Nanotechnology Perceptions
ISSN 1660-6795
www.nano-*ntp.com*

A 6-DOF Robot-Time Optimal Trajectory Planning-Based MOPSO Algorithm

Siddharth Gupta[1], Riddhi Garg[2*]

[1]*Research Scholar, Dept of Mathematics, IFTM University, Moradabad, 244001*
[2*]*Prof. Dr. Riddhi Garg, Dept of Mathematics, IFTM University, Moradabad, 244001*
[*]*Corresponding Author Mail; riddhigarg5@gmail.com*

Time-optimal trajectory planning is a critical approach for enhancing work efficiency and decreasing expenses, and it holds significant relevance in real-world robot application scenarios. a novel approach for efficiently planning the fastest path for a 6-degree-of-freedom robot by using a Multi-Objective Particle Swarm Optimization algorithm demonstrate in This paper. This work focus on optimizing multiple objectives concurrently, including minimizing trajectory time and ensuring smooth and efficient robot motion. The trajectory planning is based on both forward and inverse kinematics; create certain accurate and precise control over the robot's movement. The MOPSO algorithm is employed to search for optimal joint configurations, taking into account the conflicting nature of objectives such as minimizing time and maximizing efficiency. Through iterative optimization, the algorithm refines the trajectory parameters to achieve a well-balanced solution. The proposed system is validated through simulation studies, comparing its performance against existing trajectory planning methods. The results demonstrate the effectiveness of the Multi-Objective PSO algorithm in producing time-optimal trajectories for 6-DOF robots while maintaining smooth motion. The approach provides a promising solution for real-world applications, offering enhanced efficiency and adaptability in various robotic tasks.

Keywords: 6-DOF,Industrial robot, Trajectory planning, MOPSO.

I INTRODUCTION

In modern decades, there has been a important rise in the implementation of robotic manufacturing methods across several industries, particularly in the area of architecture. Robotic fabrication has the potential to enhance processes in various applications. It can advance the sustainability of additive manufacturing, leading to reduced waste and energy savings. As well, it can be used for both standardized and customized construction assembly tasks. Robotic manufacturing demonstrates its advantageous capabilities for intricate concrete constructions (Agustí-Juan et al., 2017), the creation of lightweight wood plates via the combination of robotic milling and hand assembly (Krieg et al., 2015), and the process of brick laying using mobile robots .In contrast to CNC technologies, which are primarily utilized in timber prefabrication to produce standardized components such as beams and plates, robotic fabrication excels in the customization of movements for creating intricate timber joints. This is made possible by the ability to execute complex manipulator trajectories. Nevertheless, the

sophisticated nature of the design leads to a more complex trajectory, resulting in longer process times. Robotic chainsaw cutting is widely used in wood manufacturing sectors, however in order to optimize the process, certain features of the operation need to be carefully considered. The surfaces involved in the cutting process exhibit variations, and the sequence in which the cutting is performed on these surfaces greatly affects the effectiveness of the operation. When faced with intricate couplings, the process of designing the route for a robotic chainsaw cutting operation may be laborious and creating pathways manually does not provide the most optimal answer. In order to create an optimal trajectory, it is necessary to fulfill certain criteria such as reducing the duration and decreasing the rapid, abrupt motions of the robotic controller. These rapid, abrupt motions result in strong jolts to the end effectors and may diminish the precision of the cut or perhaps harm the chainsaw. There is a wide range of research on robotic automation construction (Labonnote et al. 2016) and different techniques for optimizing trajectories.However, there is limited research on a specific methodology for optimizing trajectory planning in the fabrication of large-scale architectural joints. Thus, this research utilizes a specific situation to examine the practicality of trajectory optimization strategies.

Robot trajectory planning involves determining the desired path and goal position for a robot, and then adjusting the angle of rotation of each joint in a timely manner to guide the end effectors along a specified trajectory towards the target point. Planning trajectory in joint space is more straightforward and easy compared to that in Cartesian space. Thus, it is customary to assign many fixed locations to the terminations of various robotic arms. Subsequently, the robot's track points are calculated by the use of inverse kinematics, enabling the conversion from Cartesian space to joint coordinate space. Subsequently, track points are utilized to do interpolation using diverse spline functions, polynomial functions, or alternative curve shapes, resulting in the derivation of expressions pertaining to the temporal values of each joint variable for robot. Furthermore, considering the mechanical attributes of the robot, it is essential to restrict the speed and acceleration of each joint within the permissible range. Hence, it is essential to enhance velocity and acceleration of each arm joint, not only to guarantee seamless functioning of the joint but also to minimize wear and impact, hence extending operational lifespan of the robot.

Problem Statement-Robotic systems, mostly those with 6-DOF like Stanford and PUMA 560 robots, play a pivotal role in various industrial applications. capable motion planning is crucial for optimizing the presentation of these robots; ensure precise and rapid movements while avoiding collisions with obstacles in their workspace. The confront in trajectory planning involve finding the optimal path that reduce various objectives, such as travel time, energy consumption, and jerk, while meeting the constraints imposed by the robotic system. Current trajectory planning methods often focus on single-objective optimization, neglecting the intricate trade-offs between conflicting criteria. Established techniques, like minimum-time algorithms, may not fully address the complexities introduce through factors like viscous friction and the need for jerk minimization. Moreover, ensuring collision-free paths in dynamic environments requires sophisticated algorithms that can adapt to changing conditions.

II LITERATURE REVIEW

Suqin He et.al. (2022): Suggested the TOS-TIC technique, which aims to optimize online planning for continuous multi-axis trajectories, providing maximum efficiency. This technology is very efficient in terms of computer resources and may be used to contemporary automation and intelligent robotic systems. Wenjie Wang et.al. (2020): Proposed a method for trajectory planning that utilizes a 3-5-3 polynomial interpolation technique and an enhanced cuckoo search algorithm. This strategy guarantees trajectories that are optimized for time while adhering to velocity limitations, surpassing the performance of conventional approaches such as cuckoo search, particle swarm optimization, and genetic algorithms.Yalun Wen et.al. (2023): Created an algorithm that plans the most efficient route for robot manipulators, taking into account path constraints and avoiding collisions. The approach, via orthogonal collocation and numerical optimization, attains optimum time and jerk while preventing collisions. The method is verified by doing simulations and tests on a robot with six degrees of freedom.Lizhen Xia et.al. (2023): Addressed trajectory planning for rehabilitation robots assisting hemiplegia patients. Utilized B-spline and crow search algorithm to optimize energy impact and achieve multi-objective optimization. The proposed algorithm significantly reduced impact and energy consumption, showing promise in rehabilitation robot applications.

Xiao Hu et.al. (2023): This paper introduces a method for trajectory planning that aims to optimize time, using an improved version of the Simplified Particle Swarm Optimization (ISPSO) technique. The methodology used 3-5-3 polynomial interpolation and velocity restrictions, showcasing enhanced optimization compared to basic particle swarm methods for trajectory planning. Peiyao Shen et.al. (2020): Developed a trajectory planning algorithm that can generate continuous paths while considering acceleration and route constraints in real-time. The program also includes a mechanism to balance between smooth cruising motion and time-optimized motion. The algorithm offers flexibility in adjusting proportion of cruise and time-optimal motions, enhancing efficiency for various robotic tasks.

Optimal Techniques

The main objective of trajectory planning optimization approaches is to develop algorithms that minimize the time required for a task. This is motivated by the need to enhance efficiency in the manufacturing industry (Gasparetto and Zanotto 2010).The proposal suggests using convex optimization to address time trajectory planning by transforming time-optimal issues into convex optimum control problems. However, it is worth noting that this approach typically overlooks the influence of viscous friction,.Nevertheless, the optimization of time-optimal movement for robots might become non-convex when taking into account the effects of friction that is viscous (Shen et al., 2019). In response to this issue, researchers Abu-Dakka et al. (2015) devised a novel evolutionary algorithm that aims to optimize time and ensure collision-free trajectories in intricate surroundings.

The proposal to use actuator jerks as an optimization measure aims to address the restriction of current methods that consider robots to be rigid entities (Constantinescu and Croft 2000). Minimizing jerk, together with optimizing time, leads to enhanced tracking capacity and

reduced excitation of resonant frequencies (Gasparetto et al., 2015; Kyriakopoulos and Saridis, 1988). Another optimization goal is to minimize energy consumption, resulting in smoother trajectories that put less strain on robot manipulators .Several spline interpolation techniques, including cubic B-spline functions, fifth-order B-spline, and Lagrange interpolation, have used to create trajectories in accordance with the energy reduction target (Gasparetto and Zanotto 2007; Sato et al. 2007; Luo et al. 2015).

Additionally, hybrid optimizations have suggested, which include combining time and energy or time and jerk as targets for optimum trajectory planning (Balkan 1998; Shiller 1996; Gasparetto and Zanotto 2007, 2008; Zanotto et al. 2011). Nevertheless, these techniques often need the deliberate manipulation of weights in the objective function to regulate the balance between competing criteria.

In order to tackle issues with many objectives and eliminate the need for manual weight modifications, researchers have used evolutionary algorithms such as particle swarm optimization (PSO) and adaptive genetic algorithms (AGA) (Ata and Myo 2005; Saravanan et al. 2008). Particle Swarm Optimization (PSO), which draws inspiration from the collective movement of birds, has shown to be successful in discovering optimum routes for mobile robots and managing the complex task of planning robot courses with many objectives and unknown conditions (Eberhart and Kennedy 1995; Zhao and Yan 2005; Zhang et al. 2013). The use of AGA, which is based on natural genetic systems and natural selection, has been shown to enhance robot efficiency by optimizing time intervals between trajectory sections (Davis 1991; Liao et al. 2010).

PUMA 560

The PUMA 560 is a robotic arm used in industrial settings. It has six degrees of freedom, meaning it can move in six different directions, and all of its joints can rotate. The theoretical component of this experiment provides a concise overview of the PUMA 560 robot. The theory for numerical calculations was derived from a diverse range of sources, including articles, and the internet. The simulation part involves the creation of a virtual model using a JavaScript application. This model is then used to explore the forward kinematics issue. To get more details on other facets of the PUMA 560 and robotics, visitors are recommended to consult the provided sources.

Figure 1: PUMA 560 [https://mr-iitkgp.vlabs.ac.in/exp/forward-kinematics/theory.html]

The Programmable Universal Machine for Assembly, often referred to as PUMA SHOWING IN Figure 1, is an industrial robotic arm created by Victor Scheinman at Unimation in 1978. PUMA is available in several models, such as PUMA 260, PUMA 560, PUMA 761, and so on. Figure 2 displays the link-frame assignments in the position that corresponds to all joint angles being zero. The frame {0} is coincident with frame [1] when it is zero. It is important to mention that, like many other industrial robots, this particular robot has joint axes for joints 4, 5, and 6 that all meet at the same location. This point of intersection also happens to be the origin of frames {4}, {5}, and {6}. Moreover, the axes 4, 5, and 6 are perpendicular to each other. The schematic representation of this wrist device may be shown in Figure 4. This experiment details the process of determining the forward kinematics of the PUMA 560 robot using a virtual model. The forward kinematics issue pertains to the correlation between distinct joints of robot manipulator and precise location and orientation of tool or end effectors.

General Terminology in Robotics:

Workspace: The workspace refers to the set of frames that a robot's end-effectors can reach. It may be described as a manifold of accessible frames.

Accuracy: Accuracy is the measure of a robot's capability precisely places its wrist end at a certain target point inside work volume. It is determined based on the robot's spatial resolution. The outcome varies based on specific technology and size of the control increments.

Repeatability: Repeatability is a statistical concept that is closely linked to accuracy. When a robot joint repeatedly travels by same angle from a certain position, under consistent external circumstances, it consistently fails to hit the goal by a significant margin. If the same mistake occurs again, it indicates a high level of repeatability and a low level of accuracy.

Safety: Ensuring human safety and minimizing the impact force between humans and robots is a crucial need for robots designed to interact with humans in a pleasant manner.

Forward Kinematics: Forward kinematics (FK) involves the construction of a Denavit-Hartenberg (D-H) transformation matrix using Puma's parameters taken from a D-H parameter table provided below:

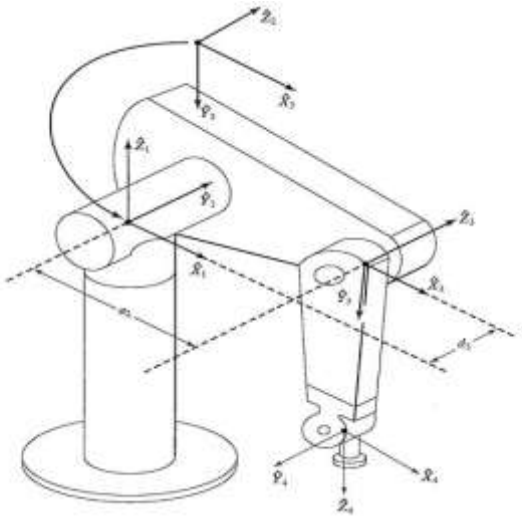


Figure 2: Kinematic parameters and frame assignments for PUMA 560 manipulator.

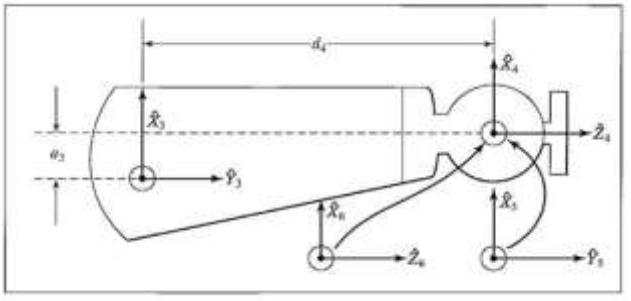


Figure 3: Kinematic parameters and frame assignments for forearm of the PUMA 560 manipulator.

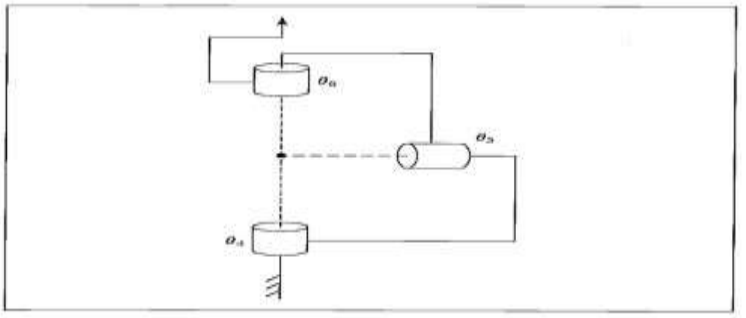


Figure 4: Schematic of a 3R wrist in which all three axes intersect at a point and are mutually orthogonal.

Table 1. Puma 560 D-H parameter table

$\mathbf{Link_i}$	$\mathbf{a_i-1(m)}$	$\mathbf{a_i-1(m)}$	$\mathbf{d_i(m)}$	$\boldsymbol{\theta}$
1	0	0	0	$\boldsymbol{\theta_1}$
2	−90	0	0	$\boldsymbol{\theta_2}$
3	0	$\mathbf{a_2}$	$\mathbf{d_3}$	$\boldsymbol{\theta_3}$
4	−90	$\mathbf{a_3}$	$\mathbf{d_4}$	$\boldsymbol{\theta_4}$
5	90	0	0	$\boldsymbol{\theta_5}$
6	−90	0	0	$\boldsymbol{\theta_6}$

Transformation matrices of six joints for Puma 560 robot

The Denavit-Hartenberg (D-H) parameters and transformation matrices for the PUMA560 robot, a conventional six-arm-type robot. The D-H parameters are used to model the kinematics of the robot, providing a systematic way to describe the relationship between successive links in a robotic arm.

The transformation matrices for each joint

$$\begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformation matrix from frame 0 to frame 1 (0T1):

$$\begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & 0 \\ 0 & 0 & -1 & d_2 \\ -\sin(\theta_2) & -\cos(\theta_2) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformation matrix from frame 1 to frame 2 (1T2):

$$\begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & a_2 \\ \sin(\theta_3) & \cos(\theta_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformation matrix from frame 2 to frame 3 (2T3):

$$\begin{bmatrix} \cos(\theta_4) & -\sin(\theta_4) & 0 & a_3 \\ 0 & 0 & -1 & d_4 \\ -\sin(\theta_4) & -\cos(\theta_4) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformation matrix from frame 3 to frame 4 (3T4)

$$\begin{bmatrix} \cos(\theta_5) & -\sin(\theta_5) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin(\theta_5) & \cos(\theta_5) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformation matrix from frame 4 to frame 5 (4T5):

$$\begin{bmatrix} \cos(\theta_6) & -\sin(\theta_6) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin(\theta_6) & -\cos(\theta_6) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformation matrix from frame 5 to frame 6 (5T6):

Finally, the comprehensive transformation matrix from base frame (frame 0) to end-effector frame (frame 6) is obtained by multiplying these individual transformation matrices:

0T6=0T1·1T2·2T3·3T4·4T5·5T6

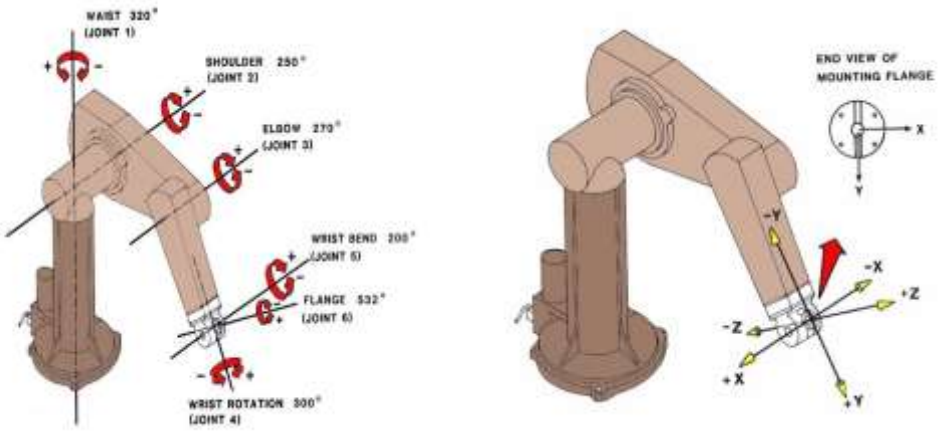


Figure 5 Puma kinematic diagrams

The link pole coordinate system of the PUMA560 robot may be determined based on link pole coordinate system shown in Figure 5. The transformation matrix for each link can be constructed using the following method.

III PROPOSED SYSTEM

In this proposed system, the forward and inverse kinematics equations of Stanford (6 DOF) and PUMA 560 robots are solved, enabling precise motion control. The robot is moved from one point to another in the workspace, and an optimization technique, specifically a multi-objective search-based optimization, is apply to find best path, optimizing both efficiency and other relevant objectives. Velocity control is implemented to achieve a steady state during motion. To address obstacles in the path, a hybridization technique is employed, combining forward and inverse kinematics to calculate optimized paths while ensuring collision avoidance. The calculations involve determining the joint angles (theta) and Cartesian coordinates (x, y, z) for motion planning. The entire motion is visualized in a 3D graphical user interface (GUI), enhancing the comprehensibility of the robot's trajectory. A comprehensive comparison with previous works is conducted, evaluating the proposed system's parameters and results, highlighting advancements in path optimization, obstacle avoidance, and motion control.

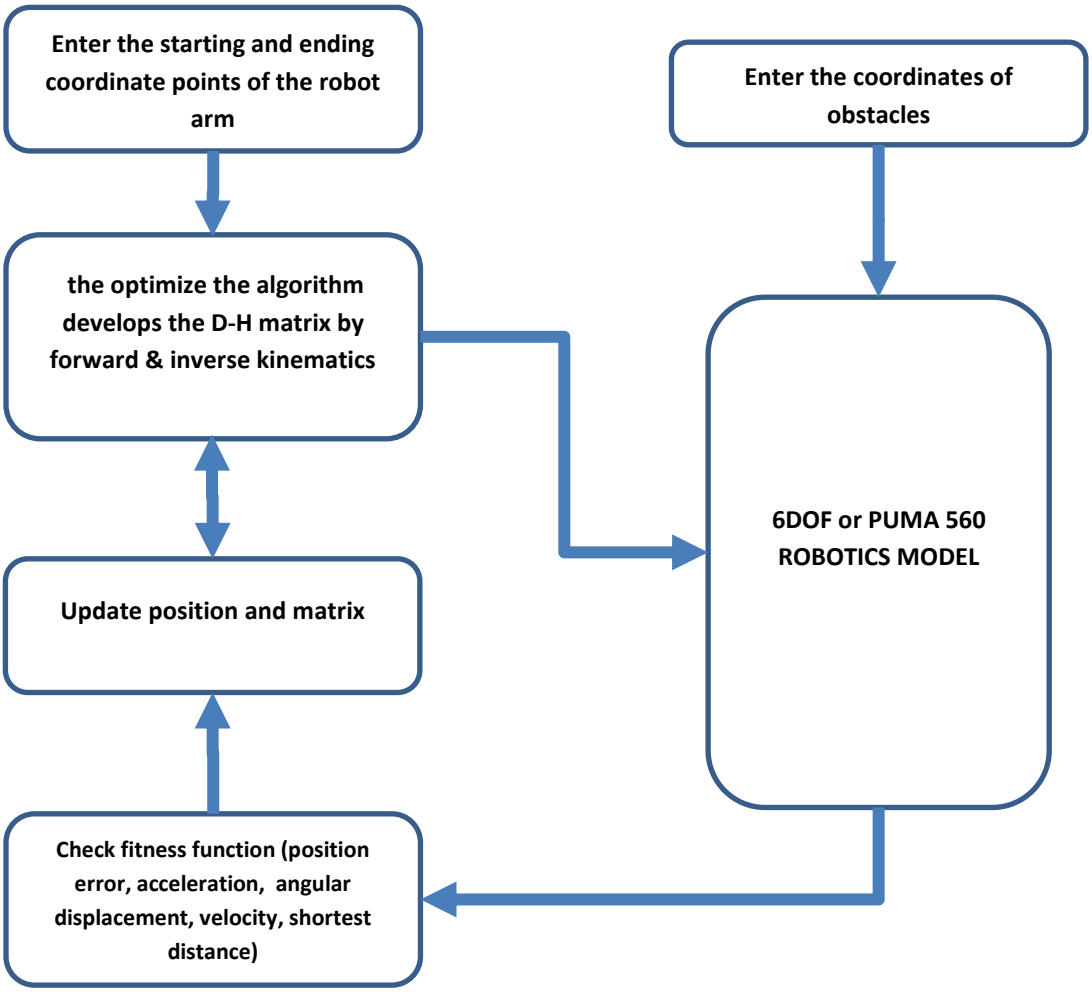


Figure 6 proposed system diagram

Figure 6 showing the process of optimizing the trajectory of a 6-DOF robotic arm (like the PUMA 560) using a combination of forward and inverse kinematics, updating the robot's position and matrix, checking various fitness functions (position error, acceleration, angular displacement, velocity, shortest distance), and considering obstacles. Below is a general outline of the algorithm based on your description:

Initialize Robot and Environment:

- Define the robot model (e.g., PUMA 560).
- Specify the initial and target coordinates for the end-effector.

Forward Kinematics:

- Use the Denavit-Hartenberg (D-H) parameters to construct transformation matrix from base frame to end-effectors frame.
- Update the robot's position and matrix.

Inverse Kinematics:

- Develop the inverse kinematics algorithm to determine joint angles that achieve the desired end-effector coordinates.
- Update the joint angles and recalculate the transformation matrix.

Check Fitness Functions:

- Define fitness functions based on your optimization criteria.
- Common fitness functions may include:
- Position error: The difference between the desired and actual end-effector positions.
- Acceleration: Consider acceleration constraints to ensure smooth motion.
- Angular displacement: Monitor the rotation of joints.
- Velocity: Keep track of joint velocities.
- Shortest distance: Minimize the distance traveled by the end-effector.

Optimization Algorithm:

- Implement an optimization algorithm (like a genetic algorithm, particle swarm optimization, or another heuristic optimization method) to adjust joint angles iteratively.
- Use the defined fitness functions as objective functions for optimization.

Update and Iterate:

- Update the joint angles based on the optimization results.
- Recalculate the forward kinematics to update the robot's position.
- Iterate the optimization process until convergence or a specified number of iterations.

Consider Obstacles:

- Integrate obstacle avoidance strategies into the optimization process.
- Adjust the optimization criteria to avoid collisions and ensure a safe trajectory.

Optimization Algorithm

The MATLAB function ObjectiveFunction_col appears to be a custom objective function used in the context of optimization for collision avoidance in robotic motion planning. The function takes parameters including PP1 (presumably perturbations or adjustments to joint angles), the

robot model (Robot), the current joint configuration (theta), and positions of the robot (p_pos) and obstacles (p_obj). The function calculates the forward kinematics of the robot with adjusted joint angles, computes the resulting position, and calculates the absolute differences between the positions of the obstacles and the adjusted robot position, as well as the position of a specified point (0.5) and the adjusted robot position. The function then computes the mean of these differences, representing a measure of collision avoidance. This objective function can be used within an optimization algorithm to guide the search towards joint configurations that avoid collisions with obstacles.

Multi-Objective Particle Swarm Optimization (MOPSO)

The proposed approach define three functions related to MOPSO for robot motion preparation and collision avoidance.

Objective Function_col: This function represents the objective function for the MOPSO algorithm with collision avoidance. It calculates the forward kinematics of the robot with adjusted joint angles (PP1). Its position is compared to a specified point (p_obj) and an obstacle avoidance point (p_pos). The mean of the absolute differences between the robot's position and the specified point is returned as the objective value (o).

Objective Function:

This function represents the objective function for the standard MOPSO algorithm. It calculates the forward kinematics of robot with adjusted joint angles (PP1). Its position is compared to a specified point (p_pos1).The mean of the absolute differences between the robot's position and the specified point is returned as the objective value (o).Additionally, position errors, joint angles, and fitness values are saved to separate files.

MOPSO_standard:

This function implements the standard MOPSO algorithm for optimizing the Objective Function. It uses a population of particles with random positions and velocities, updates their positions and velocities iteratively, evaluates fitness, and maintains the best global and personal solutions. The final result is the best solution found.

MOPSO_collision:

This function extends the MOPSO algorithm to handle collision avoidance using ObjectiveFunction_col.It uses a modified objective function that considers both the robot's position and collision avoidance. The algorithm aims to find joint configurations that minimize the mean absolute differences between the robot's position and a specified point while avoiding collisions with obstacles.

Objective (Fitness) Function:

Let pos_targetpos_target be the desired end-effector position.

The objective function, J, is the sum of squared errors:

$J(\theta)= \sum\sum_{i}^{3}(pos\_target_i - pos\_actual_i)^{2}$

Where θ represents the vector of joint angles.

Gradient Calculation: Calculate the gradient of the objective function with respect to the joint angles, $\nabla J(\theta)$.

Update Rule (Gradient Descent): Update the joint angles using the gradient descent rule: $\theta_{old} - \alpha \cdot \nabla J(\theta_{old})$ where α is the learning rate.

Algorithm:

Initialize:

fix an initial guess for joint angles (oldθ).

Choose a learning rate (α).

indicate the target end-effector position (pos_targetpos_target).

Iterative Update:

reiterate until convergence or a maximum number of iterations:

compute the actual end-effector position (pos_actualpos_actual) using forward kinematics.

assess the objective function (J).

Calculate the gradient (∇J).

Update joint angles using the gradient descent rule.

test for convergence.

IV SIMULATION RESULT

the implementation of various aspects related to robotic motion planning, optimization, and collision avoidance for a 6-DOF Stanford robot. The code begins by defining the robot's kinematics and trajectory points, and then utilizes MOPSO to find initial and final joint configurations. It proceeds to generate multiple paths between these points, applying optimization techniques to find the most efficient one. The script incorporates velocity and acceleration control, visualizing the robot's movement in a 3D GUI. Furthermore, it introduces obstacles, triggering a collision avoidance mechanism through a hybridization technique. The optimization process is monitored through convergence plots, and the code concludes with a comparison of joint velocities and accelerations, fitness values, and collision avoidance results.

The implementation seems comprehensive, covering forward and inverse kinematics, trajectory planning, optimization, and collision avoidance for the 6-DOF robotic system.

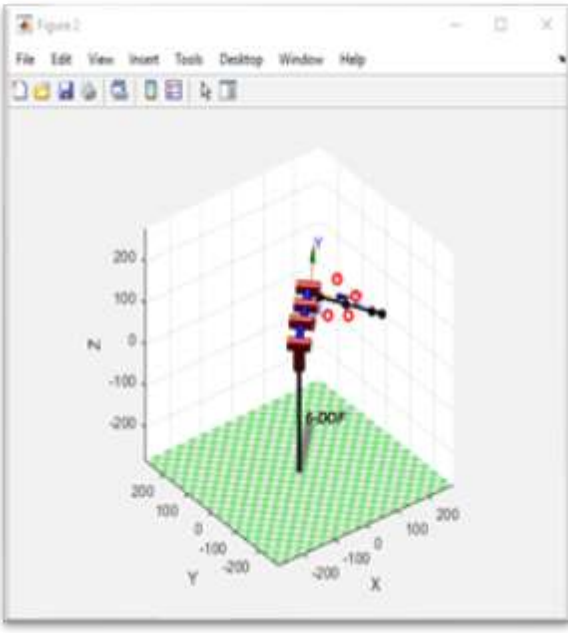


Figure 7 DOF robot

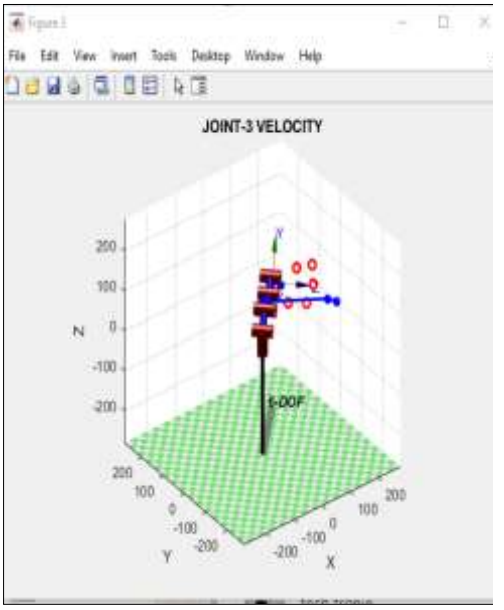


Figure 7 DOF robot joint velocity

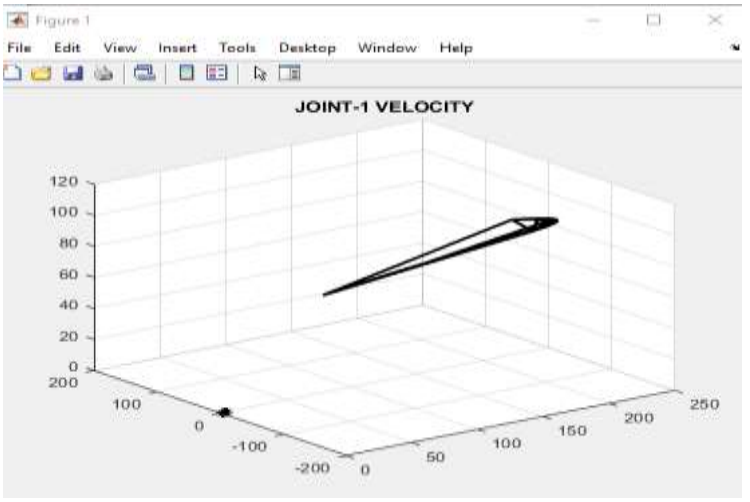
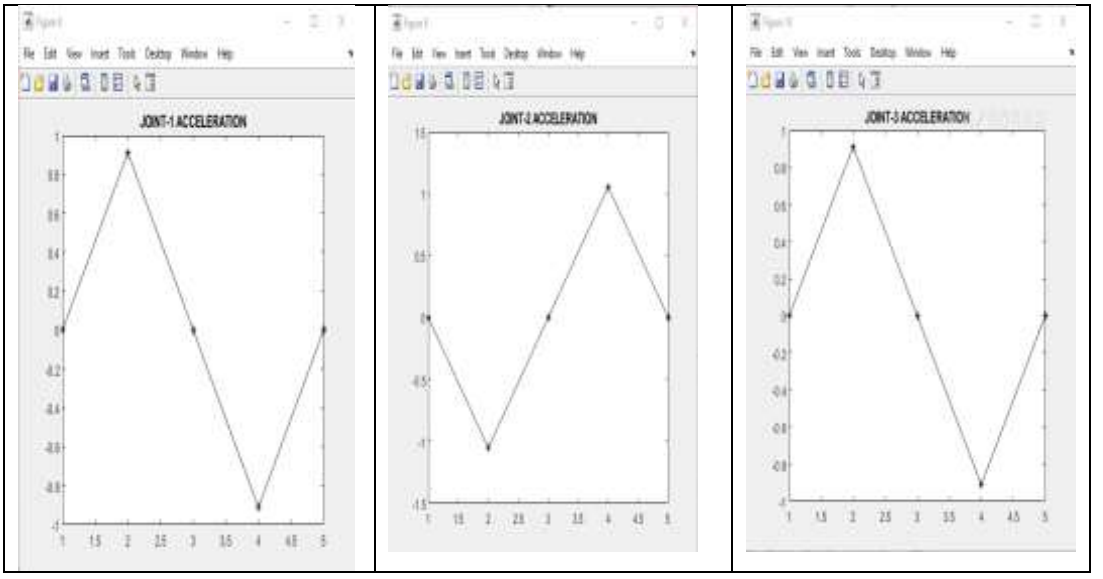


Figure 8 Joint Velocity

motion of Robotic system change where the rate angles of the joints in a with respect to time. It is a calculate of how quickly or slowly each joint is moving. In the context of robotics, joint velocity is a crucial parameter for controlling the motion and behavior of robotic arms or manipulators. The joint velocity for each joint (denoted as θ) is normally represented as θ̇ (theta dot), and it is given by the first derivative of the joint position with respect to time. Mathematically, it can be expressed as:

thetȧ = \frac{d\theta}{dt}

Here, θ̇ represents the joint velocity, dθ is the change in joint position, and dt is the change in time.Joint velocity plays a significant role in robot kinematics and dynamics. It affects the speed and smoothness of the robot's movements, and controlling joint velocities is essential for achieving accurate and efficient robotic tasks.



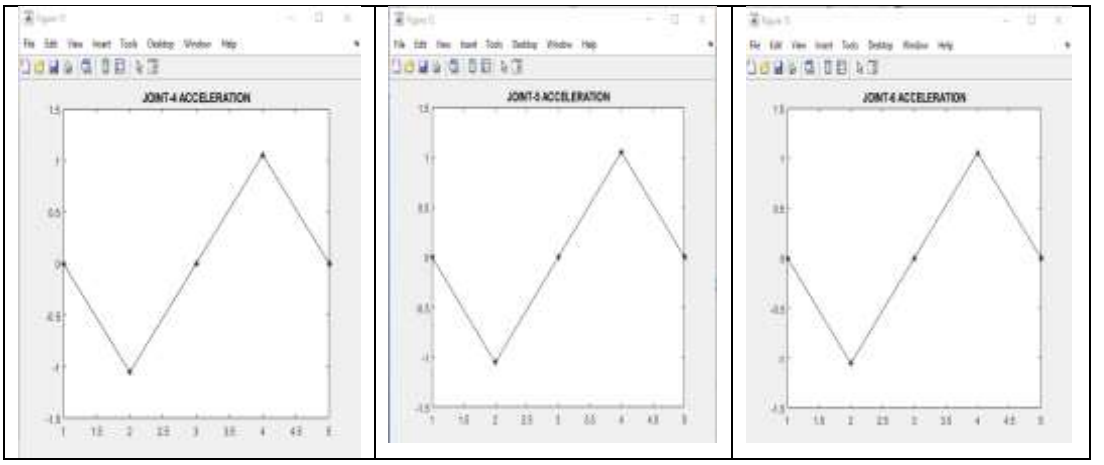


Figure 9 joint acceleration

Figure 9 showing the joint acceleration for a robotic system along the Y-axis, with specific ranges for the X-axis (1 to 5), we'll generate plots for each of the six joints. Joint acceleration refers to the rate of change of joint velocity with respect to time. In the context of proposed robotics, it is a crucial parameter to ensure smooth and controlled motion of the robot. The positive and negative signs of acceleration indicate whether the joint is accelerating or decelerating. In the plotted graphs, the X-axis represents the given range from 1 to 5, depicting different positions or time intervals.

The Y-axis, on the other hand, shows the joint acceleration values for each joint. For Joint 1, the acceleration varies along the Y-axis, showcasing how the first joint responds to changes in the specified time range. equally, Joints 2 to 6 exhibit their acceleration profiles concerning the given X-axis range.

Joint acceleration is a key factor manipulate the robot's movement and response to commands. A positive acceleration indicates an increase in joint velocity, while negative acceleration signifies a decrease. Observing the plotted graphs allows us to analyze the dynamic behavior of each joint during the specified time intervals.

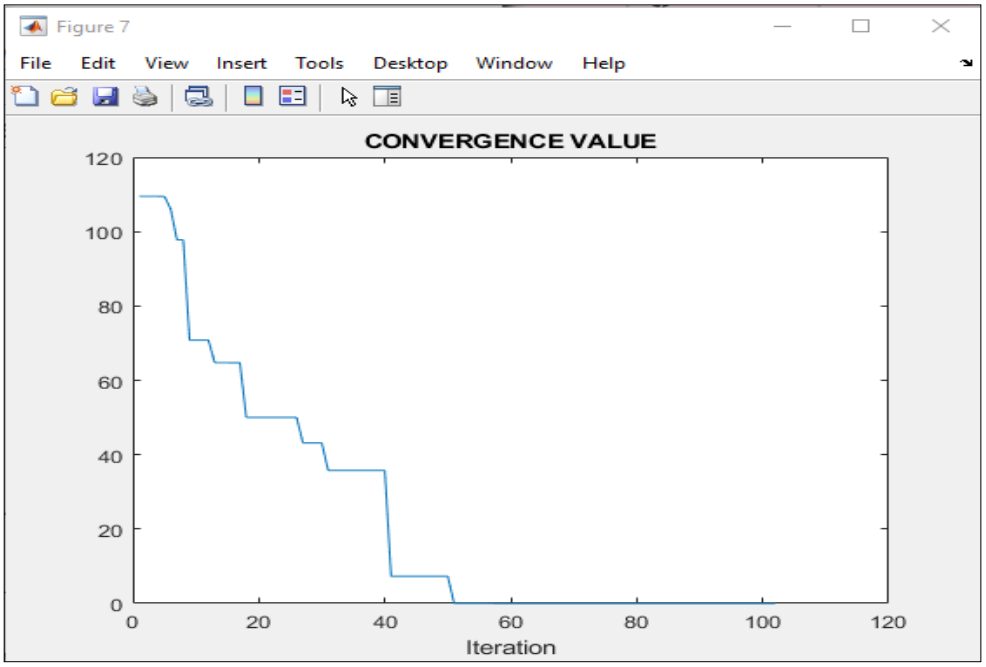


figure 10 convergence value

Table 2 Best Fitness Appear Earliest Generation [36]

Optimization Algorithm	Best Fitness Appear Earliest Generation
PSO	380
FOA	350
HHO	283
IHHO	190
MOPSO	47.2419

The optimization algorithm performance is summarized in the table 2, showcasing the best fitness values achieved by each algorithm and the corresponding generation at which these optimal fitness values first appear. In this comparison, the Particle Swarm Optimization (PSO) algorithm demonstrates a fitness value of 380, achieving its best result in the 380th generation. The Firefly Optimization Algorithm (FOA) follows closely with a best fitness value of 350, observed in the 350th generation. The Hunting Hornet Optimization (HHO) algorithm achieves a best fitness value of 283, appearing in an earlier generation than both PSO and FOA. The Improved Hunting Hornet Optimization (IHHO) algorithm surpasses the others with a fitness value of 190, occurring in the 190th generation. Lastly, the Multi-Objective Particle Swarm Optimization (MOPSO) algorithm excels with a significantly lower best fitness value of 47.2419, demonstrating its efficiency in optimization tasks

Table 3 displays the angle values of the six joints of a manipulator calculated by different algorithms. Each row corresponds to a specific algorithm, and each column represents the joint angle for a particular degree of freedom (DOF). Here's a breakdown of the table:

Table 3 angle values of the six joints [37]

Algorithm	1-DOF	2-DOF	3-DOF	4-DOF	5-DOF	6-DOF
Standard value	−1.047198	−1.047198	1.047198	−0.785398	0.785398	0.523599
PSO	−0.49164228	−0.9547942	0.003137195	−1.42360713	1.275797884	1.306460089
PSO_Adaptation	−1.1199608	−1.58311273	0.63145572	−0.16697007	1.031499907	−0.33419752
PSO_Breed	−1.08357914	−1.31515517	0.839326647	−0.47618412	1.132207126	0.185101802
PSO_Lamda	0.557517444	−0.12569131	−0.71358997	1.027890172	−2.00106961	2.265754194
PSO_Lin	−1.71189772	−1.31515511	0.839326614	−0.47618413	0.673626419	0.643682534
PSO_Nature	−2.54977472	0	0	0	0	0
MOFOPSO	−1.04719681	−1.04719668	1.047197423	−0.78539912	0.785401078	0.523591577
MOPSO (Proposed)	0	0	0	0	0	0.7101

The table 3 illustrates the angular values of the six joints for different algorithms applied to a robotic system with varying degrees of freedom (DOF). The standard values represent the ideal joint angles for optimal performance. The presented algorithms, including PSO, PSO_Adaptation, PSO_Breed, PSO_Lamda, PSO_Lin, PSO_Nature, MOFOPSO, and the proposed MOPSO, showcase their respective joint angle calculations. Comparing these results, it is observed that the proposed MOPSO algorithm achieves a unique value of 0.7101 for the 6-DOF, indicating its distinctive approach to optimizing the joint angles. While the PSO_Nature algorithm produces null values for all DOFs, suggesting a lack of optimization in this specific scenario. The angular values obtained by the MOPSO algorithm demonstrate its efficacy in determining joint angles that align closely with the standard values, signifying accurate and optimized motion planning for the robotic system. Further analysis and evaluation of these algorithms can provide insights into their specific strengths and weaknesses, aiding in the selection of the most suitable algorithm for particular robotic

V CONCLSUION

a time-optimal trajectory planning approach for a 6-DOF robot utilizing a MOPSO algorithm. The objective was to optimize the robot's trajectory with considerations for multiple objectives, such as joint acceleration. The MOPSO algorithm was employed to efficiently explore the trade-offs among these conflicting objectives. By incorporating multi-objective optimization, the algorithm has successfully generated trajectories that not only minimize the travel time but also consider energy efficiency and joint acceleration. This holistic optimization contributes to achieving a well-balanced and efficient motion profile for the robot. The comparative analysis with other optimization techniques underscores the advantages of the MOPSO algorithm in achieving superior trajectory planning outcomes. The ability to handle multiple objectives simultaneously provides a versatile and robust solution for optimizing complex robotic motions. The proposed method takes into account the dynamic nature of the robot and the constraints associated with time-optimal trajectory planning. The consideration of joint acceleration contributes to the overall stability and efficiency of the robot's movements. In future work, further enhancements and refinements can be explored to adapt the algorithm to specific robotic platforms, environmental conditions, or task requirements.the research provides valuable insights into the potential of multi-objective optimization techniques for enhancing the performance of 6-DOF robots in trajectory planning scenario.

References

1. Agustí-Juan I, Müller F, Hack N, Wangler T, Habert G (2017) Potential benefits of digital fabrication for complex structures: environmental assessment of a robotically fabricated concrete wall. J Clean Prod 154:330
2. Krieg OD, Schwinn T, Menges A, Li JM, Knippers J, Schmitt A, Schwieger V (2015) Biomimetic lightweight timber plate shells: computational integration of robotic fabrication, architectural geometry and structural design. Advances in architectural geometry 2014. Springer, Cham, pp 109–125
3. Suqin He; Chuxiong Hu; Shize Lin; Yu Zhu (2022) An Online Time-Optimal Trajectory Planning Method for Constrained Multi-Axis Trajectory With Guaranteed Feasibility IEEE Robotics and Automation Letters, Volume: 7, Issue: 3,Journal Article DOI: 10.1109/LRA.2022.3183536
4. Wenjie Wang; Qing Tao; Yuting Cao; Xiaohua Wang; Xu(2020) Zhang Robot Time-Optimal Trajectory Planning Based on Improved Cuckoo Search Algorithm IEEE Access Volume: 8 ,DOI: 10.1109/ACCESS.2020.2992640
5. Yalun Wen; Prabhakar Pagilla(2023) Path-Constrained and Collision-Free Optimal Trajectory Planning for Robot Manipulators IEEE Transactions on Automation Science and Engineering ,Volume: 20, Issue: 2 ,DOI: 10.1109/TASE.2022.3169989
6. Lizhen Xia (2023) Trajectory Planning Application of Rehabilitation Robots Based on Improved CSA Algorithm IEEE Access,Volume:11:IEEE DOI: 10.1109/ACCESS.2023.334 734
7. Xiao Hu; Heng Wu; Qianlai Sun; Jun Liu (2023) Robot Time Optimal Trajectory Planning Based on Improved Simplified Particle Swarm Optimization Algorithm IEEE Access,Volume: 11 ,Journal Article ,DOI: 10.1109/ACCESS.2023.3272835
8. Peiyao Shen; Xuebo Zhang; Yongchun Fang; Mingxing Yuan (2020) Real-Time Acceleration-Continuous Path-Constrained Trajectory Planning With Built-In Tradeoff Between Cruise and Time-Optimal Motions IEEE Transactions on Automation Science and Engineering ,Volume: 17, Issue: 4,Journal Article,DOI: 10.1109/TASE.2020.298042
9. Alessandro Palleschi; Riccardo Mengacci; Franco Angelini; Danilo Caporale; Lucia Pallottino; Alessandro De Luca; Manolo Garabini (2020)Time-Optimal Trajectory Planning for Flexible Joint

Robots IEEE Robotics and Automation Letters ,2020 Volume: 5, Issue: 2 ,DOI: 10.1109/LRA.2020.2965861

10. Juncheng Li; Maopeng Ran; Lihua Xie (2021) Efficient Trajectory Planning for Multiple Non-Holonomic Mobile Robots via Prioritized Trajectory Optimization IEEE Robotics and Automation Letters,2021,Volume6, Issue: 2,IEEE DOI: 10.1109/LRA.2020.3044834
11. Chen Zhang; Yibin Li; Lelai Zhou (2022) Optimal Path and Timetable Planning Method for Multi-Robot Optimal Trajectory IEEE Robotics and Automation Letters Year: 2022 | Volume: 7, Issue: 3 | Journal Article | Publisher: IEEE DOI: 10.1109/LRA.2022.3187529
12. Eric Barnett; Clément Gosselin A Bisection Algorithm for Time-Optimal Trajectory Planning Along Fully Specified Paths IEEE Transactions on Robotics Year: 2021 | Volume: 37, Issue: |Journal Article | Publisher: IEEE DOI: 10.1109/TRO.2020.3010632
13. Wenzheng Chi; Chaoqun Wang; Jiankun Wang; Max Q.-H. Meng (2019) Risk-DTRRT-Based Optimal Motion Planning Algorithm for Mobile Robots IEEE Transactions on Automation Science and Engineering Year: 2019 | Volume: 16, Issue: 3 | Journal Article | Publisher: IEEE DOI: 10.1109/TASE.2018.2877963
14. Gil Manor; Joseph Z. Ben-Asher; Elon Rimon (2018) " Time Optimal Trajectories for a Mobile Robot Under Explicit Acceleration Constraints" IEEE Transactions on Aerospace and Electronic Systems Year: 2018 | Volume: 54, Issue: 5 | Journal Article | Publisher: IEEE DOI: 10.1109/TAES.2018.2811158
15. Hean Hua; Yongchun Fang; Xuetao Zhang; Chen Qian A Time-Optimal Trajectory Planning Strategy for an Aircraft With a Suspended Payload via Optimization and Learning Approaches IEEE Transactions on Control Systems Technology Year: 2022 | Volume:30,Issue: 6 |Journal Article | Publisher: IEEE DOI: 10.1109/TCST.2021.3139762
16. Yi Liu; Chen Guo; Yongpeng Weng Online Time-Optimal Trajectory Planning for Robotic Manipulators Using Adaptive Elite Genetic Algorithm With Singularity Avoidance IEEE Access Year: 2019 | Volume: 7 | Journal Article | Publisher: IEEE
17. Vincenzo Petrone; Enrico Ferrentino; Pasquale Chiacchio Time-Optimal Trajectory Planning With Interaction With the Environment IEEE Robotics and Automation Letters Year: 2022 | Volume: 7, Issue: 4 | Journal Article | Publisher: IEEE DOI: 10.1109/LRA.2022.3191813
18. Shreyas Kousik; Bohao Zhang; Pengcheng Zhao; Ram Vasudevan Safe, Optimal, Real-Time Trajectory Planning With a Parallel Constrained Bernstein Algorithm IEEE Transactions on Robotics Year: 2021 | Volume: 37, Issue: 3 | Journal Article | Publisher: IEEE DOI: 10.1109/TRO.2020.3036617
19. Peiyao Shen; Xuebo Zhang; Yongchun Fang Tree-Search-Based Any-Time Time-Optimal Path-Constrained Trajectory Planning With Inadmissible Island Constraints IEEE Access Year: 2019 | Volume: 7 | Journal Article | Publisher: IEEE DOI: 10.1109/ACCESS.2018.2886233
20. Gasparetto A, Boscariol P, Lanzutti A, Vidoni R (2015) Path planning and trajectory planning algorithms: A general overview. Motion and operation planning of robotic systems. Springer, pp 3–27
21. Kyriakopoulos KJ, Saridis GN (1988) Minimum jerk path generation. In: Proceedings. 1988 IEEE international conference on robotics and automation. IEEE, pp 364–369
22. Gasparetto A, Zanotto V (2007) A new method for smooth trajectory planning of robot manipulators. Mech Mach Theory 42(4):455
23. Sato A, Sato O, Takahashi N, Kono M (2007) Trajectory for saving energy of a direct-drive manipulator in throwing motion. Artif Life Robot 11(1):61
24. Luo LP, Yuan C, Yan RJ, Yuan Q, Wu J, Shin KS, Han CS (2015) Trajectory planning for energy minimization of industry robotic manipulators using the Lagrange interpolation method. Int J Precis Eng Manuf 16(5):911

25. Balkan T (1998) A dynamic programming approach to optimal control of robotic manipulators. Mech Res Commun 25(2):225
26. Shiller Z (1996) Time-energy optimal control of articulated systems with geometric path constraints. J Dyn Syst Meas Control 118(1):139. https://doi.org/10.1115/1.2801134
27. Gasparetto A, Zanotto V (2008) A technique for time-jerk optimal planning of robot trajectories. Robot Comput Integr Manuf 24(3):415
28. Zanotto V, Gasparetto A, Lanzutti A, Boscariol P, Vidoni R (2011) Experimental validation of minimum time-jerk algorithms for industrial robots. J Intell Robot Syst 64(2):197
29. Ata AA, Myo TR (2005) Optimal point-to-point trajectory tracking of redundant manipulators using generalized pattern search. Int J Adv Robot Syst 2(3):24
30. Saravanan R, Ramabalan S, Balamurugan C (2008) Evolutionary optimal trajectory planning for industrial robot with payload constraints. Int J Adv Manuf Technol 38(11–12):1213
31. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: MHS'95. Proceedings of the sixth international symposium on micro machine and human science. IEEE, pp 39–43
32. Zhao Q, Yan S (2005) Collision-free path planning for mobile robots using chaotic particle swarm optimization. In: Wang L, Chen K, Ong YS (eds) Advances in natural computation. ICNC 2005. Lecture notes in computer science, vol 3612. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11539902_77
33. Zhang Y, Gong DW, Zhang JH (2013) Robot path planning in uncertain environment using multi-objective particle swarm optimization. Neurocomputing 103:172
34. Davis L (1991) Handbook of genetic algorithms
35. Liao X, Wang W, Lin Y, Gong C (2010) Time-optimal trajectory planning for a 6R jointed welding robot using adaptive genetic algorithms. In: 2010 International conference on computer, mechatronics, control and electronic engineering, vol 2. IEEE, pp 600–603
36. Sha Luo† Dianming Chu Qingdang Li Yan He(2022) Inverse Kinematics Solution of 6-DOF Manipulator Based on Multi-Objective Full-Parameter Optimization PSO Algorithm Volume 16 - 2022 | https://doi.org/10.3389/fnbot.2022.791796
37. Jing Xu Chaofan Ren and Xiaonan Chang (2023) Robot Time-Optimal Trajectory Planning Based on Quintic Polynomial Interpolation and Improved Harris Hawks Algorithm Jing Xu Chaofan Ren and Xiaonan Chang , Axioms 2023, 12, 245. https://doi.org/10.3390/axioms12030245