartesian task-space. For a redundant manipulator, corresponding to a given trajectory in cartesian space, infinite numbers of trajectories exist in joint-space. Traditionally, Jacobian-matrix-pseudo-inverse (JMPI) [20] is to used resolve the redundancy. However, JMPI can only be used to solve equality constraint and therefore, does not respect the joint-angle limits. Additionally, it cannot accommodate obstacle avoidance, which usually modeled as inequality constraints [11], [21]. Furthermore, the calculation of pseudo-inverse of Jacobian is a computationally extensive task. Modern approaches to redundancy resolution model the kinematic control as a constrained optimization problem [8]–[10]. These optimization-centric approaches are capable

A.H. Khan is with Department of Computing, Hong Kong Polytechnic University, Hong Kong (email: ameer.h.khan@connect.polyu.hk).

S. Li is with Swansea University, Swansea, UK (email: shuaili@ieee.org).

X. Luo is with the Chongqing Engineering Research Center of Big Data Application for Smart Cities, and Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China (email: luoxin21@cigit.ac.cn).

S. Li and X. Luo are the corresponding authors.

of solving additional inequality constraint simultaneously with the tracking control problem. For example, Wei *et al.* [22] and Wang *et al.* [23] used it for tracking control of manipulators with flexible joints. Li *et al.* [19] proposed a dual Recurrent Neural Network (RNN) for solving the tracking-control optimization problem for multiple manipulators in real-time. Adaptive control techniques have also been proposed in literature [24]–[27] which estimate the system model in real-time to increases the robustness of the tracking controller.

Obstacle avoidance is also an essential goal, along with the tracking control of the robotic manipulator [11]. The industrial robots often need to operate in a complex environment and interact with other robots and objects present in the surrounding. A traditional method for obstacle avoidance uses the concept of "artificial force field", like one proposed by Khatib [28] in which the goal position act as attractive pole whereas obstacles act as repulsive poles. However, their proposed algorithm is formulated in cartesian space and needs further computation for calculating the necessary control actions in the joint-space. Similarly, Flacco *et al.* [29] proposed an algorithm based on the robot-obstacle distance information obtained using a depth sensor. Guo and Zhang [21] proposed an approach at joint-acceleration level approach by minimizing the joint-acceleration norm. Zhang *et al.* [11] proposed a tracking control and obstacle avoidance algorithm, however, there proposed algorithm treats the obstacle as a point object and does not account for 3D-geometry of the manipulator and the obstacle. The traditional methods mostly incorporate obstacle avoidance as inequality constraint in the optimization problem. These constraints do not actively reward the optimizer for avoiding obstacles and only act passively. Our proposed problem formulation tries to overcome this issue by using a penalty term in the objective function. To summarize, the problems being addressed in this paper are

1) Formulating a tracking controller for a redundant manipulator to compute the necessary control actions in joint-space to track a specified task-space trajectory.
2) While tracking the reference trajectory, the controller should satisfy the joint-angle limits.
3) The objects present in the surrounding of the manipulator are considered obstacles, and their collision with the manipulator should be avoided.

In this paper, we take advantage of the fact that optimization-centric approach allows any arbitrary goal to be achieved by adequately formulating the objective function [8], [9]. We incorporate the obstacle avoidance problem into the tracking control framework by adding a penalty term in the objective function along with an inequality constraint. The penalty term approach used in our paper actively reward the optimizer for avoiding the obstacle, which is in contrast with the traditional obstacle avoidance approaches, which simply add an inequality constraint [11]. With this, the obstacle avoidance and tracking control problem essentially reduces to solving the formulated optimization problem in real-time. Our formulated objective function has two goals: 1) Tracking control, i.e., minimize the Euclidean distance between reference trajectory and the end-effector's trajectory, 2) Maximize the distance between the links of the manipulator and the obstacles. We used the Gilbert-Johnson-Keerthi (GJK) algorithm [30], to calculate the distance between manipulator's links and the obstacle by directly using their 3D-geometries.

To solve the optimization problem in real-time, we take a metaheuristic approach; called Beetle Antennae Olfactory Recurrent Neural Network (BAORNN). We leverage the metaheuristic optimization algorithms, which are well-known [31], [32] for their ability to efficiently solve the complex nonlinear non-convex optimization problem. Our proposed algorithm is based on a nature-inspired metaheuristic optimization algorithm; Beetle Antennae Olfactory (BAO) algorithm [33], [34], inspired by the food searching behavior of beetles. Although recently introduced, BAO has shown practical applications in several real-world scenarios [35], [36] and therefore, the reason for our choice for solving the formulated optimization problem. Specifically, The formulation of the BAO algorithm allows the use of the "virtual robots", which virtually anticipate the consequences of joint-actions and only move the real robot when accuracy and collision-safety are guaranteed. We modeled the BAO algorithm as a Recurrent Neural Network (RNN) which enables fast prototyping and will be able to leverage the hardware acceleration, distributed processing, and software optimizations, offered by modern computing frameworks when implemented in an industrial setting.

It should be further noted that the tracking controller presented in this paper is designed on position-level as opposed to velocity-level as done by the most traditional works on tracking control of redundant manipulators [11], [17], [19]. This approach is advantageous because it does not require that initial position of end-effector to lie on the reference trajectory, whereas the velocity-level controllers explicitly require moving the end-effector to the initial point on the reference trajectory. Additionally, the velocity-level controllers require the computation of Jacobian pseudo-inverse at each time-step, resulting in high computation cost. Position-level control, however, altogether avoid the mathematical manipulation of the Jacobian matrix, thereby significantly reducing the computation cost. Additionally, It is also worth noting that unlike the traditional obstacle avoidance algorithms, the proposed algorithm does not make an assumption about the shape of the obstacle, neither consider it as a point object [11]. The proposed algorithm directly use the 3D-model of the manipulator to calculate the distance of its link from the obstacle. As such, it works for any arbitrary manipulator and obstacle shape, which makes it realistic for an actual industrial setup. Although the algorithm requires 3D-geometry of the obstacle, with modern depth mapping sensors, this can be easily achieved. The main highlights of this paper are:

1) We propose an optimization framework for unifying the tracking control and obstacle avoidance problems by using the penalty term approach. It fulfills two objectives: i) Minimize the tracking error, ii) Maximize the manipulator-obstacle distance.
2) We formulate the tracking control problem on position-level as compared to velocity-level as done in most traditional works. The position-level control avoids

the manipulation and pseudo-inversion of the Jacobian matrix, consequently reducing the computation cost.
3) Using the GJK algorithm, to efficiently measure the distance between a manipulator and an arbitrarily-shaped obstacle by directly using their 3D-geometries, without making any assumption about their shapes.
4) We propose a metaheuristic based recurrent neural network, BAORNN, to efficiently solve the formulated optimization problem so that the manipulator can be controlled in real-time.
5) Extensive numerical analysis using a simulated model of KUKA LBR IIWA-14, a popular 7-DOF industrial robotic arm, are performed to demonstrate the performance of the proposed algorithm.

The remainder of this paper is organized as follows: Section II presents the problem formulation of the tracking control and obstacle avoidance. In Section III, firstly, the GJK algorithm is described briefly, and then the details of BAORNN algorithm are laid down. Theoretically, analysis is presented to prove the global convergence of the algorithm. Section IV outlines the simulation methodology, present the results, and discuss their implications. Section V concludes the paper.

II. Problem Formulation

In this section, we will mathematically formulate the tracking control and obstacle avoidance problem and unify them into one optimization framework.

A. Tracking Control

Consider the task of moving a payload using robotic manipulator along a specified trajectory, say a circular path. Tracking control deals with the calculation of the joint-space trajectory, which will move the end-effector in the specified circular path. For a given robotic manipulator, the position of its end-effector is a function of its joint-angles. For example, consider a m-DOF robotic manipulator operating in a n-dimensional task-space ($n = 3$ for position control). The forward kinematic mapping is a surjective function of the joint-space coordinates

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

where $\mathbf{x}(t) \in \mathbb{R}^n$ and $\boldsymbol{\theta}(t) \in \mathbb{R}^m$ are the task-space and joint-space coordinates respectively. Note that $m > n$ for a redundant manipulator. The forward kinematic mapping $f(.)$ is a nonlinear vector-valued function, which is trivial to formulate using the mechanical design and Denavit–Hartenberg (DH) parameters for a given manipulator. However, the task for a manipulator is usually specified in the cartesian task-space instead of the joint space. Therefore, we are more interested in the inverse mapping, i.e., mapping from the task-space to the joint-space. Using 1, an inverse kinematics model can be defined

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

where $f^{-1}(.)$ denotes the inverse kinematic mapping. Now consider a reference trajectory $\mathbf{x}_r(t)$ for end-effector position in cartesian task-space. To trace this trajectory, the corresponding angles $\boldsymbol{\theta}_r(t)$ in joint-space must satisfy the following equation,

$$\mathbf{x}_r(t) = f(\boldsymbol{\theta}_r(t)). \tag{3}$$

Our goal is to solve the above equation for the value of $\boldsymbol{\theta}_r(t)$. If a closed-form expression for $f^{-1}(.)$ exist, we can trivially solve this equation using $\boldsymbol{\theta}_r(t) = f^{-1}(\mathbf{x}_r(t))$. However, for a redundant manipulator, the forward kinematic mapping $f(.)$ is surjective-only and not one-to-one, i.e., there exist an infinite solutions $\boldsymbol{\theta}_r(t)$ in the joint-space, which are mapped to the same reference trajectory $\mathbf{x}_r(t)$.

To resolve the redundancy, i.e., calculate an optimal joint-space trajectory out of infinitely many possible trajectories; we model the tracking control as following optimization problem

$$\min_{\boldsymbol{\theta}(t)} \quad g_{tr}(\mathbf{x}_r(t), \boldsymbol{\theta}(t)), \tag{4}$$

where $g_{tr}(.)$ is the tracking objective function and defined as

$$g_{tr}(\mathbf{x}_r, \boldsymbol{\theta}) = ||\mathbf{x}_r - f(\boldsymbol{\theta})||_2. \tag{5}$$

where $\mathbf{x}_r$ is the current point on the reference trajectory and $\boldsymbol{\theta}$ are the current joint-angles.

Remark 1. In the formulation of tracking control objective function, only the kinematic model of the manipulator is considered. The tracking control algorithm based on kinematic models are intensively studied for the control of the manipulator as shown by recent works [17], [37]. Apart from the academic research, kinematic control is also widely used in commercial robotic systems such as ping-pong manipulator [38], ABB IRB 360 [39], Adept Quattro 650HS [40], DOBOT, and UR 10 manipulator.

B. Obstacle Avoidance

The solution to optimization problem (4) does not guarantee that the manipulator does not collide with an obstacle. Our obstacle avoidance strategy is based on the principle: maximizing the minimum distance the links of the manipulator and the obstacle. To incorporate this principle into our control framework, we formulate an additional objective function which penalizes the angles in joint-space which bring the robot close to the obstacle. The obstacle avoidance optimization problem is defined as

$$\min_{\boldsymbol{\theta}(t)} \quad g_{\mathcal{O}I}(\mathcal{O}, \boldsymbol{\theta}(t)), \tag{6}$$

where $g_{\mathcal{O}I}(.)$ is called the obstacle avoidance objective function; which is a function of $\mathcal{O} \in \mathbb{R}^{n_\mathcal{O} \times 3}$, the 3D-geometry of the obstacle, i.e., cartesian coordinates of all its vertices, and $\boldsymbol{\theta}$, the joint-angles of the manipulator. Here $n_\mathcal{O}$ is the number of vertices in the 3D model of the obstacle. High value of $n_\mathcal{O}$ results in a fine-grained 3D-mesh. The objective function $g_{\mathcal{O}I}(.)$ is defined as

$$g_{\mathcal{O}I}(\mathcal{O}, \boldsymbol{\theta}) = \frac{1}{[\min_{i \in \{1,2,\ldots,m\}}\{d_i(\mathcal{O}, \boldsymbol{\theta})\}]^\beta}, \tag{7}$$

where $\boldsymbol{\theta}$ are the current joint-angles, m is the total number of links in the manipulator and $d_i(\mathcal{O}, \boldsymbol{\theta})$ is the distance of

i^{th} link from the obstacle $\mathcal{O}$. The reciprocal relation ensures that decreasing the value of objective function increases the distance between links and obstacle. β is a hyper-parameter, and from simulation analysis, we found that $\beta = 1$ provides the best performance. The value of the distance is calculated using the GJK algorithm (refer to Section III-C for further details)

$$d_i(\mathcal{O}, \boldsymbol{\theta}) = \text{GJK}(\mathcal{O}, \mathcal{M}_i(\boldsymbol{\theta})), \tag{8}$$

here again $i \in \{1, 2, ..., m\}$. Since GJK algorithm requires 3D-geometry of the two objects, so we defined a function $\mathcal{M}_i(\boldsymbol{\theta}) \in \mathbb{R}^{n_i \times 3}$ which returns the vertices of i^{th} link. Similar to $n_{\mathcal{O}}$, n_i is the number of vertices in the 3D-geometry i^{th} link. It must be noted that the location of vertices change when manipulator moves, i.e., it is a function of joint-angles $\boldsymbol{\theta}$. The initial geometry, $\mathcal{M}_i(\mathbf{0})$, is a given information for a manipulator based on its mechanical model. The subequent values of $\mathcal{M}(\boldsymbol{\theta})$ are calculated using

$$\mathcal{M}_i(\boldsymbol{\theta}) = \mathcal{R}_i(\boldsymbol{\theta})\mathcal{M}_i(\mathbf{0}) + \mathcal{T}_i(\boldsymbol{\theta})$$

where $\mathcal{R}_i(\boldsymbol{\theta})$ and $\mathcal{T}_i(\boldsymbol{\theta})$ are the rotation and translation matrix for the i^{th} link. These matrices can be calculates using the forward kinematic model of manipulator.

C. Joint-Angle Limits

A solution to optimization problem of (4) or (6) does not guarantee that the solution will lie within the mechanical limit of the joint. To guarantee the solution does not violate the joint-angle limits, the following constraint must be satisfied

$$\boldsymbol{\theta}^- < \boldsymbol{\theta}(t) < \boldsymbol{\theta}^+, \tag{9}$$

where $\boldsymbol{\theta}^-$ and $\boldsymbol{\theta}^+$ denote the lower and upper limits on the joints-angles respectively, the value of these limits depend on the mechanical construction of the manipulator and the type of actuator used to move the joints.

D. Unified Tracking Control and Obstacle Avoidance

Above, we formulated three seperate component of the problem: tracking control (4), obstacle avoidance (6) and joint-angle limits (9). These can be unified into the following optimization problem

$$\begin{aligned} \min_{\boldsymbol{\theta}(t)} \quad & g(\mathcal{O}, \mathbf{x}_r(t), \boldsymbol{\theta}(t)) \\ \text{s.t.} \quad & \boldsymbol{\theta}^- < \boldsymbol{\theta}(t) < \boldsymbol{\theta}^+, \end{aligned} \tag{10}$$

where $g(.)$ is the unified objective function defined as

$$g(\mathcal{O}, \mathbf{x}_r, \boldsymbol{\theta}) = g_{tr}(\mathbf{x}_r, \boldsymbol{\theta}) + \Lambda g_{\mathcal{O}I}(\mathcal{O}, \boldsymbol{\theta}), \tag{11}$$

where Λ is a constant parameter which controls the trade-off between tracking performance and maximizing the manipulator-obstacle distance. A value of $\Lambda = 0$ turns off the obstacle avoidance completely. The value of Λ greatly affect obstacle avoidance performance. Its effect is discussed in detail in Section IV.

Remark 2. The obstacle avoidance objective function $g_{\mathcal{O}I}(\mathcal{O}, \boldsymbol{\theta})$ acts as a penalty term in the unified objective function above. When the manipulator is moved far from the obstacle, the value of the penalty term becomes small, and the algorithm rewards the optimizer by reducing the overall value of the objective function.

Although the penalty term approach actively reward the optimizer to avoid the obstacle, but consider a circumstance where the position of obstacle makes it impossible to track the reference trajectory; to avoid the collision in such a condition, we add inequality constraint to (10),

$$\begin{aligned} \min_{\boldsymbol{\theta}(t)} \quad & g(\mathcal{O}, \mathbf{x}_r(t), \boldsymbol{\theta}(t)) \\ \text{s.t.} \quad & \boldsymbol{\theta}^- < \boldsymbol{\theta}(t) < \boldsymbol{\theta}^+ \\ & d_i(\mathcal{O}, \boldsymbol{\theta}(t)) > d_{min} \text{ for } i \in \{1, 2, ..., m\}. \end{aligned} \tag{12}$$

The second constraint puts a hard lower-bound, d_{min}, on obstacle-manipulator distance.

Based on the above formulation, the complete form of the optimization problem can be written as

$$\begin{aligned} \min_{\boldsymbol{\theta}(t)} \quad & ||\mathbf{x}_r(t) - f(\boldsymbol{\theta}(t))||_2 \; + \Lambda \frac{1}{[\min_{i \in \{1,2,...,m\}}\{d_i(\mathcal{O}, \boldsymbol{\theta}(t))\}]^{\beta}} \\ \text{s.t.} \quad & \boldsymbol{\theta}^- < \boldsymbol{\theta}(t) < \boldsymbol{\theta}^+ \\ & d_i(\mathcal{O}, \boldsymbol{\theta}(t)) > d_{min} \quad \text{for } i \in \{1, 2, ..., m\}. \end{aligned} \tag{13}$$

The solution to this optimization problem gives the joint-space trajectory $\boldsymbol{\theta}_r(t)$. Now we will formulate the BAORNN algorithm in Section III to solve this optimization problem in real-time.

III. Control System Design

In this section, we will first formulate the BAORNN algorithm. Then we will briefly describe the GJK algorithm used for calculating the distance between the manipulator and the obstacle.

A. BAORNN Algorithm

After the problem formulation in Section II, the tracking control and obstacle avoidance finally boils down to solving the optimization problem (13) in real-time while the manipulator is operating. BAORNN algorithm mimics the behavior of a beetle; which uses its pair of antennae and olfactory sense to probe an unknown environment in search of food (i.e., search for the region with maximum smell). At each step, beetle measure magnitude of smell at both antennae before deciding the direction of its next step. Especially, note the intermediary action; i.e., instead of randomly moving in any directions, the beetle stops after each step, uses just the olfactory sense to develop better intuition about goal direction and only then makes a calculated decision to take the next step. This overall behavior especially the intermediary action inspired us to incorporate the concept of "virtual manipulators" (analogous to antennae's olfactory sense) into our control framework and develop a heuristic mechanism to control the manipulator.

Suppose, at time-step k, the manipulator starts at $\boldsymbol{\theta}_0$ in joint-space. The algorithm generates a normalized normally distributed random direction vector $\vec{\mathbf{b}} \in \mathbb{R}^m$ analogous to the

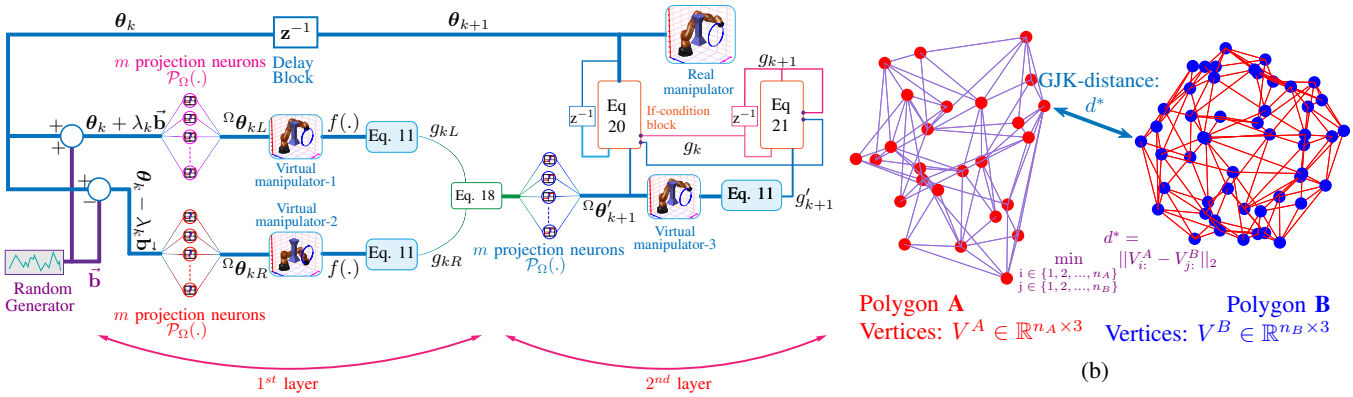


Fig. 1: (a) The topology of the RNN for the BAORNN algorithm. The diagram illustrates the working of the algorithm formulated in Section III-A. (b) Illustration of GJK algorithm.

Algorithm 1: BAORNN algorithm - Tracking control & Obstacle avoidance

Input: kinematic model $f(.)$ and 3D-geometry matrix $\mathcal{M}_i(\mathbf{0})$ $(i \in \{1, 2, ..m\})$ of the manipulator, 3D-geometery of the obstacle $\mathcal{O}$, reference trajectory $\mathbf{x}_r(t) \in \mathbb{R}^n$, an objective function $g(\mathcal{O}, \mathbf{x}, \boldsymbol{\theta})$. Additionally, the values of hyper-parameters: β, Λ, c_1 and c_2.

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

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

$k \leftarrow 0$ $k_{stop} \leftarrow$ maximum number of time-steps allowed

while $k < k_{stop}$ **do**

- Generate a normalized random direction vectors, $\vec{\mathbf{b}} \in \mathbb{R}^m$ in the joint-space.
- Use the generated random vector to calculate the location of left and right antennae, $\boldsymbol{\theta}_{kL}$ and $\boldsymbol{\theta}_{kR}$ respectively, using (15).
- Project the location of these antennae on the constrained set Ω using the projection function $\mathcal{P}_\Omega$ as defined in (16) to . Calculate the value of objective function at both location using "Virtual manipulators" as defined in (17).
- Calculate he updated location in joint-space using (18).
- Check if the updated location improves the value of objective function using (20).
- Move the manipulator to $\boldsymbol{\theta}_{k+1}$ and update the value of g_{k+1}.
- $k \leftarrow k + 1$

end

antenna of the beetle. Using direction vector $\vec{\mathbf{b}}$, the location of end-point of the antennae can be calculated as

$$\boldsymbol{\theta}_{kL} = \boldsymbol{\theta}_k + \lambda_k \vec{\mathbf{b}}, \qquad \boldsymbol{\theta}_{kR} = \boldsymbol{\theta}_k - \lambda_k \vec{\mathbf{b}}, \tag{14}$$

where λ_k is a hyper-parameter representing the length of the antenna, $\boldsymbol{\theta}_{kL}$ and $\boldsymbol{\theta}_{kR}$ denotes the location of left and right antennae respectively at time-step k. However, these vectors might not satisfy the constraint of the problem (13). Therefore we project these vectors onto the constrained set

$${}^\Omega\boldsymbol{\theta}_{kX} = \mathcal{P}_\Omega(\boldsymbol{\theta}_{kX}), \tag{15}$$

where $X \in \{L, R\}$, $\mathcal{P}_\Omega(.)$ is the projection function which restrict the input inside the constrained set Ω. The set Ω is mathematically defined as,

$$\Omega = \{\boldsymbol{\theta} \in \mathbb{R}^m | \boldsymbol{\theta}^- < \boldsymbol{\theta} < \boldsymbol{\theta}^+ \wedge d_i(\mathcal{O}, \boldsymbol{\theta}) > d_{min}\}.$$

There are several way to project a vector $\boldsymbol{\theta}$ on a set Ω, here we define a computationally straightforward projection function

$$\mathcal{P}_\Omega(\boldsymbol{\theta}_{kX}) = \begin{cases} \max\{\boldsymbol{\theta}^-, \min\{\boldsymbol{\theta}_{kX}, \boldsymbol{\theta}^+\}\} & \text{if } \ d_i > d_{min} \\ \boldsymbol{\theta}_k & \text{if } \ d_i < d_{min}, \end{cases} \tag{16}$$

where again $X \in \{L, R\}$, d_i is same as defined in (8). The projected antennae locations ${}^\Omega\boldsymbol{\theta}_{kL}$ and ${}^\Omega\boldsymbol{\theta}_{kR}$, is then used to evaluate the value of objective function

$$g_{kX} = g(\mathcal{O}, \mathbf{x}_r(t), {}^\Omega\boldsymbol{\theta}_{kX}), \tag{17}$$

where g_{kX} $(X \in \{L, R\})$ is the value of objective function at antenna locations.

We then use the calculated values of the objective function at antennae locations, g_{kX}, to move in a direction, inside joint-space, where the value of the objective function is decreasing by using the following update rule

$${}^\Omega\boldsymbol{\theta}'_{k+1} = \mathcal{P}_\Omega(\boldsymbol{\theta}_k - \delta_k(\lambda_k)\text{sign}(g_{kL} - g_{kR})\vec{\mathbf{b}}), \tag{18}$$

where ${}^\Omega\boldsymbol{\theta}'_{k+1}$ is the updated location in joint-space projected on set Ω, $\text{sign}(g_L - g_R)\vec{\mathbf{b}}$ term ensure that the next step is taken in direction of the antennae with small objective function value. $\delta_k(\lambda_k)$, is a hyper-parameter and denotes the step-size, i.e., euclidean distance between ${}^\Omega\boldsymbol{\theta}'_{k+1}$ and $\boldsymbol{\theta}_k$ locations. The step-size is a function of antennae length λ_k; there relationship will also be discussed later. After calculating ${}^\Omega\boldsymbol{\theta}'_{k+1}$, the value of objective function is re-evaluated

$$g'_{k+1} = g(\mathcal{O}, \mathbf{x}_r(t), {}^\Omega\boldsymbol{\theta}'_{k+1}), \tag{19}$$

the value of g'_{k+1} is compared to the value of objective function at previous time-step g_k. If there is any improvement (i.e., the updated value is smaller), then the robot moves to ${}^{\Omega}\boldsymbol{\theta}'_{k+1}$ in joint-space; otherwise, it remains at the current location

$$\boldsymbol{\theta}_{k+1} = \begin{cases} {}^{\Omega}\boldsymbol{\theta}'_{k+1} & \text{if } \ g'_{k+1} < g_k \\ \boldsymbol{\theta}_k & \text{if } \ g'_{k+1} \geq g_k. \end{cases} \tag{20}$$

Similarly, the value of g_{k+1} is assigned to use in next iteration

$$g_{k+1} = \begin{cases} g'_{k+1} & \text{if } \ g'_{k+1} < g_k \\ g_k & \text{if } \ g'_{k+1} \geq g_k. \end{cases} \tag{21}$$

After moving to $\boldsymbol{\theta}_{k+1}$, the iterative procedure is repeated for the next time-steps. The steps of the proposed BAORNN algorithm are systematically presented in Algorithm 1.

The choice of hyper-parameter λ_k and $\delta_k(\lambda_k)$, where k denotes time-step, affects the speed of convergence. By empirical analysis, we found that the following rules for the selection of hyper-parameters provide a reasonable convergence rate

$$\lambda_k = c_1\sqrt{g'_k} \tag{22}$$

$$\delta_k(\lambda_k) = c_2\lambda_k \tag{23}$$

where c_1 and c_2 are constant design parameters. The above relation regulate step-size such that the algorithm takes large steps when the end-effector is far from the goal position and make the steps extremely small when reached near the goal. The small step-size is necessary to prevent the overshooting of end-effector near goal position. For c_1 and c_2, we propose the following rules for fast convergence.

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

where T_s is the sampling time of the control loop.

The proposed BAORNN is formulated as RNN, as shown in Fig. 1a. The formulated RNN has a two-layered topology with a temporal-feedback connection from the second layer to the first layer. RNN architecture has a total of $3m + 6$ neurons. The block "Random" represents a random vector generator and provide normally distributed unit direction vector $\vec{\mathbf{b}}$ for the BAORNN algorithm. The neurons, shown as circles, use projection $\mathcal{P}_{\Omega}(.)$ as their activation function. The neurons shown as curved rectangular boxes represent "virtual manipulators", and their activation function is given by $f(.)$. Similarly, the neurons represented by curved boxes (in cyan) represent the objective function evaluation, and their activation function is given by $g(.)$.

By parsing the RNN architecture shown in Fig. 1a, it can be shown that the algorithm has a complexity of $O(m)$, 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 efficiently on embedded processors since modern embedded processors have dedicated hardware unit for floating-point calculations.

B. Theoritical Analysis

Theorem 1. *For the tracking control and obstacle avoidance of a redundant manipulator, starting from an initial joint-space angles $\boldsymbol{\theta}_0$; the joint-space trajectory $\boldsymbol{\theta}_r(t)$ generated by BAORNN algorithm is stable, i.e.,*

$$g_{k+1} \leq g_k, \quad \forall \quad k \geq 0, \tag{24}$$

the values of objective function are monotonically decreasing.

Proof. See Lemma 1 of [34]. □

Theorem 2. *For the tracking control and obstacle avoidance of a redundant manipulator, starting from an initial joint-space angles $\boldsymbol{\theta}_0$; the end-effector trajectory $f(\boldsymbol{\theta}_r(t))$ is convergent to the reference trajectory $\mathbf{x}_r(t)$, i.e.,*

$$f(\boldsymbol{\theta}(t)) \to \mathbf{x}_r(t), \quad as \quad t \to \infty. \tag{25}$$

Proof. See Theorem 1 of [34]. □

C. GJK-Distance Algorithm

GJK algorithm is an efficient algorithm to calculate the minimum distance between two arbitrarily shaped convex 3D-polygons. Although, in our case, the 3D-geometry of a manipulator link or the obstacle might be non-convex shape, however, the collision avoidance between convex-hulls of both objects is a sufficient condition for real collision avoidance.

Consider two convex polygons A and B in 3D-space, their vertices defined by matrices $V^A \in \mathbb{R}^{n_A \times 3}$ and $V^B \in \mathbb{R}^{n_B \times 3}$ respectively. n_A and n_B are the numbers of vertices of polygon A and B respectively. Each row of these matrices represents the location of a vertex of the corresponding polygon. The GJK algorithm takes these matrices and calculates the minimum distance between the closest vertices of the two polygons,

$$\text{GJK}(V^A, V^B) = \min_{\substack{i \in \{1,2,...,n_A\} \\ j \in \{1,2,...,n_B\}}} ||V^A_{i:} - V^B_{j:}||_2$$

where the notation $V_{i:}$ is used to represent the i^{th} row of a matrix V. Fig. 1b illustrates GJK-algorithm.

D. Computational Complexity

Here we will estimate the computational complexity of the BAORNN algorithm formulated in Section III-A. The first step in the algorithm is the generation of is a random vector $\vec{b}$ with m elements; the operation requires m floating-point operations. Next, we calculate θ_{kL} and θ_{kR}, each requiring m multiplication and m additions, totalling $4m$ floating-point operations. Next step requires the projection of two vectors using the projection function $f_{\Omega}(.)$, which require a total of $4m$ comparisons. Then we use (17) to calculate the value of objective function at both antennae location. The evaluation of objective function is the most computationally intensive step of the algorithm since it requires the calculation of Euclidean distance as well as GJK-distance, as given in (11). The calculation of Euclidean distance require a total of $3m - 1$ floating-point operations (m subtractions, m squaring operations and $m - 1$ additions). The calculation of GJK-distance depends on the number of vertices in the

3D models of two objects and require a total of $n_A + n_B$ operations, as shown by [41]. Where n_A and n_B are the numbers of vertices in the 3D model of both objects, respectively. For the case of manipulator and obstacle distance, using the notation of Section II-B, the total number of operation comes out to be $n_{\mathcal{O}} \sum_{i=1}^{m} n_i$. Although this number is large, these operations are only required in the first iterations of the algorithm, the later iterations of GJK-algorithm are near-constant time, as pointed out by [41], [42]. Therefore, the total number of operations required by GJK-algorithm are effectively m. It means that a total of $4m + 2$ operations are required for evaluating the objective function; some additional operations are required for the scalar addition and multiplication as given in (11). Since objective function is evaluated twice in (17), therefore this step require a total of $8m + 4$ operations. The next step, as given in (18), requires a total of $2m + 1$ floating-point operations. Similarly, the subsequent step is again objective function evaluation requiring $4m + 2$ operations. The final step of the algorithm, as given in (20) and (21), require a total of $2m$ comparisons. Adding the floating-point operations required for each step as calculated above; the final count comes out to be $(m + 4m + 4m + (8m + 4) + (2m + 1) + (4m + 2) + 2m) = 25m + 7$.

The above analysis shows that the BAORNN algorithm have a complexity of $O(m)$, where m are the total number of links of the manipulator. This show that the complexity of the BAORNN algorithm is only linearly related to the number of links of manipulator. For $m = 7$, as in the case of IIWA14 manipulator, the required number of operations per iterations are of the order of 182. Modern embedded processors can efficiently perform floating-point operations of this order withing few hundred of microseconds.

IV. Simulation Results & Discussion

In this section, simulation methodology, for evaluating the performance of the proposed algorithm, is presented along with the obtained results and discussion. Simulated model of KUKA LBR IIWA-14 manipulator is used as a testbench. The IIWA-14 has 7-DOF. 3D-model of the manipulator is shown in Fig. 2.

A. Simulation Methodology

We used the model of IIWA-14 provided by MATLAB Robotic System Toolbox [43]. The model provides an excellent representation of a real-world industrial manipulator and therefore act as a desirable simulation testbed. To test the obstacle avoidance performance, we placed an arbitrarily shaped obstacle in front of the manipulator. The simulation setup, including manipulator and obstacle, is shown in Fig. 2.

We used two reference trajectories [19] in our simulations: a rectangular and a circular trajectory as shown in Fig. 3. The four corners of the rectangular paths used in simulation are: $[0.2\ \ 0.6\ \ 0.8]^T$, $[-0.1\ \ 0.6\ \ 0.8]$, $[-0.1\ \ 0.6\ \ 0.2]$, and $[0.2\ \ 0.6\ \ 0.2]$. The total time for tracking the rectangular trajectory is 50 seconds. For generating the circulat trajectory we used following parameteric equation

$$\mathbf{x}_r^{\text{circle}}(t) = \vec{\mathbf{C}} + r\cos(2\pi t/T)\vec{\mathbf{A}} + r\sin(2\pi t/T)\vec{\mathbf{B}}. \quad (26)$$

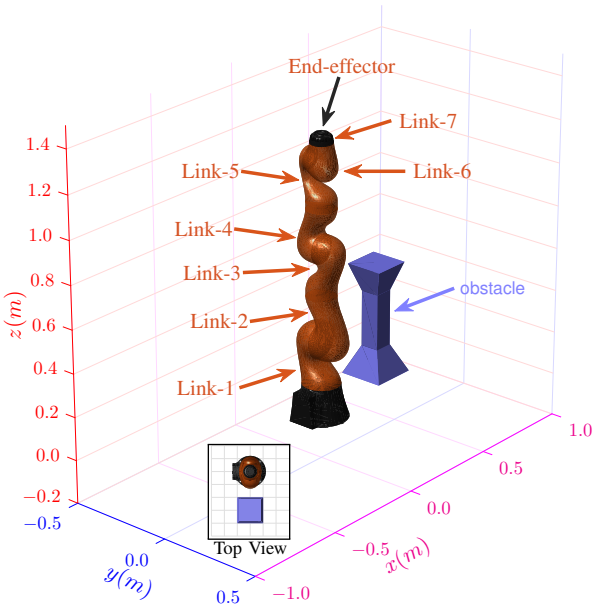


Fig. 2: 3D model of KUKA LBR IIWA-14 7-DOF manipulator with the obstacle used in simulations. The obstacle is placed in front of the operational region of the robot.

where $\vec{\mathbf{C}}$ is a position vector to center of the circle, $\vec{\mathbf{A}}$ and $\vec{\mathbf{B}}$ are two perpendicular unit vectors defining the plane of the circle in 3D space, r is the radius of the circle. T denotes the total time duration. Following values were used in simulation: $\vec{\mathbf{C}} = [0.0\ \ 0.6\ \ 0.5]$, $\vec{\mathbf{A}} = [1\ \ 0\ \ 0]$, and $\vec{\mathbf{B}} = [0\ \ 0\ \ 1]$. These values represent a circlular path in $x - z$ plane at $y = 0.6$. The two trajectories mentioned above were chosen for developing simulation results in this paper. Without the loss of generality, the proposed algorithm works for an arbitrarily shaped reference-trajectory, provided that all the points on the trajectory are reachable by the end-effector without violating the mechanical limits of the manipulator's joints.

To systematically study the effect of the proposed algorithm, we first conducted a simulation without any obstacle-avoidance, i.e., setting $\Lambda = 0$ in (13) and ignoring the 2^{nd} constraint. Then we performed simulations with different values of Λ and its effect on the obstacle avoidance performance is discussed in details.

B. Trajectory tracking results

The first set of the simulation consists of analyzing the response of the manipulator without obstacle avoidance as described in Section IV-A. The results for rectangular and circular trajectories are both shown in Fig. 3. It can be seen that several angles in joint-space resulted in a collision with the obstacle. It is because the algorithm calculated a joint-space trajectory which minimized the tracking error without considering the obstacle in its path.

Next we simulated the response of the system with different values of Λ as defined in (11). Fig. 4 shows the result for rectangular reference trajectory. The initial configuration of the manipulator's joint is assumed to be home configuration, i.e., all joint-angles are zero at the beginning. Fig. 4a to Fig. 4e summarizes the manipulator's response for $\Lambda = 0.002$.

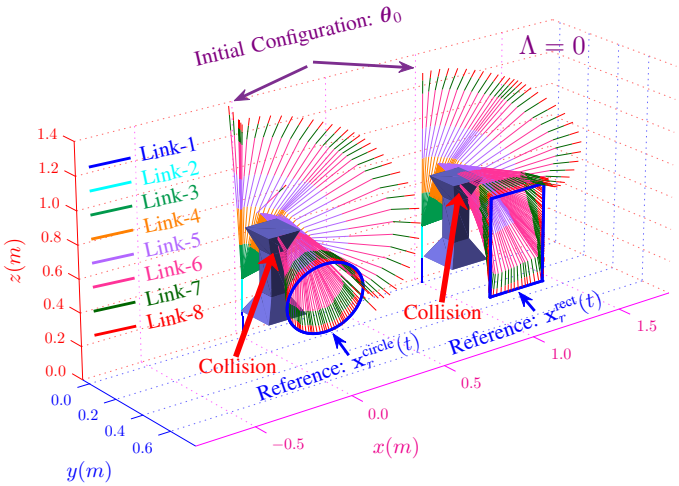


Fig. 3: Performance of tracking controller after switching-off the obstacle avoidance term, i.e., $\Lambda = 0$ as defined in (11). The links collide with obstacle for both trajectories.

Fig. 4a shows the motion of each links of the manipulator along with the reference trajectory (shown in blue). The initial portion of the manipulator's trajectory lies away from the reference trajectory because the manipulator starts from home configuration and algorithm takes some time to find an optimal joint-space trajectory which takes the end-effector near the reference trajectory while avoiding the obstacle. Once the end-effector reaches near the reference trajectory, it accurately follows it for the rest of the path. Top view of the manipulator's trajectory is also shown as inset graphic, which confirms that the manipulator does not collide with an obstacle at any point. Fig. 4b shows the cartesian coordinates of end-effector motion and Fig. 4c shows the joint-space coordinates of the manipulator trajectory. It is worth pointing out that the unsmooth response shown by these trajectories is typical of metaheuristic algorithms because of the stochastic nature; however, the resultant gain in computational efficiency is much greater. Fig. 4d shows the position tracking error which is defined as $\mathbf{e}(t) = \mathbf{x}_r(t) - f(\boldsymbol{\theta}_r(t))$. At $t = 0$, it shows a huge tracking error of $\approx [0.5 \;\; -0.5 \;\; 0.7]^T$; again, this is the result of starting from the home configuration, which requires some time for reaching the reference trajectory. However, after some time, he tracking error converges as the end-effector finally converges to the reference trajectory. It also proves that the global convergence performance of the proposed algorithm, i.e., the tracking error converges to zero and does not rise again, except for some small ripples caused by the stochastic nature of the algorithm. Similarly, Fig. 4e shows the minimum distance of any link of the manipulator from the obstacle as defined in (7). A high value is desirable because it reduces the risk of collision in case of uncertainty in obstacle position or error in the manipulator model. We set $d_{min} = 0.02$, which acts as a lower limit for the minimum manipulator-obstacle distance. We repeated the same set of simulations with $\Lambda = 0.0002$. Fig. 4f to Fig. 4j summarizes the manipulator's response. The major difference between these two situations is the quality of the obstacle avoidance performance. Fig. 4j shows that the minimum manipulator-obstacle distance is smaller as compared to Fig. 4e, i.e., the links of manipulator were closer to the obstacle as compared to the latter case, increases the risk of collision. The same conclusion can be drawn from the inset graphics of Fig. 4f and Fig. 4f which shows that the links are much closer to obstacle in second case as compared to the first case. We had to reduce the value of d_{min} to 0.002 to successfully simulate a complete rectangular trajectory without colliding with an obstacle.

The simulation results for the circular reference trajectory are shown in Fig. 5. These results show a similar trend. For a small value of Λ, the manipulator-obstacle distance decrease, and we had to reduce d_{min} to complete the simulation. However, for a higher value of Λ, the algorithm shows an excellent performance in avoiding the obstacle. It should, however, be noted that increasing the value too much will significantly decrease the tracking performance because the algorithm will aggressively try to avoid the obstacle.

V. Conclusion

In this paper, we proposed a framework to simultaneously address the problem of tracking control and obstacle avoidance in real-time. The proposed framework unifies the two goals into a single constrained optimization problem. The penalty term approach significantly improves the performance of the proposed algorithm by actively rewarding the optimizer for avoiding the obstacle. This approach results in a joint-space trajectory which maximizes the distance manipulator-obstacle distance. To solve the formulated optimization problem, in real-time, we proposed an RNN based on a metaheuristic optimization algorithm, called Beetle Antennae Olfactory. A key feature of the proposed framework is that it does not make assumptions about a specific shape of the obstacle. It directly uses the 3D-geometries of the manipulator and obstacle for formulating the penalty term using GJK-algorithm. Potential application of such an approach includes the operation of a manipulator in a dynamic environment where the shape of the obstacle is time-varying. The application of the GJK-algorithm to measure manipulator-obstacle distance allows the controller to work for an arbitrarily-shaped obstacle. Similarly, the proposed algorithm is also particularly useful for surgical-robots where it is critical to maintaining a safe distance, between the links of the manipulator and the patient, to ensure safety. The theoretical treatment is also presented in the paper to prove the stability and convergence of the proposed algorithm. Simulations using a KUKA LBR 7-DOF industrial manipulator are presented to prove the performance of the proposed algorithm.

VI. Future Work

Potential further directions to improve the capability and performance of the proposed algorithm includes; extending the formulation of the optimization problem to incorporate multiple obstacles while keeping the calculation of manipulator-manipulator distance computationally efficient. Another exciting application of the proposed algorithm is to enhance the safety of surgical-robots by actively

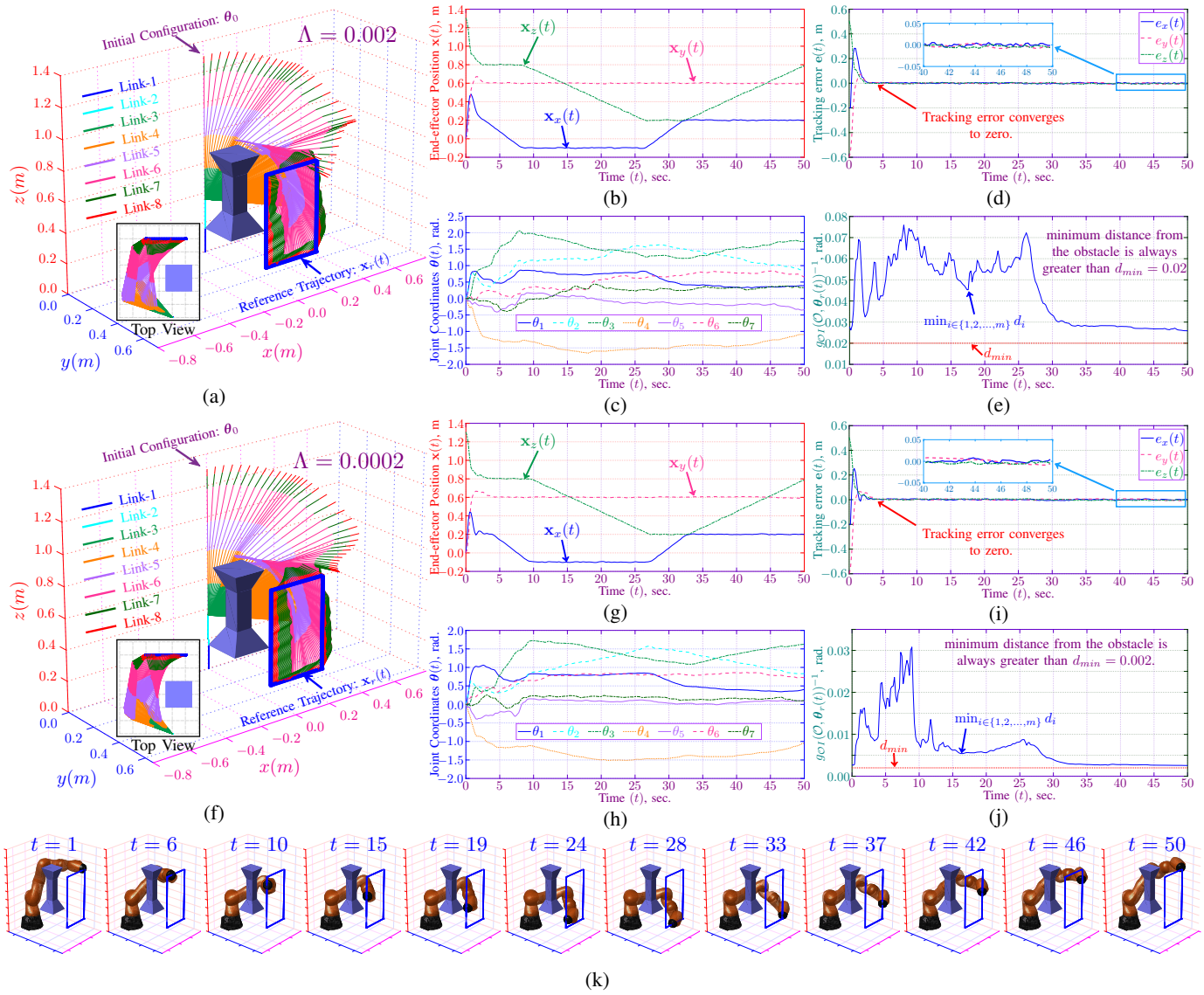


Fig. 4: Simulation results for rectangular trajectory tracking for different values of Λ as defined in (11). (a-e) shows result for $\Lambda = 0.002$. (a) The trajectory of each manipulator link along with reference trajectory. (b) Profile of task-space trajectory of the end-effector. (c) Profile of joint-space trajectory of the manipulator. (d) Profile of the position tracking error. (e) Minimum GJK-distance of the manipulator from obstacle as defined in (7). (f-j) Shows similar results for $\Lambda = 0.0002$. It must be noted that the minimum manipulator-obstacle distance for $\Lambda = 0.002$ is much better (i.e., larger) as compared to $\Lambda = 0.0002$. (k) Simulation model of the LBR IIWA-14 robot while tracking the reference trajectory for $\Lambda = 0.002$.

ensuring that the link of manipulator does not touch the patient. An advanced version of the proposed algorithm will incorporate multiple mobile-manipulators operating in a dynamic environment with several obstacles of different shapes. Such an algorithm will require decentralized control algorithm and collaboration between manipulators to ensure efficient operation.

REFERENCES

[1] C. Yang, C. Zeng, Y. Cong, N. Wang, and M. Wang, "A learning framework of adaptive manipulative skills from human to robot," *IEEE Trans. on Ind. Informatics*, vol. 15, no. 2, pp. 1153–1161, 2018.

[2] 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.

[3] 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 Trans. on Ind. Informatics*, vol. 13, no. 3, pp. 1162–1171, 2016.

[4] 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 Trans. on Cybern.*, vol. 45, no. 8, pp. 1390–1400, 2014.

[5] C. Yang, G. Peng, L. Cheng, J. Na, and Z. Li, "Force sensorless admittance control for teleoperation of uncertain robot manipulator using neural networks," *IEEE Trans. on Syst., Man, and Cybern.: Syst.*, 2019.

[6] 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.

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

[8] A. M. Zanchettin, L. Bascetta, and P. Rocco, "Achieving humanlike motion: Resolving redundancy for anthropomorphic ind. manipulators," *IEEE Robot. & Autom. Mag.*, vol. 20, no. 4, pp. 131–138, 2013.

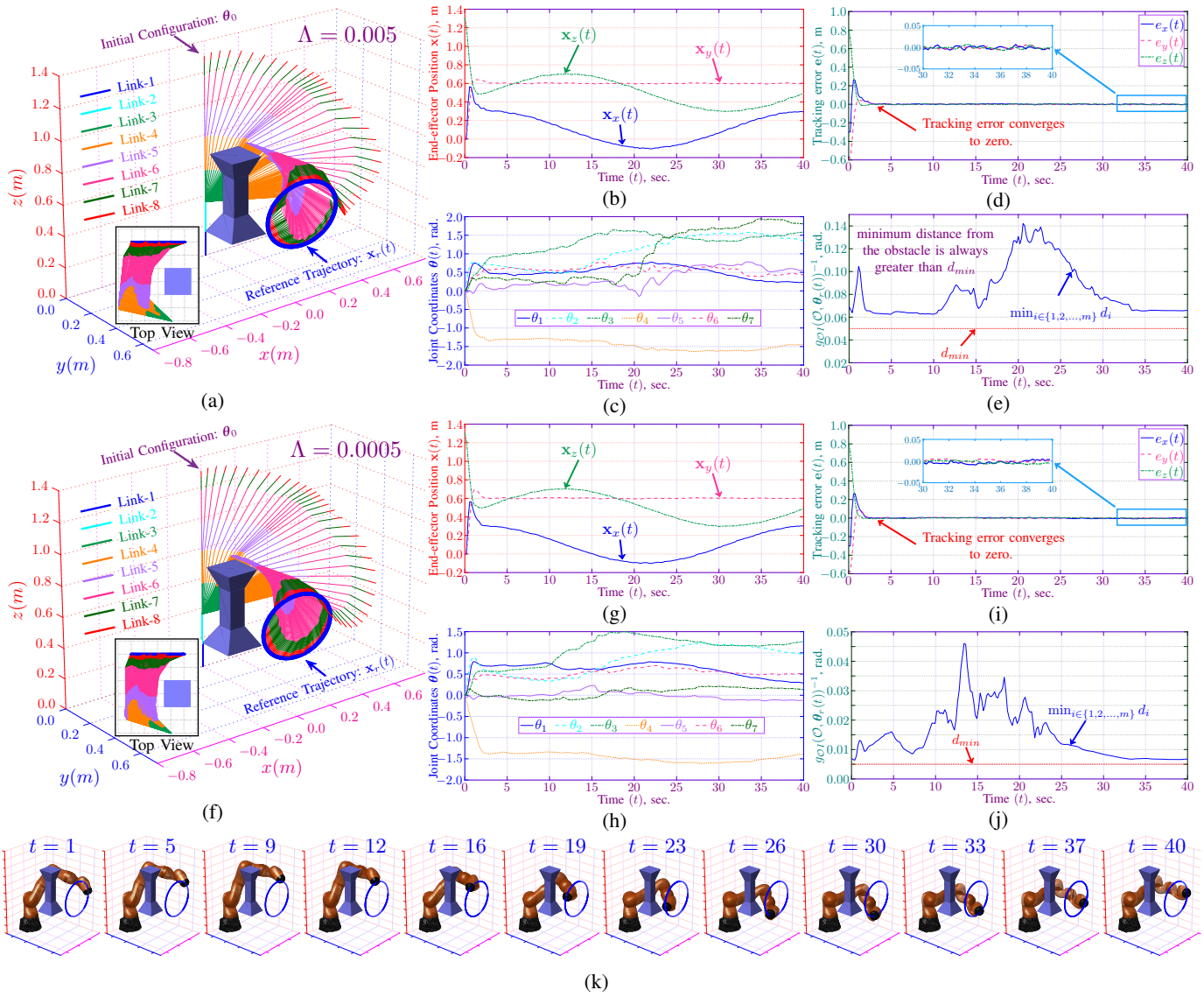


Fig. 5: Simulation results for circular trajectory tracking for different values of Λ as defined in (11). (a-e) shows result for $\Lambda = 0.005$. (a) The trajectory of each manipulator link along with reference trajectory. (b) Profile of task-space trajectory of the end-effector. (c) Profile of joint-space trajectory of the manipulator. (d) Profile of the position tracking error. (e) Minimum GJK-distance of the manipulator from obstacle as defined in (7). (f-j) Shows similar results for $\Lambda = 0.0005$. It must be noted that the minimum manipulator-obstacle distance for $\Lambda = 0.005$ is much better (i.e., larger) as compared to $\Lambda = 0.0005$. (k) Simulation model of the LBR IIWA-14 robot while tracking the reference trajectory for $\Lambda = 0.005$.

[9] 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.

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

[11] Z. Zhang, S. Chen, X. Zhu, and Z. Yan, "Two hybrid end-effector posture-maintaining and obstacle-limits avoidance schemes for redundant robot manipulators," *IEEE Trans. on Ind. Informatics*, 2019.

[12] Y.-J. Chen, M.-Y. Ju, and K.-S. Hwang, "A virtual torque-based approach to kinematic control of redundant manipulators," *IEEE Trans. on Ind. Electron.*, vol. 64, no. 2, pp. 1728–1736, 2016.

[13] C.-S. Tsai, *Online Trajectory Generation for Robot Manipulators in Dynamic Environment–An Optimization-based Approach*. PhD thesis, UC Berkeley, 2014.

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

[15] J. Józwik, D. Ostrowski, P. Jarosz, and D. Mika, "Industrial robot repeatability testing with high speed camera phantom v2511," vol. 10, no. 32, 2016.

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

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

[18] V. Lippiello, J. Cacace, A. Santamaria-Navarro, J. Andrade-Cetto, M. A. Trujillo, Y. R. Esteves, and A. Viguria, "Hybrid visual servoing with hierarchical task composition for aerial manipulation," *IEEE Robot. and Autom. Letters*, vol. 1, no. 1, pp. 259–266, 2015.

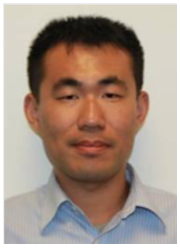
[19] S. Li, S. Chen, B. Liu, Y. Li, and Y. Liang, "Decentralized kinematic control of a class of collaborative redundant manipulators via recurrent neural networks," *Neurocomputing*, vol. 91, pp. 1–10, 2012.

[20] 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.

[21] 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.
[22] W. He, Z. Yan, Y. Sun, Y. Ou, and C. Sun, "Neural-learning-based control for a constrained robotic manipulator with flexible joints," *IEEE Trans. on neural networks and learning Syst.*, vol. 29, no. 12, pp. 5993–6003, 2018.
[23] 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.
[24] W. He, Z. Yin, and C. Sun, "Adaptive neural network control of a marine vessel with constraints using the asymmetric barrier lyapunov function," *IEEE Trans. on Cybern.*, vol. 47, no. 7, pp. 1641–1651, 2016.
[25] 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," vol. 27, no. 3, pp. 574–588, 2018.
[26] J. Na, B. Jing, Y. Huang, G. Gao, and C. Zhang, "Unknown system dynamics estimator for motion control of nonlinear robotic systems," 2019.
[27] H. Wang, Y. Zou, P. X. Liu, and X. Liu, "Robust fuzzy adaptive funnel control of nonlinear systems with dynamic uncertainties," vol. 314, pp. 299–309, 2018.
[28] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*, pp. 396–404, Springer, 1986.
[29] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," in *2012 IEEE Intl. Conf. on Robot. and Autom.*, pp. 338–345, IEEE, 2012.
[30] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in 3d space," *IEEE Journ. on Robot. and Autom.*, vol. 4, no. 2, pp. 193–203, 1988.
[31] X.-S. Yang, *Nature-inspired metaheuristic algorithms*. Luniver, 2010.
[32] X.-S. Yang, *Engineering optimization: an introduction with metaheuristic applications*. John Wiley & Sons, 2010.
[33] X. Jiang and S. Li, "Bas: beetle antennae search algorithm for optimization problems," *arXiv preprint 1710.10724*, 2017.
[34] Y. Zhang, S. Li, and B. Xu, "Convergence analysis of beetle antennae search algorithm and its applications," *arXiv preprint 1904.02397*, 2019.
[35] 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 Conf. on Ind. Electron. and Applications (ICIEA)*, pp. 1599–1603, IEEE, 2018.
[36] X. Yin and Y. Ma, "Aggregation service function chain mapping plan based on beetle antennae search algorithm," in *Proceedings of the 2nd Intl. Conf. on Telecommunications and Communication Engineering*, pp. 225–230, ACM, 2018.
[37] Y. Zhang, S. Li, J. Zou, and A. H. Khan, "A passivity-based approach for kinematic control of redundant manipulators with constraints," *IEEE Trans. on Ind. Informatics*, 2019.
[38] S. Huang, Y. Peng, W. Wei, and J. Xiang, "Clamping weighted least-norm method for the manipulator kinematic control with constraints," *Intl. Journ. of Control*, vol. 89, no. 11, pp. 2240–2249, 2016.
[39] 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.
[40] 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.
[41] M. Montanari, N. Petrinic, and E. Barbieri, "Improving the gjk algorithm for faster and more reliable distance queries between convex objects," *ACM Trans. on Graphics (TOG)*, vol. 36, no. 3, p. 30, 2017.
[42] C. J. Ong and E. G. Gilbert, "The gilbert-johnson-keerthi distance algorithm: A fast version for incremental motions," in *Proceedings of Intl. Conf. on Robot. and Autom.*, vol. 2, pp. 1183–1189, IEEE, 1997.
[43] P. I. Corke *et al.*, "A robotics toolbox for matlab," *IEEE Robot. & Autom. Mag.*, vol. 3, no. 1, pp. 24–32, 1996.

Ameer Hamza Khan received the BS degree in electrical engineering from the Pakistan Institute of Engineering and Applied Sciences, Pakistan, in 2015. He worked as a research assistant at the Department of Computing, the Hong Kong Polytechnic University and joined the department as a Ph.D. student since 2016. His research interests include nonlinear optimization, metaheuristic algorithms, adaptive control, and machine learning.



Shuai Li received the B.E. degree in Precision Mechanical Engineering from Hefei University of Technology, China in 2005, the M.E. degree in Automatic Control Engineering from University of Science and Technology of China, China in 2008, and the Ph.D. degree in Electrical and Computer Engineering from Stevens Institute of Technology, USA in 2014. Dr. Li is currently an Associate Professor (reader) leading a robotic lab in college of Engineering, Swansea University, UK. His current research interests include dynamic neural networks, wireless sensor networks, robotic networks, machine learning, and other dynamic problems defined on a graph. Dr. Li is currently the Editor-in-Chief of International Journal of Robotics and Control, and is on the editorial board of the Neural Computing and Applications and the International Journal of Distributed Sensor Networks.

Xin Luo received the B.S. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2005 and the Ph.D. degree in computer science from Beihang University, Beijing, China, in 2011. In 2016, he joined the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China, as a Professor of computer science and engineering. He is currently also a Distinguished Professor of computer science with the Dongguan University of Technology, Dongguan, China. His current research interests include big data analysis and intelligent control. He has published over 100 papers (including over 30 IEEE TRANSACTIONS papers) in the above areas. Dr. Luo was a recipient of the Hong Kong Scholar Program jointly by the Society of Hong Kong Scholars and China Post-Doctoral Science Foundation in 2014, the Pioneer Hundred Talents Program of Chinese Academy of Sciences in 2016, and the Advanced Support of the Pioneer Hundred Talents Program of Chinese Academy of Sciences in 2018. He is currently serving as an Associate Editor for the IEEE/CAA JOURNAL OF AUTOMATICA SINICA, IEEE ACCESS, and Neurocomputing. He has received the Outstanding Associate Editor reward of IEEE ACCESS in 2018. He has also served as the Program Committee Member for over 20 international conferences.