Journal of Robotics and Control (JRC)
Volume 5, Issue 6, 2024
ISSN: 2715-5072, DOI: 10.18196/jrc.v5i6.23109

Trajectory Planning and Tracking Control for 6-DOF Yaskawa Manipulator based on Differential Inverse Kinematics

Ngo Xuan Khoat [1], Cao Thanh Vinh Hoa [2], Nguyen Bui Nguyen Khoa [3], Ngo Manh Dung [4*]
[1] Ho Chi Minh City University of Transport, Ho Chi Minh City, Vietnam
[2,3,4] Ho Chi Minh City University of Technology, VNU-HCM, Ho Chi Minh City, Vietnam
Email: [1] khoatnx.ncs@ut.edu.vn, [2] ctvhoa.sdh222@hcmut.edu.vn,
[3] nbnkhoa.sdh232@hcmut.edu.vn,[4] nmdung@hcmut.edu.vn
*Corresponding Author

***Abstract*—In the realm of robotics research, there is a strong focus on trajectory planning and control, driven by the increasing need to integrate robots across diverse industries. Drawing on the traditional Artificial Potential Field (for short, APF) method for path planning, the author proposes modifications on the force field calculation functions and time coefficients. These proposed functions improve the robot arm's movement to better interact with identified obstacles, regardless of distance conditions. This will help reduce calculation time compared to traditional methods. The research aims to enhance the operational system of the manipulator by developing an external program that interfaces with the central controller. The program guides the robot arm to follow a specific path using the Differential Inverse Kinematics (for short, DIK) method to ensure the smoothness of trajectory tracking. Facing the issue of the invertibility of the Jacobian matrix, the research team addressed it by adding a Moore–Penrose right pseudoinverse of the Jacobian and avoiding the shock velocity around the singularity using a Damping Constant technique. In this research, the proposed APF is validated and compared to the traditional method using MATLAB. The DIK method utilizes the optimal path from previous to control the Yaskawa MotoMINI manipulator - the physical robot arm system.**

***Keywords*—Trajectory Planning; Tracking Control; Differential Inverse Kinematics; Obstacle Avoidance**

I. Introduction

Manufacturers of robotic arms are developing integrated packages in their controllers [1] to assist users in path-planning by enabling them to "teach" the required coordinates [2], selecting trajectories, and offering user-friendly programming features to create a sequence of tasks for the manipulator [3]–[5]. Morover, the central controller allows communication with other devices via Ethernet, enabling users to connect and control the robot with a personal computer and expand it with cameras and computer vision algorithms [6]. However, teaching points and robot control using available functional packages are challenging for complex environments [7], and the communication packets only offer position control levels [8], resulting in latency and reduced flexibility. This paper aims to develop a computer program to solve trajectory planning and tracking control robot with low-latency communication. The computer and the central controller communicate via ROS, an open operating system with manufacturer-provided data packages [9], [10]. ROS assists users in creating an efficient communication environment and enables robot control at high speeds through the central controller [11].

Trajectory planning phase is a significant challenge in determining the best movement sequence [12] for a robot to reach its target without encountering obstacles [13], [14] in natural working environments. The path must meet the requirements for both optimal movement time [15] and total distance in other to conserve energy [16], [17]. Currently, this is a significant focus of research and experimental testing to ensure it meets the needs of robotic systems, such as 6-DOF. Common algorithms such as Rapidly-exploring Random Tree (RRT) [18]–[24], Fast Marching Tree (FMT) [25]–[30], as well as methods that utilize powerful algorithms in machine learning, such as Reinforcement Learning (RL) [31]–[33] and Artificial Neural Network (ANN) [34]–[38].

RRT and FMT are probabilistic sampling algorithms successfully developed and applied in planning phase. The core operation of these algorithms involves working on a randomly initialized map with lots of movable "nodes" [39]. The process begin at the start node and expands by connecting to the other within a given radius, similar to "tree branches", until it reaches the target node [40]. These algorithms effectively exploit the working environment without establishing obstacle positions based on the robot's configuration [41]. However, they need help finding paths in unstructured environments with numerous random obstacles [42]. Additionally, an excessive number of nodes or an inappropriate choice of exploration radius can lead to reduced computation speed and convergence [23]. Research on these algorithms often focuses on improving search capabilities and convergence or combining them with other algorithms for higher efficiency. Qingyang Gao [18] used the RRT and the Back

Propagation (BP) optimization algorithm to find the optimal path for controlling a humanoid system. Their results were significantly reduce the number of connection points in the trajectory compared to the basic algorithm. However, this method required a large amount of training data for the best model, and the trajectory generated was not smooth. Ying-Hao Yu [30] introduced the Slice-based Heuristic technique into the FMT algorithm for controlling the Human–robot Collaboration system. This research aims to reduce computation time and eliminate the need for prior knowledge of the working environment while still avoiding obstacles. However, the resulting path is too long and not smooth.

APF is a path-planning method that utilizes virtual force fields placed at key points in the working environment [43], [44]. These fields exert attraction and repulsion forces to guide the robot along a safe path [45], [46]. The Attraction force field pulls the robot toward the target point, while the Repulsion force field pushes the robot away from obstacles to prevent collisions. The APF was first proposed by Khalib in 1985 for systems such as autonomous cars and self-driving ships [47]. In systems with multiple operating devices, APF demonstrates its capability by forming various potential fields at critical points [48], [49], bringing efficiency and high safety to complex industrial operating systems [50].

The research conducted by Hao Li and his team showcased the computational efficiency and feasibility of the Artificial Potential Field (APF) algorithm for robotic manipulators [51]. This was demonstrated through simulations on a system with the plate number 3-DOF. Many publications have proposed enhancements and variations of the traditional APF due to its simple structure and low computational cost. For instance, Min Zhuang combined APF with the A* algorithm to shorten the travel distance for apple-picking robots [52]. At the same time, in 2023, Xinkai Xia [53] introduced another variant called the Velocity Potential Field (VPF) for obstacle avoidance path planning. The velocity-based approach is particularly suitable for high-precision applications such as human-robot interaction and medical robotics as it offers improved safety in control.

We have identified shortcomings in the mathematical functions and distance conditions used in the classical Artificial Potential Field (APF) method. These impact the quality of generated paths, causing abrupt changes in trajectory and oscillations near obstacles. We propose using alternative mathematical functions without distance conditions for the attractive and repulsive force fields, and mention changes in the math function of iteration to reduce computational time.

In the next phase, the tracking control supervises the robot's movement along a predefined trajectory. Passive control at the position level leads to movement errors [54]. Using an external position-level controller requires constant monitoring and causes significant control delays [55]. The experimental setup utilizes the YRC1000Micro controller to manage the Yaskawa MotoMINI robot arm via Ethernet, mainly using a UDP-based protocol [56]. Nonetheless, a smooth and precise end effector motion encounters significant limitations. The research by Sana Baklouti reveals conventional position control delays up to 0.78s (>0.6s) [57]. This delay timeframe is unsuitable for real-time communication, in which the transmitted packets accounting for nearly 70% of the delay. Thus, an overreliance on position-level control packages may pose challenges that must be addressed:

- *Limited Motion Planning*: The controller's inability to handle complex motion planning online. Online planning method allows real-time adjustments based on sensor's feedback [58], which is crucial for tasks requiring adaptation [59]. Position-level control necessitates pre-programmed paths, limiting the robot's ability to respond to the dynamic situations.
- *Inflexibility for Complex Kinematics*: Many robotic manipulators have complex kinematic structures, leading to multiple joint configuration and/or solutions to achieve a desired end-effector pose [60]. Traditional position control, which relies on pre-computed inverse kinematics, struggles with selecting the optimal solution amidst these nonlinearities. This inflexibility can lead to an inefficient motion or even a failure to reach specific poses in an adaptive planning environment [61].
- *Inability to Address Time Constraints*: Real-world applications often have specific time constraints for completing tasks [62], [63]. Position control offers limited control over the time it takes for the robot to reach a designated position. This lack of fine-grained control over motion speed makes it challenging to guarantee task completion within a desired time frame [64].

DIK is a velocity-level manipulator control technique used to replace complex traditional control methods. This technique helps to overcome the disadvantages of solving robot kinematic problems. To improve the control accuracy and overcome the problem of increased velocity when the manipulator passes through singular points, we propose to enhance the Close-loop controller and the Damping Constant technique.

The research contributions is: developing a fast and accurate path-planing algorithm that operates in real-time and establishing communication with the manipulator's central controller in minimal delay to ensure precise trajectory tracking control. From our analysis of various methods, we combine the proposed APF and DIK techniques for the manipulator control program. Our research focus on creating the integrated solutions for computer-controlled robot systems while maintaining high accuracy and real-time communication. We provide further details on the method enhancements in the following sections.

The remaining part of this article presents the contents in the following order: In section 2, we will introduce the manipulator structures and ROS system for seamless data communication

with the central controller. The impactful changes implemented within the Artificial Potential Field are highlighted in section 3. In section 4, the process of creating a control system is outlined, which is based on Differential Inverse Kinematics. This system is designed to calculate and execute precise control of the robot arm. Section 5 will showcase the results and thoroughly assess trajectory formation, tracking errors, and program response simulation. Finally, section 6 concludes the article and outlines future development plans.

II. System Description

The MotoMINI Robot, shown in Fig.1 and Table I, features six revolute joints, which can provide high acceleration capabilities and excellent performance. This versatile robot is commonly used for factory assembly, packaging, and sorting operations. With a maximum payload of 0.5 kg and a reach of up to 350 mm, the MotoMINI is well-suited for collaborative tasks such as assembly, product inspection, and classification. This product weights only 7kg, suitable for installation in confined spaces and is easy to maneuver. The MotoMINI stands out among other robots in its segment, boasting speeds that are 20% faster alongside exceptional processing capabilities, resulting in significantly enhanced operational efficiency.

To operate this robotic system, the software built on the ROS must handle communication tasks and control task. The control loop involves a computer running ROS Nodes and the YCR1000micro central controller equipped with integrated firmware running ROS1-MotoROS. In this process, offline path planning and computation are carried out by MATLAB software, while the robot's kinematic algorithms for control are computed in real time at a rate of 40 Hz. The planned trajectory is stored as a data file in the computer's hard drive for easy access by programs in the ROS ecosystem, which then engage in post-processing and initiate the control process. Fig. 1 depicts an overview of the system connections. In which:

- *Finding the best path*: Using the improved APF proposals to find the trajectory between two points without collision with obstacles in nature working environment.
- The *"data file"*: contains the pre-planned path data, including the coordinates of each point along the trajectory in the $(\mathbf{p_e}, \mathbf{\Delta t})$ motion configuration. Each line in the file corresponds to one point on the path.
- *"Path Planning Input"* Node: converts the $(\mathbf{p_e}, \mathbf{\Delta t})$ motion configuration to the $(\mathbf{p_e}, \mathbf{v_e})$ format. The node then further divides the points along the path into individual points with a time interval of 25ms (corresponding to 40Hz) between two consecutive points to create a smooth trajectory for the controller. The node also configures the requirements for the robot and the settings for generating control values for the robot.
- *"Smooth Trajectory Generation"* Node: the main node responsible for inverse kinematics calculations and sending control signals to the robot.
- *"Motoman ROS"* Node: contains nodes related to communication between the ROS nodes on the PC and the YRC1000micro firmware to update the status and receive control commands related to the robot, known as the Motoman ROS Driver.

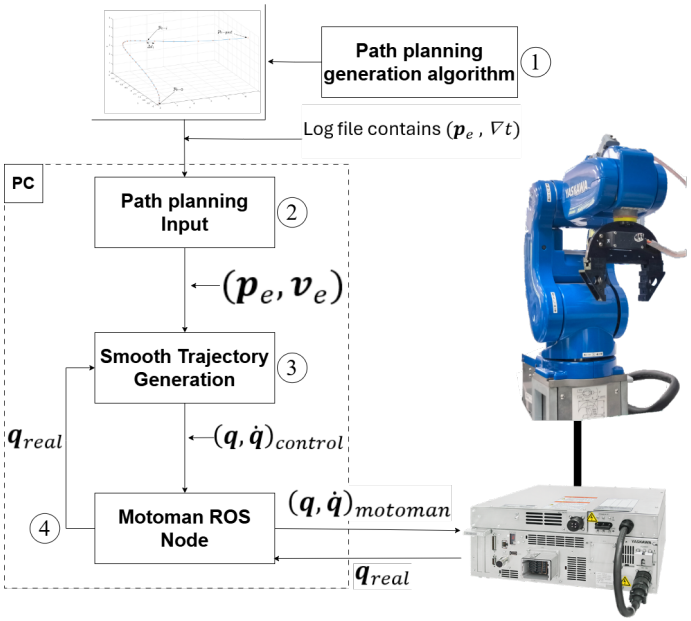


Fig. 1. An overview of the control process diagram.

TABLE I. MotoMINI Parameters

Axis	S	L	U	R	B	T
Max range (degree)	±170	+90 /-85	+90 /-50	±140	+210 /-30	±360
Max speed (rad/s)	5.5	5.5	7.3	10.5	10.5	10.5

III. Artificial Potential Field

A. Modified Attractive Potential Field

In the Cartesian workspace of the traditional APF method, the target is considered a "valley" which has an ability to generate an attractive force pulling the end-effector closer. Khatib proposes a fundamental equation characterizing the attractive force field in the traditional method [47]:

$$\mathbf{U}_{t_{att}}(\mathbf{p}) = \frac{1}{2} k_{att} d^2(\mathbf{p}, \mathbf{p}_{goal}) \quad (1)$$

Where p is the current position of the end-effector, p_{goal} is the target position and k_{att} is the gain of the attractive force. Hence $d(p, p_{goal})$ is the shortest distance in Cartesian space from the manipulator to the target and is determined by the Euclidean distance formula in (5).

The traditional attractive force's potential field function extends across the entire operating space and increases gradually

with distance. However, it only depends on distance, as the end-effector gets closer to the target point. In that case, the potential attraction will decrease, potentially causing an issue if an obstacle exist near the target point. To overcome this constraint, the research team recommends employing an exponential distance function to preserve the appealing quality of the target point, described as follows:

$$\mathbf{U}_{m_{att}}(\mathbf{p}) = k_{att} e^{\alpha d(\mathbf{p},\mathbf{p}_{goal})} \tag{2}$$

Where α is a coefficient that affects the magnitude of the attractive field. The attractive force is determined by the gradient vector of the attractive field ($\mathbf{F}_{att} = -\nabla \mathbf{U}_{att}$) throughout the considered workspace:

$$\begin{cases} \mathbf{F}_{t_{att}} & = -k_{att} d(\mathbf{p}, \mathbf{p}_{goal}) \nabla d(\mathbf{p}, \mathbf{p}_{goal}) \\ \mathbf{F}_{m_{att}} & = -k_{att} \alpha e^{\alpha d(\mathbf{p},\mathbf{p}_{goal})} \nabla d(\mathbf{p}, \mathbf{p}_{goal}) \end{cases} \tag{3}$$

In equation (3) $\nabla d(p, p_{goal})$ is the direction of the attractive force acting on the robot arm. it is defined by following (4):

$$\nabla d(\mathbf{p}, \mathbf{p}_{goal}) = \frac{\mathbf{p}, \mathbf{p}_{goal}}{d(\mathbf{p}, \mathbf{p}_{goal})} \tag{4}$$

$$d(\mathbf{p}, \mathbf{p}_{goal}) = \sqrt{(x - x_{goal})^2 + (y - y_{goal})^2 + (z - z_{goal})^2} \tag{5}$$

When $d(\mathbf{p}, \mathbf{p}_{goal})$ decreases, the attractive force is maintained at a certain level and cannot be equal to zero because of the nature of the exponential function. The parameter α reduces the attractive force for distant objects, avoiding excessive pulling force that could lead to collisions with nearby obstacles. $\nabla d(\mathbf{p}, \mathbf{p}_{goal})$ represents the direction of the applied force. If the distance approaches zero, the direction of movement will gradually vanish, allowing the end-effector to reach the target. In Fig. 2, we can observe that the proposed method makes the attractive force smaller, changing smoother than traditional.

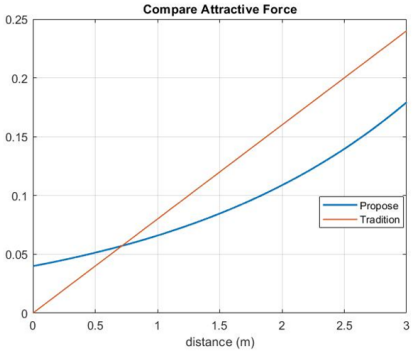


Fig. 2. Compare attractive force between two method.

B. Modified Repulsive Potential Field

The robot needs to identify obstacles in its workspace to avoid collisions. There is a repulsive potential field like a 'shield' surrounding the obstacle, altering the robot's movement direction when it gets close. Equation (6) deploy the essential characteristics of the repulsive potential field, as first described by Khatib [47]:

$$\mathbf{U}_{t_{rep}}(\mathbf{p}) = \begin{cases} \frac{1}{2} k_{rep} (\frac{1}{d(\mathbf{p},\mathbf{p}_{obs})} - \frac{1}{d_0})^2 & d(\mathbf{p}, \mathbf{p}_{obs}) < d_0 \\ 0 & d(\mathbf{p}, \mathbf{p}_{obs}) > d_0 \end{cases} \tag{6}$$

Where, k_{rep} is the gain of the repulsive field, $d(p, p_{obs})$ is the shortest distance in Catesian space from manipulator to that obstacle and d_0 is the size of the repulsive field region. When the robot arm is moving near the obstacle in this region ($d(p, p_{obs}) < d_0$), the repulsive force changes the moving trajectory of the moving manipulator by the suction field mentioned above.

The traditional method of restricting the distance to generate repulsive forces can lead to an abrupt changes in the manipulator's path, challenging operation and affecting the system's dynamic nature. Furthermore, this algorithm requires a careful choice of coefficients to avoid generating powerful repulsive forces. The authors recommend using a negative exponential function with respect to the distance to characterize the repulsive field around the obstacles while maintaining the core concept of the traditional repulsive potential field, as follows:

$$\mathbf{U}_{m_{rep}}(\mathbf{p}) = k_{rep} exp(-\frac{d^2(\mathbf{p}, \mathbf{p}_{obs})}{2d_0^2}) \tag{7}$$

The repulsive force $\mathbf{F}_{rep}$ impact the end-effector is determined by the gradient vector of the repulsive field ($\mathbf{F}_{rep} = -\nabla \mathbf{U}_{rep}$) around the obstacle:

$$\begin{cases} \mathbf{F}_{t_{rep}}(\mathbf{p}) & = \begin{cases} k_{rep} (\frac{1}{d(\mathbf{p},\mathbf{p}_{obs})} - \frac{1}{d_0}) \frac{\nabla d(\mathbf{p},\mathbf{p}_{obs})}{d^2(\mathbf{p},\mathbf{p}_{obs})} \\ 0 \end{cases} \\ \mathbf{F}_{m_{rep}}(\mathbf{p}) & = k_{rep} \frac{d(\mathbf{p},\mathbf{p}_{obs})}{d_0^2} exp(-\frac{d^2(\mathbf{p},\mathbf{p}_{obs})}{2d_0^2}) \nabla d(\mathbf{p}, \mathbf{p}_{obs}) \end{cases} \tag{8}$$

If the environment has many obstacles, the repulsive force generated will be equal to the sum of the forces of all obstacles acting on the robot $\mathbf{F}_{rep}(\mathbf{p}) = \sum_{i=1}^{n} F_{rep}(\mathbf{p}, \mathbf{p}_{obs}(i))$. Determining the distance from the end-effector to an obstacle with an indefinite shape poses challenges because it requires finding a point on the obstacle closest to the robot. Powerful computer vision-based algorithms can assist in recognizing the structural shape of the obstacles [65], [66], but the computation time also increases as the objects' complexity increases. Moreover, to avoid the obstacles in the APF approach can be done without high levels of detail. Therefore, Yu Chen proposed a solution to construct the shapes of the obstacles as spheres encompassing the objects entirely [41]. By this way, the distance from the manipulator to the obstacles can be more easily determined as follows:

$$d(\mathbf{p}, \mathbf{p}_{obs}) = \|\mathbf{p} - \mathbf{p}_{obs}\| - r_{obs} - r_{thick} \tag{9}$$

With $\|\mathbf{p} - \mathbf{p}_{obs}\|$ is calculated similarly to (5) but for the center of the sphere. r_{thick} is the thickness of the center

of the end-effector, r_{obs} is the radius of the smallest sphere encompassing the object. $\nabla d(\mathbf{p}, \mathbf{p}_{obs})$ is the direction of the repulsive acting on the robot arm, as follows:

$$\nabla d(\mathbf{p}, \mathbf{p}_{obs}) = \frac{\mathbf{p} - \mathbf{p}_{obs}}{\|\mathbf{p} - \mathbf{p}_{obs}\|} \tag{10}$$

Fig. 3 shows that the repulsive force in the traditional method appears abruptly and significantly when the robot moves close to the repulsive potential field of the obstacle. Meanwhile, the proposed function makes the force changes steadily as the distance gradually decreases to a safe threshold, resulting in smoother trajectories around obstacles and ensuring path quality and obstacle avoidance capability. Furthermore, when the manipulator is far from the obstacle, the repulsive magnitude is nearly zero.

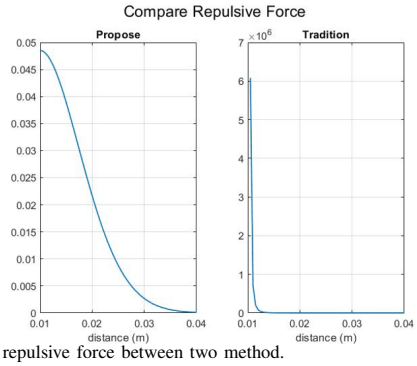


Fig. 3. Compare repulsive force between two method.

C. Iteration of Alogrithm

When the robot moves into the considered workspace, it will always be influenced by two forces: the attractive force generated at the target point and the repulsive force generated by obstacles throughout the operational space,following Fig. 4, forming a total force $\mathbf{F}_{all}$:

$$\mathbf{F}_{all}(\mathbf{p}) = \mathbf{F}_{m_{att}}(\mathbf{p}) + \mathbf{F}_{m_{rep}}(\mathbf{p}) \tag{11}$$

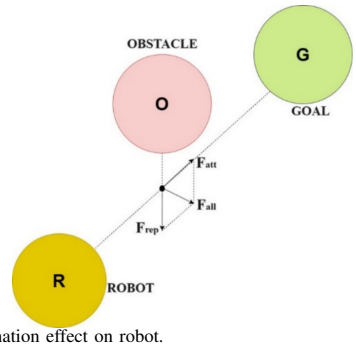


Fig. 4. Force combination effect on robot.

From (11), the algorithm is iteratively repeated according to (12) until the manipulator's end-effector moves to the target point, with p_t is the coordinate of robot in workspace at the $t-th$ iteration and λ is a coefficient of time to determine the next location of the machine under the impact of the force description from (11):

$$\mathbf{p}_t = \mathbf{p}_{t-1} + \lambda \mathbf{F}_{all} \tag{12}$$

In the traditional method, the coefficient λ is a constant. It is considered the time coefficient to determine the next position of the manipulator under the effect of the total force. In segments of the trajectory without obstacles, the manipulator needs a large force to move a long path. However, with a small λ, the robotic arm only moves a short distance before the algorithm recalculates the force. In subsequent calculations, the attractive force decreases, followed by distance, so the total force gradually decreases. This causes the algorithm to repeat many times to complete the trajectory.

Conversely, if λ is setted too large, the movement path near obstacles will not be smooth. This is because the resultant repulsive force here is extensive and exerted over a long period. The trajectory will continuously change, causing the system to appear to vibrate at this point. Therefore, the authors propose a new method that continuously updating the λ coefficient based on the distance between the manipulator's end-effector and the target point, as well as obstacles. It is defined below:

$$\lambda = \beta_1 + \beta_2 d(\mathbf{p}, \mathbf{p}_{goal}) + \beta_3 d(\mathbf{p}, \mathbf{p}_{obs}) \tag{13}$$

Where $\beta_1, \beta_2, \beta_3$ are the learning rate parameters depending on the current distance of the robot with obstacles and with the target point. When there are many obstacles in the environment, β_1 will consider with the nearest obstacle distance. β_1 is considered the basic coefficient for the time coefficient. β_2, β_3 are used for adjusting λ according to the current distance. The flowchart in Fig. 5 shows the process of proposed APF work in path-planning

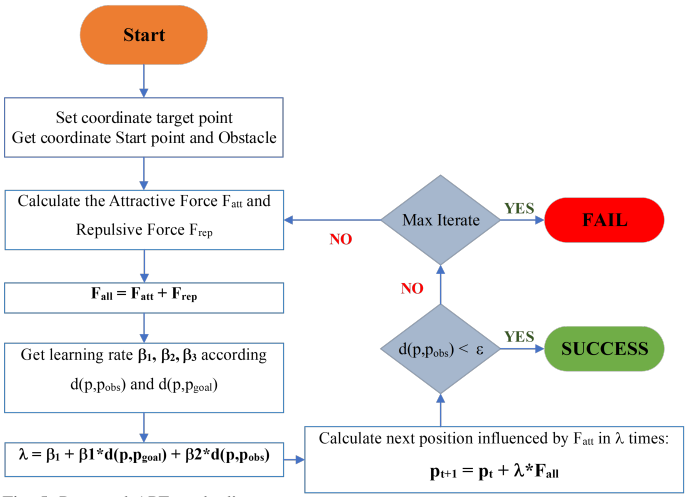


Fig. 5. Proposed APF method's process

IV. Differential Inverse Kinematics

In traditional position control for robots, significant latency is often observed due to the controller continuously checking the position. This process involves obtaining a pre-planned trajectory of end-effector positions (x, y, z) and performing inverse kinematics calculations to determine the corresponding joint angles $\mathbf{q}_i$ required to achieve each desired end-effector location. These calculated joint positions are then sent to the robot controller, which drives the motors to reach those positions. The control loop waits for confirmation from joint encoders that the commanded positions are achieved before sending the next set of commands [67]. The process sending the position commands and waiting for confirmation is repeated for every points along the trajectory, ultimately guiding the robot's end-effector through the planned path.

Acknowledging the limitations of the traditional method and the design requirements that need to be met for online motion control, the most suitable algorithm for the robot's motion is Differential Inverse Kinematics [68], [69]. This method proposed to calculate the average velocity $\mathbf{v_e} = (v_{xe}, v_{ye}, v_{ze})$ based on the time interval Δt between points p_e on the trajectory. To simplify the calculation of the average velocity, the specific formula is used to calculate the velocity v_{e_i} for any point at a given time step $\mathbf{i}$ is derived from the one-sided average velocity (focusing on the velocity for reaching the next point) $v_e = (\mathbf{p_{e_{i+1}}} - \mathbf{p_{e_i}})/\Delta t_i$.

From the perspective of robot kinematic motion control, starting with an initial motion configuration $(\mathbf{p_e}, \Delta t) = ((x_e, y_e, z_e), \Delta t)$ when applying the above formula yields a new motion configuration $(\mathbf{p_e}, \mathbf{v_e}) = ((x_e, v_e, z_e), (v_{xe}, v_{ye}, v_{ze}))$. This new configuration is directly related to the position and velocity of the robot's end-effector at each point along the trajectory shown in Fig. 6. This simplifies the kinematic motion design process. Since this type of motion configuration contains velocity parameters (instead of time), it is more compatible with the developed and proven robot kinematic equations.

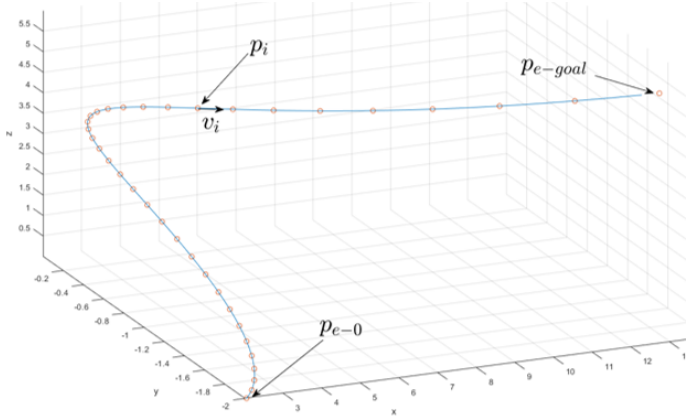


Fig. 6. Motion planning for the orbiting manipulator.

A. Concept of the Open-Loop Motion Control Method

The trajectory is defined as a series of points that represent the end-effector, containing both position and velocity information in the format: $[(\mathbf{p_{e_0}}, \mathbf{v_{e_0}}), \ldots, (\mathbf{p_{e_i}}, \mathbf{v_{e_i}}), (\mathbf{p_{e_{goal}}}, \mathbf{v_{e_{goal}}})]$. The robot control process will involve controlling the robot's joints value $\mathbf{q}$ to make the robot's end-effector move from the starting point $\mathbf{p_{e_0}}$, sequentially passing through the points $\mathbf{p_{e_i}}$ along the trajectory path with velocity $\mathbf{v_{e_i}}$, ultimately reaching the target point $\mathbf{p_{e_{goal}}}$. $\mathbf{p_e} = [\mathbf{x_e}, \phi_\mathbf{e}]^T$ is the information about the end-effector's position in the world-frame, including end-effector position parameters $\mathbf{x_e} = [x_e \quad y_e \quad z_e]$ and rotation direction parameters $\phi_\mathbf{e} = [\varphi \quad \vartheta \quad \psi]^T$ (using the fixed angel Roll-Pitch-Yaw). $\mathbf{v_e} = [\dot{x_e}, \dot{\phi_e}]^T$ is information about the end-effector's velocity move to the next point. The problem to be addressed is to build an inverse kinematics computation model, with the aim of continuously calculating the values of the joint variables [70].

In robot kinematics theory, there exists a velocity kinematics equation that describes the relation as mentioned above, which is written as follows:

$$\mathbf{v_e} = \begin{bmatrix} \dot{\mathbf{x}}_e \\ \omega_e \end{bmatrix} = \begin{bmatrix} \mathbf{J}_P(\mathbf{q}) \\ \mathbf{J}_O(\mathbf{q}) \end{bmatrix} \dot{\mathbf{q}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{14}$$

Where the matrix $\mathbf{J}(\mathbf{q})$ is the Jacobian velocity kinematics matrix, with a size of 6x6 (for the current MotoMINI robot configuration). Equation (14) includes two matrices $\mathbf{J_p}(\mathbf{q})$ and $\mathbf{J_o}(\mathbf{q})$, both of size 3x6. $\mathbf{J_p}(\mathbf{q})$ and $\mathbf{J_o}(\mathbf{q})$ describes the effect of joint velocity $\dot{\mathbf{q}}$ on the linear velocity $\dot{\mathbf{x}}$ and the angular velocity $\omega_\mathbf{e}$ of the end-effector. Based on the (14) to calculate $\dot{\mathbf{q}}$ follows:

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}\mathbf{v}_e \tag{15}$$

Subsequently, to inversely find the position of each robot joint angle $\mathbf{q}$, the ideal integration of the joint velocity found above must be taken, where $\mathbf{q}(\mathbf{0})$ is the robot joint angle position at the initial time:

$$\mathbf{q}(t) = \int_0^t \dot{\mathbf{q}}(\tau)d\tau + \mathbf{q}(0) \tag{16}$$

This inverse kinematics calculation technique is completely independent of the solvability of the traditional geometric inverse kinematics method based on robot structure [71]. The control diagram is shown in Fig. 7.

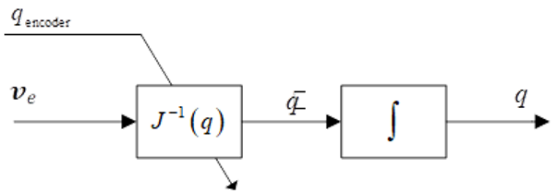


Fig. 7. Open-loop inverse kinematics algorithm.

The inverse kinematics algorithm calculates the joint values in the continuous-time domain and determines joint angle and velocity. It helps control the robot along a desired motion trajectory. But note that the Jacobian matrix must always be a square matrix and may not be invertible in specific robot configurations.

In addition, the approach also involves utilizing optimal integration to determine the solution, which is nearly impossible to carry out efficiently for computer-based numerical control systems. Thus, it is evident that this method has limitations: the non-invertibility of the Jacobian matrix affects the calculation over the entire subsequent trajectory, and the possibility of falling into the singularity point/region causes a loss of velocity control during the control process [72]. Furthermore, this open-loop calculation method makes it very difficult to avoid errors in the control response. To address the drawbacks of the method as mentioned above, in the process of building the control model, a combination of several computational techniques will be used:

- *Jacobian Pseudoinverse Matrix*: To limit the case where the Jacobian matrix cannot be inverted (loss of rank), the Jacobian pseudoinverse matrix calculation technique can be used with the locally minimization of the norm of joint's velocity vectors solution [73].
- *Damping Constant*: To minimize velocity shock when entering a neighborhood of a singularity point, the Jacobian pseudoinverse matrix can be adjusted by incorporating a damping coefficient.
- *Closed-Loop Algorithm*: To limit errors in control, it is necessary to design a closed-loop algorithm to improve control quality.

B. Jacobian Pseudoinverse Matrix Method

In robot motion control, some state positions may cause the Jacobian matrix to lose rank. Consequently the rank of the matrix is less than the number of degrees of freedom of the robot, leading (15) to infinite solutions. To avoid this, the joint velocity optimization method can be used to reduce the number of feasible solutions to an optimal solution. The quadratic cost functional of joint velocities to be minimized:

$$g\left(\dot{\mathbf{q}}\right) = \frac{1}{2}\dot{\mathbf{q}}^T\mathbf{W}\dot{\mathbf{q}} \tag{17}$$

Where $\mathbf{W}$ is a suitable $(n \times n)$ symmetric positive definite weighting matrix. A particular case occurs when the weighting matrix $\mathbf{W}$ is the identity matrix. This problem can be solved using the method of Lagrange multipliers. The sought optimal solution for $\dot{\mathbf{q}}$:

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}\mathbf{v}_e = \mathbf{J}^{\dagger} = \mathbf{J}^T\left(\mathbf{J}\mathbf{J}^T\right)^{-1}\mathbf{v}_e \tag{18}$$

Where $\mathbf{J}^{\dagger}$ is the Moore–Penrose right pseudoinverse of $\mathbf{J}$. With this optimal solution technique, the inverse velocity kinematics method is no longer constrained by the cases of rank deficiency in the Jacobian matrix. The consequence of finding this optimal solution is that the joint velocity vector $\dot{\mathbf{q}}$ obtained using this technique always yield the locally minimize-norm velocity vector optimum [74] (within the stopping criteria range during computation).

C. The Singularity Problem and the Jacobian Matrix with a Damping Constant

Another characteristic when using the inverse velocity kinematics method is related to singularity points [75]. The Jacobian matrix causes difficulties at the singularity position and its neighborhood. At singularity positions and the neighboring region, as the determinant of $\mathbf{J}$ approaches 0, it causes significant changes in the joint velocities [76]. A solution to overcome this problem is called the Damped Least Squares (DLS) inverse [75], [77]. In this case, the pseudoinverse matrix mentioned in (18) will be adjusted to become:

$$\mathbf{J}^{\star} = \mathbf{J}^T\left(\mathbf{J}\mathbf{J}^T + k^2\mathbf{I}\right)^{-1} \tag{19}$$

Where k is the damping vector that renders the inversion better conditioned, with positive and lower 1a positive value less than 1.

D. Closed-Loop Motion Controller Design

The final drawback in the calculation process is taking the ideal derivative in (18). If converting that equation to a discrete expression, it will have the form of:

$$\mathbf{q}(\mathbf{t_{k+1}}) = \mathbf{q}(\mathbf{t_k}) + \dot{\mathbf{q}}(\mathbf{t_k})\boldsymbol{\Delta}\mathbf{t} \tag{20}$$

Where $\dot{\mathbf{q}}$ is obtained by using (18) and (19), substituting the solution into equation (20) produces the formula to calculate joint value in discrete expression form:

$$\mathbf{q}(\mathbf{t_{k+1}}) = \mathbf{q}(\mathbf{t_k}) + \mathbf{J}^{\bigstar}(\mathbf{q}(\mathbf{t_k}))\mathbf{v_e}(\mathbf{t_k})\boldsymbol{\Delta}\mathbf{t} \tag{21}$$

In (21), $\mathbf{q}(t_{k+1})$ is the solution of the corresponding $\mathbf{v}_e(t_k)$, that is, the input is the desired velocity to the next point on the trajectory. When controlling velocity of a real robot, it is impossible to ensure that the robot always moves according to the set velocity due to software and mechanical issues [78], leading to the following position $\mathbf{q}(t_{k+1})$ having an error compared to the calculation. When this iterative calculation is repeated many times along the trajectory, a significant error will accumulate at the end of the trajectory.

This difficulty can be resolved by continuously updating the current end-effector coordinates $\mathbf{q}(t_k)$ from the robot's sensor [79], [80] finding the current position $\mathbf{q}_e$ after each solution step, with a predefined update cycle Δt. This is equivalent to defining the error between the desired point $\mathbf{q}_d$ and the current position of the end-effector $\mathbf{q}_e$. Let the error between these two points be defined as follows:

$$\mathbf{e} = \mathbf{p}_d - \mathbf{p}_e \tag{22}$$

From there, the system continuously references the position error between the current end-effector position and the desired point. Rewriting the velocity kinematics equation in terms of the variable $\mathbf{e}$, the velocity kinematics equation depending on the error value for each calculation step is obtained:

$$\dot{\mathbf{q}} = \mathbf{J}^{\star}\mathbf{K}\mathbf{e} \tag{23}$$

In which $\mathbf{K}$ is a positive definite matrix (in this research, a positive diagonal matrix is used) characterizing the rate of error convergence to zero. The larger this matrix $\mathbf{K}$ is, the faster the convergence rate, and the end-effector quickly returns to the desired position shown in Fig. 8.

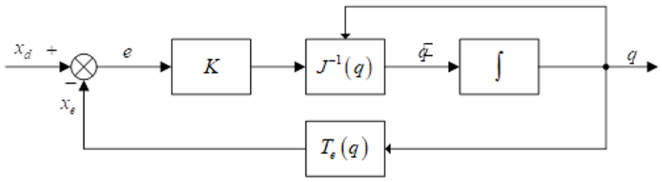


Fig. 8. Closed-loop inverse kinematics algorithm.

On the other hand, some limitations in the communication process, such as signal transmission quality affecting the update rate not being fast enough to meet the control capabilities of the model, can lead to poor control quality and overshoot [81].

V. Experimental Results

A. Environment Setup

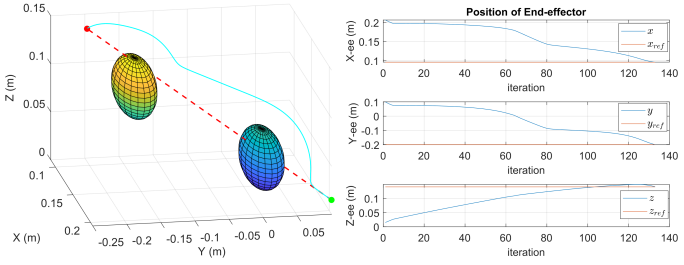
The experiments are performed on a Yaskawa MotoMINI robot with six revolute joints. In the working environment, there are two spherical obstacles with the same radius of 30mm. The start point is at [0.207 0.1 0.015] m, and the target point is at [0.096, -0.201, -0.14] m, the centers of the two obstacles are at [0.168 0.017 0.04] m and [0.117 -0.143 -0.09] m.

In the first step, following the diagram in Fig. 1, the MATLAB uses the initial parameters to find the best path based on the proposed method of APF. The second step, the input path is divided into individual points with a time interval 25ms. Next, the program guides the robot arm to follow the path using the proposed Differential Inverse Kinematics method. The final step, the program sends the control signal for robot tracking the path with the minimal error. Fig. 9 is the set up testing environment of our article.

B. Path Planning Result

The planning phase is performed and evaluated on MATLAB for both traditional and modified methods, providing the input values of the environment, along with the necessary parameters for each algorithm. The parameters of traditional APF is set: $d_0 = 0.01m, r_{thick} = 0.0005m, k_{att} = 0.07, k_{rep} = 10^{-7}$, and Modified APF is set: $d_0 = 0.01m, r_{thick} = 0.005m, k_{att} = 0.08, k_{rep} = 10^{-3}, \alpha = 0.55, d_1 = 0.005m$. To assess the impact of the parameter λ on the quality of the planned path, executing time, and the number of algorithm iterations, the traditional method will be conducted at three respective values: 0.05, 0.1, and 0.2. Fig. 10a illustrates the trajectory of the robotic arm, successfully moving from the starting point to the destination without collisions with two spherical obstacles. Fig. 10b describes the smooth and continuous change in the end-effector's position, avoiding sudden changes near obstacles throughout the entire space.

Fig. 9. Testing environment.



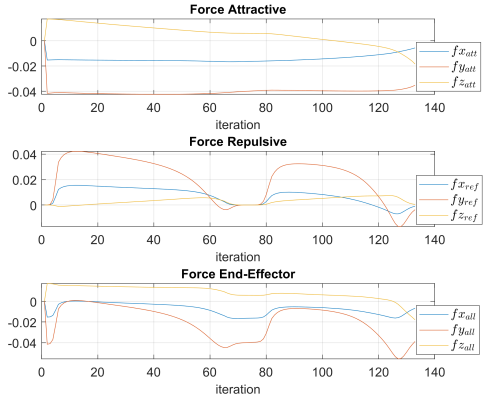
(a) Overview path (b) End-effector's pose

Fig. 10. Simulation results using the proposed method.

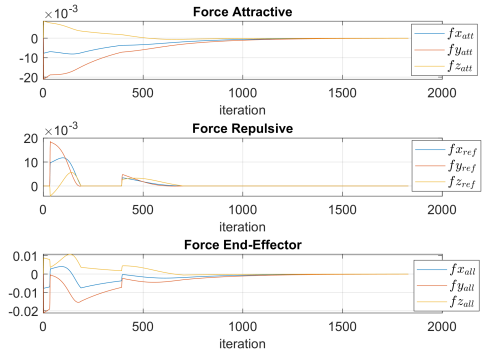
The relationship between the virtual force generated from the force field calculation and the time iteration coefficient λ is very close. It can be seen in the graph of Fig. 11b, Fig. 11c, Fig. 11d, with the same attractive and repulsive force field formations of the traditional method. However, when end-effector approaching the obstacle, the results are different. When λ is small, the force acts for a short time and then is recalculated in the next iteration so the change can be seen evenly and smoothly. Conversely, the extended application of thrust causes the trajectory to move far away. However, in the subsequent iteration, the end-effector moves beyond the influence region, resulting in a loss of thrust. As a result, the trajectory displays abrupt oscillations around the obstacle, as shown in Fig. 12b.

The proposed method generates virtual forces with smooth and uniform variations, avoiding sudden changes, as demonstrated in Fig. 11a. Along with ensuring that the λ coefficient is compatible with the current state of the manipulator, based on the distance from the desired force field points, the trajectory is significantly improved. The motion near the obstacle maintains a stable distance without oscillations, as depicted in Fig. 12a.

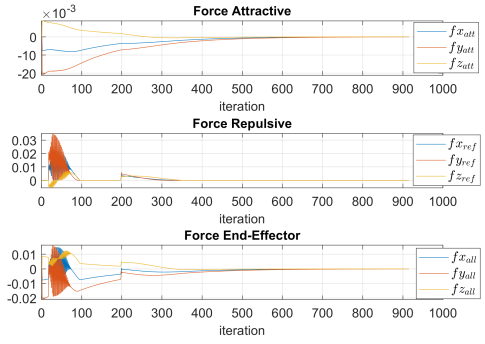
Table II shows the improvement in the calculation time of the path-finding algorithm when applying the proposed lambda



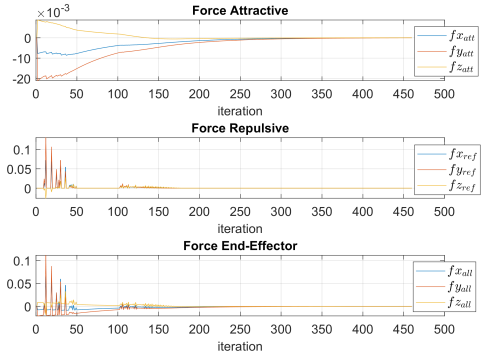
(a) Proposed method



(b) Traditional $\lambda = 0.05$

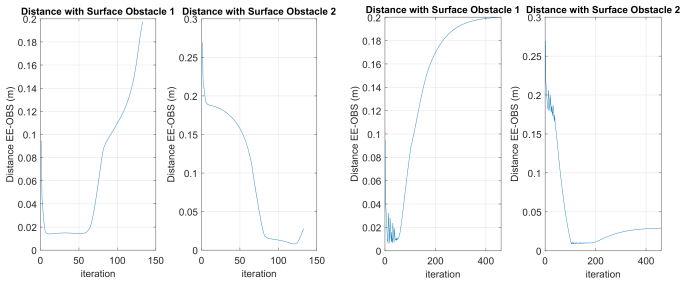


(c) Traditional $\lambda = 0.1$



(d) Traditional $\lambda = 0.2$

Fig. 11. Attractive, Repulsive and Total force.



(a) Proposed method (b) Traditional $\lambda = 0.2$

Fig. 12. Distance from end-effector to the surface obstacle.

coefficient calculation method. When choosing $\lambda = 0.005$, it will ensure that the quality of the formed trajectory is the most similar to the proposed method. However, the number of iterations is very large, it lead to 1829 iterate. Because of the large number of iterations, when controlling the robot to follow the trajectory, the completion time will also be considerable, approximately 78s more than proposed method. The evaluation results of the trajectory tracking controller presented below will use the results from the proposed method.

TABLE II. COMPARE TIME, DISTANCE TRAVEL, ITERATE BETWEEN TRADITIONAL $\lambda = 0.05, 0.1, 0.2$ AND PROPOSED METHOD

	$\lambda = 0.05$	$\lambda = 0.1$	$\lambda = 0.2$	**Modified**
Distance traveled(m)	0.3824	0.4221	0.5291	0.383
Iterate	1829	914	459	132
Time(s)	91.5	91.4	91.8	14.2

C. Trajectory Tracking Results

To illustrate the impact of the delay from traditional process, Fig. 13 visualizes the position control delay. This experiment was conducted with a 133-point trajectory. The total predefined time for that trajectory was 14.2 seconds. However, due to the inherent delays in the control loop, the robot took nearly 100 seconds to complete the trajectory. The delay between each point ranged from 0.6 to 1 second. This substantial delay between the planned and actual execution time highlights the limitations of the traditional position control methods regarding efficiency and responsiveness.

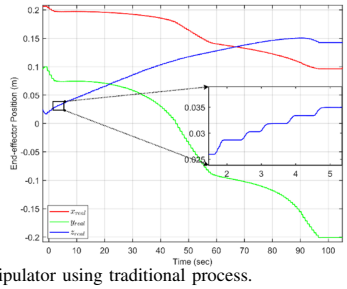
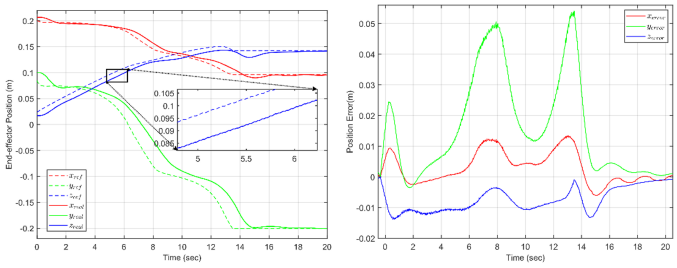


Fig. 13. Control manipulator using traditional process.

While utilizes the velocity control method according to (23), the control results are obtained based on a trajectory dataset

consisting of 133 points, with a total movement time of 14.2303 seconds. The results are achieved using a hardware consisting of an Intel i7 10875H CPU and a software included ROS1 Noetic. The results of applying the differential kinematic control algorithm are divided into two results: fast and slow velocity coefficient to clearly demonstrate the characteristics of tracking control and the importance of choosing the coefficient.

In the case of slow velocity, configure the coefficient $\mathbf{K} = 2.0, k = 0.05$. The position tracking results are depicted in Fig. 14a. The graphs illustrate reduced lag between the movement points, yielding an almost smooth trajectory with the point-to-point delays not exceeding 40ms. This reduction in latency is thanks to the ROS driver's ability to deliver the next point to the path-planning queue without delay, even if the robot is still moving. Additionally, Fig. 14b displays the position error during the moving process, which remains below 6cm. Some error peaks in the position error graph arise from the predefined trajectory's motion profile rather than the controller. In this case, abrupt velocity changes at some specific time points (e.g., seconds 8 and 14) lead to these peaks. However, the controller promptly detects and mitigates these errors, as evidenced by the valleys observed between the peaks.

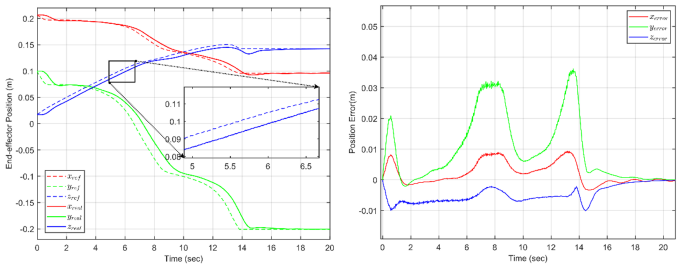


(a) Tracking path (b) Tracking error

Fig. 14. Slow speed control

In the case of high velocity, configure the coefficient $\mathbf{K} = 3.25, k = 0.05$. The obtained position tracking results are shown in Fig. 15. In the configuration with a faster tracking velocity setting, the results can clearly show the improvement in the shape of the tracking trajectory, tracking error, and convergence time. For the trajectory shape, the tracking trajectory shows a higher similarity to the planned trajectory compared to the slow tracking velocity case. To further demonstrate this, consider the position error graph, where there is no situation where the error exceeds 4cm. The tracking error also starts to converge early, reaching below 0.25cm and starting to converge at 16 seconds, which is significantly faster than the slow tracking velocity case.

In analyzing tracking delay, which reflects the temporal displacement between planned and actual robot paths, latency is measured by comparing timestamps at corresponding positions during constant velocity phases. The average tracking delay is 0.49 seconds, ranging from 0.46 to 0.52 seconds. This demonstrates an improvement over the conventional ROS-based position control method described in [57].



(a) Tracking path (b) Tracking error

Fig. 15. High speed control

VI. Conclusion

The paper has presented a distance-based exponential function as an alternative to traditional functions. MATLAB simulation results comparing the proposed method with the traditional approach show noticeable improvements in path quality, motion trajectory smoothness, and obstacle avoidance. The change in a mathematical function for the λ time coefficient significantly aids expedited pathfinding and efficient interaction with the working environment. In addition, the article introduces an external program to ensure close adherence to the trajectory with minimal communication delay. The closed-loop controller continuously feedbacks the end-effector's position, which helps to guide the robot to follow the trajectory more accurately. With this implementation, enabled by the DIK technique, were evaluated through integration with the YRC1000Micro controller and ROS Motoman driver, demonstrating their effectiveness in the results.

However, certain challenges remain, particularly in scenarios involving simple environments with two static obstacles. Further testing of the method's coefficients is necessary for broader application. Additionally, attention to the robot body's position structure is essential to prevent collisions with obstacles. Significant errors persist in the tracking control process, especially during continuous trajectory changes, rendering it unsuitable for applications demanding high precision, such as human-machine interaction control.

In the future work, with the findings from this study, we will challenge complex environments with multiple or moving obstacles and expand the system with computer vision applications to enhance accuracy in location determination and collision avoidance. Furthermore, future advancements in controlling, such as incorporating velocity feedback at each joint, promise even more precise control and improved tracking accuracy during operation.

Acknowledgment

We acknowledge the support of time and facilities from Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for this study.

REFERENCES

[1] M. Marsono, Y. Yoto, A. Suyetno, and R. Nurmalasari, "Design and programming of 5 axis manipulator robot with GrblGru open source software on preparing vocational students' robotic skills," *Journal of Robotics and Control*, vol. 2, no. 6, pp. 539–545, 2021, doi: 10.18196/jrc.26134.

[2] A. Alipour, M. J. Mahjoob, and A. Nazarian, "A new 4-DOF robot for rehabilitation of knee and ankle-foot complex: Simulation and experiment," *Journal of Robotics and Control*, vol. 3, no. 4, pp. 483–495, 2022, doi: 10.18196/jrc.v3i4.14759.

[3] J. Qi, Q. Yuan, C. Wang, X. Du, F. Du, and A. Ren, "Path planning and collision avoidance based on the rrt* fn framework for a robotic manipulator in various scenarios," *Complex & Intelligent Systems*, vol. 9, no. 6, pp. 7475–7494, 2023, doi: 10.1007/s40747-023-01131-2.

[4] X. Cao, X. Zou, C. Jia, M. Chen, and Z. Zeng, "Rrt-based path planning for an intelligent litchi-picking manipulator," *Computers and Electronics in Agriculture*, vol. 156, pp. 105–118, 2019, doi: 10.1016/j.compag.2018.10.031.

[5] N. P. Higueras, F. Caballero, and L. Merino, "Teaching robot navigation behaviors to optimal rrt planners," *International Journal of Social Robotics*, vol. 10, pp. 235–249, 2018, doi: 10.1007/s12369-017-0448-1.

[6] J. A. Kay, D. C. Mazur and R. A. Entzminger, "Basics of Communication Networks for Electrical Engineers in the Forest Products Industries," *2018 IEEE IAS Pulp, Paper and Forest Industries Conference*, pp. 1-5, 2018, doi: 10.1109/PPIC.2018.8502239.

[7] H. Ryu and Y. Park, "Improved informed rrt* using gridmap skeletonization for mobile robot path planning," *International Journal of Precision Engineering and Manufacturing*, vol. 20, no. 11, pp. 2033–2039, 2019, doi: 10.1007/s12541-019-00224-8.

[8] K. Kunal, A. Z. Arfianto, J. E. Poetro, F. Waseel, and R. A. Atmoko, "Accelerometer implementation as feedback on 5 degree of freedom arm robot," *Journal of Robotics and Control*, vol. 1, no. 1, 2020, doi: 10.18196/jrc.1107.

[9] A. R. Al Tahtawi, M. Agni, and T. D. Hendrawati, "Small-scale robot arm design with pick and place mission based on inverse kinematics," *Journal of Robotics and Control*, vol. 2, no. 6, 2021, doi: 10.18196/jrc.26124.

[10] J. D. Téllez, R. S. G. Ramírez, J. Pérez-Pérez, J. E. Carreón, and M. A. C. Rosales, "ROS-based controller for a two-wheeled self-balancing robot," *Journal of Robotics and Control*, vol. 4, no. 4, pp. 491–499, 2023, doi: 10.18196/jrc.v4i4.18208.

[11] A. D. Sabiha, M. A. Kamel, E. Said, and W. M. Hussein, "Ros-based trajectory tracking control for autonomous tracked vehicle using optimized backstepping and sliding mode control," *Robotics and Autonomous Systems*, vol. 152, 2022, doi: 10.1016/j.robot.2022.104058.

[12] P. Chotikunnan and Y. Pititheeraphab, "Adaptive P control and adaptive fuzzy logic controller with expert system implementation for robotic manipulator application," *Journal of Robotics and Control*, vol. 4, no. 2, pp. 217–226, 2023, doi: 10.18196/jrc.v4i2.17757.

[13] N. Chao, Y. K. Liu, H. Xia, A. Ayodeji, and L. Bai, "Grid-based rrt* for minimum dose walking path-planning in complex radioactive environments," *Annals of Nuclear Energy*, vol. 115, pp. 73–82, 2018, doi: 10.1016/J.ANUCENE.2018.01.007.

[14] H. I. Lin and M. F. Hsieh, "Robotic arm path planning based on three-dimensional artificial potential field," in *2018 18th International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2018, pp. 740–745.

[15] S. Števo, I. Sekaj, and M. Dekan, "Optimization of robotic arm trajectory using genetic algorithm," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 1748–1753, 2014, doi: 10.3182/20140824-6-ZA-1003.01073.

[16] M. A. Mousa, A. T. Elgohr, and H. Khater, "Path planning for a 6 dof robotic arm based on whale optimization algorithm and genetic algorithm," *Journal of Engineering Research*, vol. 7, no. 5, pp. 160–168, 2023, doi: 10.21608/erjeng.2023.237586.1256.

[17] P. Nithya, L. P. PS, G. E. Benjamin, and J. Venkateswaran, "Optimal path planning and static obstacle avoidance for a dual arm manipulator used in on-orbit satellite servicing," *IFAC-PapersOnLine*, vol. 53, no. 1, pp. 189–194, 2020, doi: 10.1016/j.ifacol.2020.06.032.

[18] Q. Gao, Q. Yuan, Y. Sun, and L. Xu, "Path planning algorithm of robot arm based on improved rrt* and bp neural network algorithm," *Journal of King Saud University-Computer and Information Sciences*, vol. 35, no. 8, 2023, doi: 10.1016/j.jksuci.2023.101650.

[19] I.-B. Jeong, S.-J. Lee, and J.-H. Kim, "Quick-rrt*: Triangular inequality-based implementation of rrt* with improved initial solution and convergence rate," *Expert Systems with Applications*, vol. 123, pp. 82–90, 2019, doi: 10.1016/j.eswa.2019.01.032.

[20] F. Kiani, A. Seyyedabbasi, R. Aliyev, M. U. Gulle, H. Basyildiz, and M. A. Shah, "Adapted-rrt: novel hybrid method to solve three-dimensional path planning problem using sampling and metaheuristic-based algorithms," *Neural Computing and Applications*, vol. 33, no. 22, pp. 15569–15599, 2021, doi: 10.1007/s00521-021-06179-0.

[21] Y. Li, W. Wei, Y. Gao, D. Wang, and Z. Fan, "Pq-rrt*: An improved path planning algorithm for mobile robots," *Expert systems with applications*, vol. 152, 2020, doi: 10.1016/j.eswa.2020.113425.

[22] J. Qi, H. Yang, and H. Sun, "Mod-rrt* : A sampling-based algorithm for robot path planning in dynamic environment," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 8, pp. 7244–7251, 2021, doi: 10.1109/TIE.2020.2998740.

[23] K. Wei and B. Ren, "A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved rrt algorithm," *Sensors*, vol. 18, no. 2, 2018, doi: 10.3390/s18020571.

[24] W. Xinyu, L. Xiaojuan, G. Yong, S. Jiadong, and W. Rui, "Bidirectional potential guided rrt* for motion planning," *IEEE access*, vol. 7, pp. 95046–95057, 2019, doi: 10.1109/ACCESS.2019.2928846.

[25] W. Gao, Q. Tang, J. Yao, Y. Yang, and D. Yu, "Heuristic bidirectional fast marching tree for optimal motion planning," in *2018 3rd Asia-Pacific Conference on Intelligent Robot Systems*, pp. 71–77, 2018, doi: 10.1109/ACIRS.2018.8467243.

[26] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International journal of robotics research*, vol. 34, no. 7, pp. 883–921, 2015, doi: 10.1177/0278364915577958.

[27] B. Wu, X. Wu, N. Hui, and X. Han, "Trajectory planning and singularity avoidance algorithm for robotic arm obstacle avoidance based on an improved fast marching tree," *Applied Sciences*, vol. 14, no. 8, 2024, doi: 10.3390/app14083241.

[28] J. Xia, Z. Jiang, H. Zhang, R. Zhu, and H. Tian, "Dual fast marching tree algorithm for human-like motion planning of anthropomorphic arms with task constraints," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 5, pp. 2803–2813, 2021, doi: 10.1109/TMECH.2020.3047476.

[29] J. Xu, K. Song, D. Zhang, H. Dong, Y. Yan, and Q. Meng, "Informed anytime fast marching tree for asymptotically optimal motion planning," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 6, pp. 5068–5077, 2021, doi: 10.1109/TIE.2020.2992978.

[30] Y.-H. Yu and Y.-T. Zhang, "Collision avoidance and path planning for industrial manipulator using slice-based heuristic fast marching tree," *Robotics and Computer-Integrated Manufacturing*, vol. 75, 2022, doi: 10.1016/j.rcim.2021.102289.

[31] M. F. Rahman and N. Sharma, "Reinforcement learning based approach for urban resource allocation and path planning problems," in *2020 International Conference on Intelligent Data Science Technologies and Applications*, pp. 115–118, 2020, doi: 10.1109/IDSTA50958.2020.9264062.

[32] C. Wang, X. Yang, and H. Li, "Improved q-learning applied to dynamic obstacle avoidance and path planning," *IEEE Access*, vol. 10, pp. 92879–92888, 2022, doi: 10.1109/ACCESS.2022.3203072.

[33] T. Yan, Y. Zhang, and B. Wang, "Path planning for mobile robot's continuous action space based on deep reinforcement learning," in *2018 International Conference on Big Data and Artificial Intelligence*, pp. 42–46, 2018, doi: 10.1109/BDAI.2018.8546675.

[34] A. Abdi, D. Adhikari, and J. H. Park, "A novel hybrid path planning method based on q-learning and neural network for robot arm," *Applied Sciences*, vol. 11, no. 15, 2021, doi: 10.3390/app11156770.

[35] Y.-W. Chen and W.-Y. Chiu, "Optimal robot path planning system by using a neural network-based approach," in *2015 international automatic control conference (CACS)*, pp. 85–90, 2015, doi: 10.1109/CACS.2015.7378370.

[36] Y. Li, R. Cui, Z. Li, and D. Xu, "Neural network approximation based near-optimal motion planning with kinodynamic constraints using rrt," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 11, pp. 8718–8729, 2018, doi: 10.1109/TIE.2018.2816000.

[37] J. Lu, T. Zou, and X. Jiang, "A neural network based approach to inverse kinematics problem for general six-axis robots," *Sensors*, vol. 22, no. 22, 2022, doi: 10.3390/s22228909.

[38] W. Wang, M. Zhu, X. Wang, S. He, J. He, and Z. Xu, "An improved artificial potential field method of trajectory planning and obstacle avoidance for redundant manipulators," *International Journal of Advanced Robotic Systems*, vol. 15, no. 5, 2018, doi: 10.1177/1729881418799562.

[39] B. Wu, X. Wu, N. Hui, and X. Han, "Trajectory planning and singularity avoidance algorithm for robotic arm obstacle avoidance based on an improved fast marching tree," *Applied Sciences*, vol. 14, no. 8, p. 3241, 2024, doi: 10.3390/app14083241.

[40] L. Chen, Y. Shan, W. Tian, B. Li, and D. Cao, "A fast and efficient double-tree rrt* -like sampling-based planner applying on mobile robotic systems," *IEEE/ASME transactions on mechatronics*, vol. 23, no. 6, pp. 2568–2578, 2018, doi: 10.1109/TMECH.2018.2821767.

[41] Y. Chen, L. Chen, J. Ding, and Y. Liu, "Research on real-time obstacle avoidance motion planning of industrial robotic arm based on artificial potential field method in joint space," *Applied Sciences*, vol. 13, no. 12, 2023, doi: 10.3390/app13126973.

[42] A. Hidalgo-Paniagua, J. P. Bandera, M. Ruiz-de Quintanilla, and A. Bandera, "Quad-rrt: A real-time gpu-based global path planner in large-scale real environments," *Expert Systems with Applications*, vol. 99, pp. 141–154, 2018, doi: 10.1016/j.eswa.2018.01.035.

[43] Z. Chen, L. Ma, and Z. Shao, "Path planning for obstacle avoidance of manipulators based on improved artificial potential field," in *2019 Chinese Automation Congress (CAC)*, pp. 2991–2996, 2019, doi: 10.1109/CAC48633.2019.8996467.

[44] S. N. Gai, R. Sun, S. J. Chen, and S. Ji, "6-dof robotic obstacle avoidance path planning based on artificial potential field method," in *2019 16th International Conference on Ubiquitous Robots (UR)*, pp. 165–168, 2019, doi: 10.1109/URAI.2019.8768792.

[45] N. Zhang, Y. Zhang, C. Ma, and B. Wang, "Path planning of six-dof serial robots based on improved artificial potential field method," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 617–621, 2017, doi: 10.1109/ROBIO.2017.8324485.

[46] H. Zhaochu, H. Yuanlie, and Z. Bi, "Obstacle avoidence path planning for robot arm based on mixed algorithm of artificial potential field method and rrt," *Industrial Engineering Journal*, vol. 20, no. 2, 2017, doi: 10.3969/j.issn.1007-7375.e17-2002.

[47] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986, doi: 10.1177/027836498600500106.

[48] A. Sepehri and A. M. Moghaddam, "A motion planning algorithm for redundant manipulators using rapidly exploring randomized trees and artificial potential fields," *IEEE Access*, vol. 9, pp. 26059–26070, 2021, doi: 10.1109/ACCESS.2021.3056397.

[49] H. Zhang, Y. Zhu, X. Liu, and X. Xu, "Analysis of obstacle avoidance strategy for dual-arm robot based on speed field with improved artificial potential field algorithm," *Electronics*, vol. 10, no. 15, 2021, doi: 10.3390/electronics10151850.

[50] Q. Yuan, J. Yi, R. Sun, and H. Bai, "Path planning of a mechanical arm based on an improved artificial potential field and a rapid expansion random tree hybrid algorithm," *Algorithms*, vol. 14, no. 11, 2021, doi: 10.3390/a14110321.

[51] H. Li, Z. Wang, and Y. Ou, "Obstacle avoidance of manipulators based on improved artificial potential field method," in *2019 IEEE international conference on Robotics and Biomimetics (ROBIO)*, pp. 564–569, 2019, doi: 10.1109/ROBIO49542.2019.8961506.

[52] M. Zhuang, G. Li, and K. Ding, "Obstacle avoidance path planning for apple picking robotic arm incorporating artificial potential field and a* algorithm," *IEEE Access*, vol. 11, pp. 100070-100082, 2023, doi: 10.1109/ACCESS.2023.3312763.

[53] X. Xia, T. Li, S. Sang, Y. Cheng, H. Ma, Q. Zhang, and K. Yang, "Path planning for obstacle avoidance of robot arm based on improved potential field method," *Sensors*, vol. 23, no. 7, 2023, doi: 10.3390/s23073754.

[54] I. J. Meem, S. Osman, K. M. H. Bashar, N. I. Tushar, and R. Khan, "Semi wireless underwater rescue drone with robotic arm," *Journal of Robotics and Control*, vol. 3, no. 4, pp. 496–504, 2022, doi: 10.18196/jrc.v3i4.14867.

[55] A. E. Newir, Y. A. Abdelfattah, and M. M. Rashed, "Upgrading gryphon puma robot arm using ros-moveit," *MSA Engineering Journal*, vol. 2, no. 2, pp. 1212–1224, 2023, doi: 10.21608/MSAENG.2023.293681.

[56] V.-H. Nguyen, H.-N. Nguyen, M.-T. Ho, and T.-S. Nguyen, "Classifying laboratory tools using robot arm associated with computer vision," in *2022 7th National Scientific Conference on Applying New Technology in Green Buildings (ATiGB)*, pp. 11–18, 2022, doi: 10.1109/ATiGB56486.2022.9984091.

[57] S. Baklouti, G. Gallot, J. Viaud, and K. Subrin, "On the improvement of ROS-based control for teleoperated yaskawa robots," *Applied Sciences*, vol. 11, no. 16, 2021, doi: 10.3390/app11167190.

[58] N. E. Du Toit and J. W. Burdick, "Robot motion planning in dynamic, uncertain environments," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 101–115, 2012, doi: 10.1109/TRO.2011.2166435.

[59] M. V. Liarokapis and A. M. Dollar, "Learning task-specific models for dexterous, in-hand manipulation with simple, adaptive robot hands," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2534–2541, 2016, doi: 10.1109/IROS.2016.7759394.

[60] W. J. Wilson, C. C. W. Hulls, and G. S. Bell, "Relative end-effector control using cartesian position based visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 684–696, 1996, doi: 10.1109/70.538974.

[61] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014, doi: 10.1109/ACCESS.2014.2302442.

[62] J. K. Behrens, R. Lange, and M. Mansouri, "A constraint programming approach to simultaneous task allocation and motion scheduling for industrial dual-arm manipulation tasks," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8705–8711, 2019, doi: 10.1109/ICRA.2019.8794022.

[63] K. E. C. Booth, T. T. Tran, G. Nejat, and J. C. Beck, "Mixed-integer and constraint programming techniques for mobile robot task planning," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 500–507, 2016, doi: 10.1109/LRA.2016.2522096.

[64] S. Xu, M. Perez, K. Yang, C. Perrenot, J. Felblinger, and J. Hubert, "Determination of the latency effects on surgical performance and the acceptable latency levels in telesurgery using the dv-trainer® simulator," *Surgical endoscopy*, vol. 28, pp. 2569–2576, 2014, doi: 10.1007/s00464-014-3504-z.

[65] L. D. Hanh and V. D. Cong, "Path following and avoiding obstacle for mobile robot under dynamic environments using reinforcement learning," *Journal of Robotics and Control*, vol. 4, no. 2, pp. 157–164, 2023, doi: 10.18196/jrc.v4i2.17368.

[66] K. Yamtuan, T. Radomngam, and P. Prempraneerach, "Visual servo kinematic control of delta robot using YOLOv5 algorithm," *Journal of Robotics and Control*, vol. 4, no. 6, pp. 818–831, 2023, doi: 10.18196/jrc.v4i6.19102.

[67] P. F. Hokayem and M. W. Spong, "Bilateral teleoperation: An historical survey," *Automatica*, vol. 42, no. 12, pp. 2035–2057, 2006, doi: 10.1016/j.automatica.2006.06.027.

[68] D. A. Drexler and I. Harmati, "Joint constrained differential inverse kinematics algorithm for serial manipulators," *Periodica Polytechnica Electrical Engineering and Computer Science*, vol. 56, no. 4, pp. 95–104, 2012, doi: 10.3311/ppee.7163.

[69] M. Li, X. Luo, and L. Qiao, "Inverse kinematics of robot manipulator based on bode-cs algorithm," *Machines*, vol. 11, no. 6, 2023, doi: 10.3390/machines11060648.

[70] S. Qiu and M. R. Kermani, "Inverse kinematics of high dimensional robotic arm-hand systems for precision grasping," *Journal of Intelligent & Robotic Systems*, vol. 101, no. 70, 2021, doi: 10.1007/s10846-021-01349-7.

[71] W. Shanda, L. Xiao, L. Qingsheng, and H. Baoling, "Existence conditions and general solutions of closed-form inverse kinematics for revolute serial robots," *Applied Sciences*, vol. 9, no. 20, p. 4365, 2019, doi: 10.3390/app9204365.

[72] D. N. Nenchev and R. Iizuka, "Emergent humanoid robot motion synergies derived from the momentum equilibrium principle and the distribution of momentum," *IEEE Transactions on robotics*, vol. 38, no. 1, pp. 536–555, 2022, doi: 10.1109/TRO.2021.3083195.

[73] M. Sekiguchi and N. Takesue, "Fast and robust numerical method for inverse kinematics with prioritized multiple targets for redundant robots," *Advanced Robotics*, vol. 34, no. 16, pp. 1068–1078, 2020, doi: 10.1080/01691864.2020.1780151.

[74] A. Taherian, A. H. Mazinan, and M. Aliyari-Shoorehdeli, "Image-based visual servoing improvement through utilization of adaptive control gain and pseudo-inverse of the weighted mean of the jacobians," *Computers & Electrical Engineering*, vol. 83, 2020, doi: 10.1016/j.compeleceng.2020.106580.

[75] S. R. Buss and J.-S. Kim, "Selectively damped least squares for inverse kinematics," *Journal of Graphics Tools*, vol. 10, no. 3, pp. 37–49, 2005, doi: 10.1080/2151237X.2005.10129202.

[76] P. P. Rebouças Filho, S. P. P. da Silva, V. N. Praxedes, J. Hemanth, and V. H. C. de Albuquerque, "Control of singularity trajectory tracking for robotic manipulator by genetic algorithms," *Journal of computational science*, vol. 30, pp. 55–64, 2019, doi: 10.1016/j.jocs.2018.11.006.

[77] A. S. Deo and I. D. Walker, "Overview of damped least-squares methods for inverse kinematics of robot manipulators," *Journal of Intelligent and Robotic Systems*, vol. 14, pp. 43–68, 1995, doi: 10.1007/BF01254007.

[78] L. Lu, J. Zhang, J. Y. H. Fuh, J. Han, and H. Wang, "Time-optimal tool motion planning with tool-tip kinematic constraints for robotic machining of sculptured surfaces," *Robotics and Computer-Integrated Manufacturing*, vol. 65, 2020, doi: 10.1016/j.rcim.2020.101969.

[79] L. Sciavicco and B. Siciliano, "A solution algorithm to the inverse kinematic problem for redundant manipulators," *IEEE Journal on Robotics and Automation*, vol. 4, no. 4, pp. 403–410, 1988, doi: 10.1109/56.804.

[80] M. Sekiguchi and N. Takesue, "Fast and robust numerical method for inverse kinematics with prioritized multiple targets for redundant robots," *Advanced Robotics*, vol. 34, no. 16, pp. 1068–1078, 2020, doi: 10.1080/01691864.2020.1780151.

[81] S. H. J. Lund, P. Billeschou, and L. B. Larsen, "High-bandwidth active impedance control of the proprioceptive actuator design in dynamic compliant robotics," *Actuators*, vol. 8, no. 4, 2019, doi: 10.3390/act8040071.