Robotica (2024), **42**, pp. 3904–3917
doi:10.1017/S0263574724001711

RESEARCH ARTICLE

A high-precision trajectory capture and playback solver for KUKA iiwa robot

Zhuoran Liu[1], Jiahang Yang[1], Ashraf Fahmy[2] and Chunxu Li[2]

[1]College of Mechanical and Electrical Engineering, Hohai University, Changzhou, China and [2]Department of Mechanical Engineering, Faculty of Science and Engineering, Swansea University, Swansea, UK
Corresponding author: Chunxu Li; Email: chunxu.li@swansea.ac.uk

Received: 15 June 2024; **Revised:** 19 August 2024; **Accepted:** 12 September 2024; **First published online:** 10 December 2024

Keywords: Collaborative robots; KUKA iiwa; trajectory capture; trajectory playback; Ju-Gibbs quaternion; sliding filter; ROS

Abstract
This paper introduces a sophisticated trajectory capture and playback mechanism for collaborative robots, aimed at enhancing accuracy and operational efficiency through several innovative techniques. The Ju-Gibbs attitude quaternion is utilized for enhanced kinematic modeling across multi-axis systems, which simplifies variables, reduces dimensions, and enhances symbolic clarity, thus surpassing the limitations of traditional rotation vectors and unit quaternions. A new sliding filter is developed to effectively reduce noise and optimize trajectory details more efficiently. Additionally, an automated mechanism is implemented for adjusting the sampling rate and removing static data points at the trajectory's start and end, further refining data collection accuracy. These advancements have been successfully replicated on the Kuka robot LBR iiwa 7 R800, demonstrating the practical applicability of the solutions in real-world settings.

1. Introduction
Collaborative robots, or cobots, are increasingly prevalent in industrial and research environments due to their ability to work alongside human operators and their adaptability to various tasks. The precision and accuracy of these robots are critical, particularly in applications requiring intricate movements and precise positioning [1]. One of the key challenges in the field of collaborative robotics is the accurate capture and playback of robot trajectories. This ability is essential for tasks such as repetitive assembly, quality inspection, and precise manipulation in dynamic environments [2].

This paper presents a novel high-precision trajectory capture and playback mechanism specifically designed for the KUKA iiwa collaborative robot [3]. The mechanism leverages advanced data processing techniques and Ju-Gibbs attitude quaternion to ensure the fidelity and accuracy of captured trajectories, addressing common challenges such as noise and variable movement speeds [4, 5].

Liu [6] developed a high-precision trajectory replication method using advanced adaptive spline interpolation, addressing the challenges in robotic trajectory replication from the aspects of spatial accuracy, temporal fidelity, and dynamic response optimization. Patel [7] proposed a novel trajectory correction algorithm based on real-time feedback and predictive control for enhanced trajectory accuracy in complex environments. Zhang [8] utilized a dual-objective optimization approach incorporating machine learning techniques to refine the trajectory by minimizing deviations and maximizing efficiency. Kalman [9] pioneered the classic Kalman filter, providing a robust statistical method that optimally estimates the states of linear dynamic systems from a series of incomplete and noisy measurements. However, traditional trajectory replication methods in robotics may face difficulties in achieving such high precision due to their dependence on simpler interpolation techniques, which lack the capability to effectively manage complex kinematic behaviors and environmental interactions [10].

The main improvements made in this study are:

1. Developing a new sliding filter to reduce noise, maintaining path integrity, and ensuring accurate playback, which optimizes trajectory details more effectively.
2. Utilizing the Ju-Gibbs attitude quaternion to optimize kinematic modeling for multi-axis systems (MAS). This method simplifies variables, reduces dimensions, and enhances symbolic clarity, overcoming the limitations of traditional rotation vectors and unit quaternions.

The proposed mechanism has been thoroughly tested and validated through a series of experiments. The results demonstrate a significant improvement in the precision of trajectory capture and playback, confirming the efficacy of the approach. This system provides a robust solution for real-time trajectory acquisition and reproduction, paving the way for more advanced and precise applications of collaborative robots in various fields.

The remainder of this paper is organized as follows. Section 2 shows the quaternion and kinematic analysis. Section 3 details the methodology includes data processing, filtering, storage, and trajectory playback. Section 4 presents the experimental procedures used to assess the proposed system, including the setup and execution of tests. Section 5 summarizes the research findings and outlines future directions for further study and potential improvements.

2. Methodology

The research began with the identification of a critical issue: the KUKA iiwa robot exhibited noticeable errors in trajectory reproduction, which posed challenges in achieving the desired precision and computational efficiency. The primary goal was to develop a solver capable of high-precision trajectory capture and playback while simultaneously improving computational efficiency.

To address this problem, new technologies were developed. The research leveraged the advantages of the Ju-Gibbs quaternion, primarily utilizing it for kinematic modeling while employing Denavit–Hartenberg (D–H) parameters and homogeneous transformation matrices as secondary aids. This approach aimed to enhance computational efficiency. Additionally, a novel filter was developed to replace the simple sliding average filter, effectively reducing noise and enabling higher precision in trajectory reproduction.

The newly developed techniques were then tested through experiments that involved running a sinusoidal trajectory. The primary objective was to evaluate the noise reduction capabilities of the new filter. The results were captured in three sets of comparative experimental figures. In the MoveIt visual interface, the end-effector's movement was traced by a blue line, visibly representing the trajectory. The experiments successfully replicated the trajectory in both the Gazebo simulation environment and on the physical KUKA iiwa robot.

Further experiments involved running an arc trajectory, which provided data for analyzing joint angles and conducting an Iinverse kinematic error analysis. The results confirmed that computations using Ju-Gibbs quaternions were faster, demonstrating the efficiency and effectiveness of the newly developed methods.

2.1. *Kinematic analysis*

The Kuka LBR iiwa 7 R800 is a seven degrees-of-freedom (DOF) manipulator. Hence, the robot manipulator is kinematically redundant, with a least one redundant DOF, with respect to the task dimension. Table I gives the modified D–H parameters [20] provided by the robot manufacturer.

Although D–H parameters and homogeneous transformation matrices are widely accepted and utilized in many robotics applications, they can introduce computational complexity and inefficiencies when dealing with systems that have complex joint configurations and multiple DOF [32]. Considering these challenges, the quaternion method has been opted to optimize the kinematic model, with D–H parameters playing a complementary role.

Table I. *LBR iiwa 7 R800 D–H parameters.*

i	α_i(degrees)	θ_i(rad)	$d_{i(mm)}$	$r_{i(mm)}$
1	0	θ_1	0	360
2	– 90	θ_2	0	0
3	90	θ_3	0	420
4	90	θ_4	0	0
5	– 90	θ_5	0	400
6	– 90	θ_6	0	0
7	90	θ_7	0	126

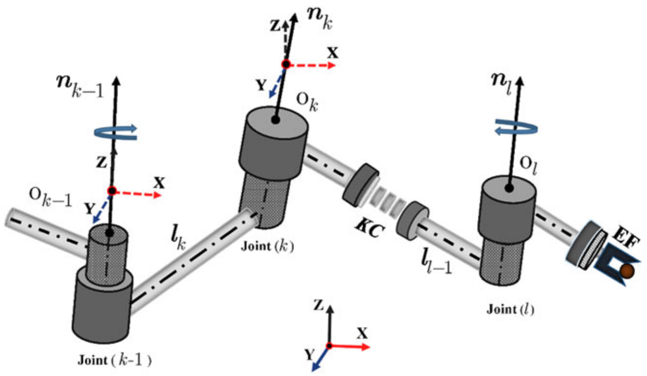


Figure 1. *Robot joints and axis of rotation.*

In the robotic manipulators, each joint's rotation can be elegantly described using quaternions. For a system like the KUKA iiwa, which features multiple articulated joints, the orientation of each segment relative to a fixed base can be computed through the product of quaternion rotations corresponding to each joint [11]. The total orientation of the end-effector relative to the robot base can be calculated by the cumulative multiplication of individual joint quaternions:

$$q_{\text{total}} = q_n \cdot q_{n-1} \cdot \ldots \cdot q_1, \tag{1}$$

where qi is the quaternion representing the rotation imparted by the i-th joint [12].

Consider the transformation at joint k. The quaternion qk rotates the local frame from $Ok-1$ to Ok as illustrated in Fig. 1. The vector lk, representing the arm link, transforms according to:

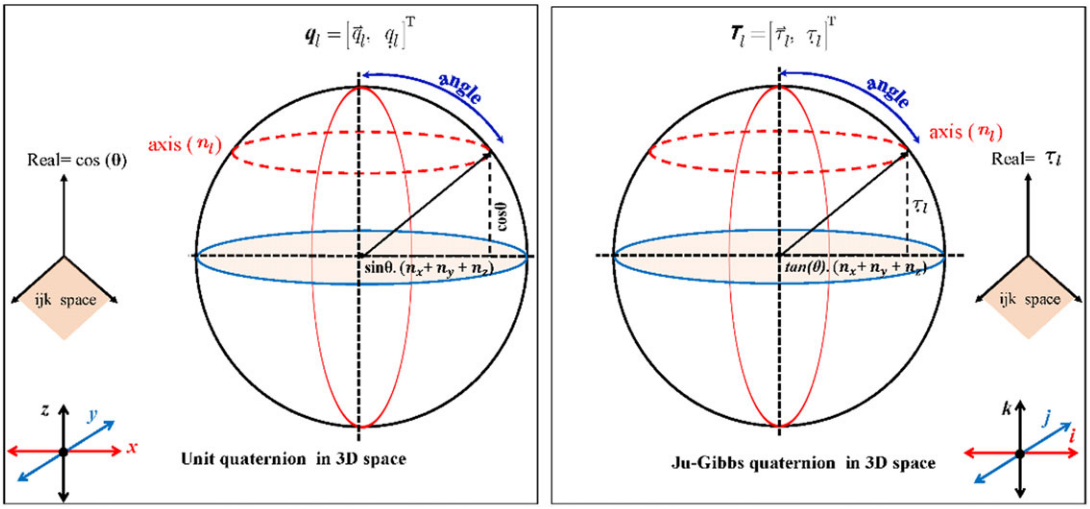
$$v_k = q_k \cdot v_{k-1} \cdot q_k^{-1}, \tag{2}$$

where $v_{\text{k}-1}$ is the vector position of the link in the previous joint's coordinate frame and q_k^{-1} is the conjugate (inverse) of q_k [13, 14].

2.2. Ju-Gibbs quaternion

In this study, an analytical quaternion was utilized to describe the rotation of multi-axis chains, achieving optimal kinematic modeling through the elimination of variables, reduction in dimensionality, and comprehensive symbol analysis. This approach facilitated more precise and simplified modeling of complex robotic movements.

The natural space has 6D; from this perspective, it is clear that the position and orientation equations based on rotation vectors, rotation matrices, and unit quaternion do not meet the minimum requirements for describing kinematic equations robots formula [15, 16]. Therefore, the kinematic equations must



$q_l = [\vec{q}_l, q_l]^T$
angle
axis (n_l)
Real= cos (θ)
cosθ
sinθ. ($n_x + n_y + n_z$)
ijk space
Unit quaternion in 3D space
$\mathcal{T}_l = [\vec{\tau}_l, \tau_l]^T$
Real= τ_l
τ_l
tan(θ). ($n_x + n_y + n_z$)
Ju-Gibbs quaternion in 3D space

Figure 2. *The quaternions rotation in 3D space [21].*

be constructed not to be redundant, for describing the kinematic transformations of MAS. To enhance quaternion features and eliminate some existing cons of a unit quaternion, the tangent quaternion named Ju-Gibbs attitude quaternion is proposed as an analytical quaternion to describe the orientation of bodies in a unified form, which is the isomorphism to unit quaternion established by Hamilton [9] and Jet Propulsion Laboratory [31] for analytical description and kinematic solutions, but not fully homomorphism for numerical calculation; it is defined as

$$\mathcal{T}_l = \left[\mathrm{n}_l \cdot \tan(\theta_l/2)\,,\, \dot{\tau}_l\right]^T = \left[\vec{\tau}_l, \dot{\tau}_l\right]^T \cong \mathrm{q}_l, \tag{3}$$

where $\vec{\tau}_l = \eta_l \cdot \tan(\theta_l/2) = \eta_l \cdot \tau_l$ is the vector part and $\dot{\tau}_l$ is a scalar part.

The multiplication algorithm of such Ju-Gibbs quaternion and the operation product still result in the rotational formula [17]. It is similar to the complex multiplication law [18–20]; rotations of quaternions and in 3D space is shown in Fig. 2. The quaternion can write in universal form as

$${}^{0}\mathcal{T}_l = {}^{0}\mathcal{T}_{l-1} \cdot {}^{l-1}\mathcal{T}_l = {}^{0}\tilde{\mathcal{T}}_{l-1} \cdot {}^{l-1}\mathcal{T}_l, \tag{4}$$

where · denotes the quaternion multiplication.

From Eqs. (3) and (4), the quaternion multiplication can represent the forward rotation operation of the kinematic axis chain as

$${}^{0}\mathcal{T}_l = \left[\prod\nolimits_{l=1}^{k} \left({}^{k-2}\tilde{\mathcal{T}}_{k-1}\right)\right] \cdot {}^{k-1}T_l, \tag{5}$$

where ${}^{k-2}\tilde{\mathcal{T}}_{k-1}$ is 4×4 matrix. Equation (5) showed that the quaternion multiplication could be replaced by its 4×4 matrix calculation called a quaternion concatenation calculation; it is written as

$${}^{l-2}\tilde{\mathcal{T}}_{l-1} = \begin{bmatrix} \tau_{l-1} \cdot 1 + \vec{\tau}_{l-1}^{\times} & \vec{\tau}_{l-1} \\ -\vec{\tau}_{l-1}^{\mathrm{T}} & \tau_{l-1} \end{bmatrix}, \tag{6}$$

Equations (5) and (6) consist of forward and backward in chain ordering $\vec{\tau}_{l-1}, -\vec{\tau}_{l-1}^{\mathrm{T}}$, respectively; the upper left contains $\tau_{l-1} \cdot 1 + \vec{\tau}_{l-1}^{\times}$ it is 3×3 matrix. For MAS, Eq. (6) can be expressed as

$${}^{i}\vec{\mathcal{T}}_j^{\times} \cdot {}^{j}\mathcal{T}_l = \begin{bmatrix} \tau^{i}{}_{j} \cdot 1 + {}^{i}\vec{\tau}_{l-1}^{\times} & {}^{i}\vec{\tau}_j \\ -{}^{i}\vec{\tau}_{l-1}^{\mathrm{T}} & \tau^{i}{}_{j} \end{bmatrix} \cdot \begin{bmatrix} \vec{\tau}_l \\ \tau_l \end{bmatrix}, \tag{7}$$

where i, j>3 kinematic joints.

Equations (4)–(6) are similar to 4D complex multiplication formula in 4D space, which corresponds to homogeneous transformation. The quaternion multiplications for two orthogonal axes can be written as

$$^{0}\mathcal{T}_{l} = {}^{0}\mathcal{T}_{l-1} \cdot {}^{l-1}\mathcal{T}_{l} = \begin{bmatrix} \dot{\tau}_{l} \cdot \vec{\tau}_{l-1} + \dot{\tau_{l-1}} \cdot \vec{\tau}_{l} + \vec{\tau}_{l-1}^{\times} \cdot \vec{\tau}_{l} \\ \dot{\tau_{l-1}} \cdot \dot{\tau}_{l} - \vec{\tau}_{l-1}^{T} \cdot \vec{\tau}_{l} \end{bmatrix} \tag{8}$$

3. Data processing algorithm design

Traditional sliding average filters, which smooth time series by averaging data within a fixed window, frequently encounter difficulties with high-frequency fluctuations. To overcome this issue, the Dynamic Adaptation Sliding Filter (DASF) has been developed. This advanced sliding average filter is specifically designed to enhance the processing of robotic trajectory data.

The moving average filter smooths the collected trajectory data, reducing noise and improving accuracy. It works by averaging a set of data points to smooth out fluctuations. When employing a simple moving average filter in data analysis, the formula used is

$$SMA(t) = \frac{1}{N} \sum\nolimits_{i=t-N+1}^{t} x_i, \tag{9}$$

where $SMA(t)$ represents the moving average at time t, N is the size of the moving window, indicating the number of data points considered, and x_i denotes the data value at time point i [22].

The approach utilizes weights w_i that are adjusted based on the distance of data points from the center of the window. The formula for calculating the weighted average of sensor data points is

$$S_t = \frac{\sum\nolimits_{i=-n}^{n} w_i x_{t+i}}{\sum\nolimits_{i=-n}^{n} w_i}, \tag{10}$$

where S_t represents the output of the DASF at time t, x_t represents the observed value at time t, and n is the half-width of the window.

Weights are calculated using a Gaussian function, emphasizing central data points and reducing the influence of edge points [23]. Weights w_i are assigned using a Gaussian distribution to prioritize central data points:

$$w_i = \exp\left(-\frac{(i-\mu)^2}{2\sigma^2}\right), \tag{11}$$

where μ represents the mean or the center of the window in the Gaussian function and σ denotes the standard deviation in the Gaussian function, effectively emphasizing more relevant data and reducing noise from outliers [24].

The core mechanism of the DASF, highlighted in step 6 of the algorithm, introduces a method to dynamically adjust the window size in order to accommodate data variability:

$$n(t) = \lfloor \alpha \cdot \sigma_t + \beta \rfloor, \tag{12}$$

where σ_t represents the standard deviation of the data within the current window, α is a scaling factor in the formula for dynamically adjusting the window size, $n(t)$, and β acts as an offset in the dynamic window size calculation, enhancing the filter's flexibility.

The weights w_i in the DASF are adjusted not only based on their position within the window but also to swiftly adapt to significant motion by refining the weighting scheme. As described in steps 12 and 13, this modification increases the weights based on the rate of change between consecutive data points; this refined weighting scheme increases the weights in response to the rate of change between consecutive data points, $\dot{x}_t$ and $\dot{x}_{t-1}$:

$$w_i(t) = \frac{w_i}{\max(w)} \cdot \left(1 + \frac{|\dot{x}_t - \dot{x_{t-1}}|}{\gamma}\right), \tag{13}$$

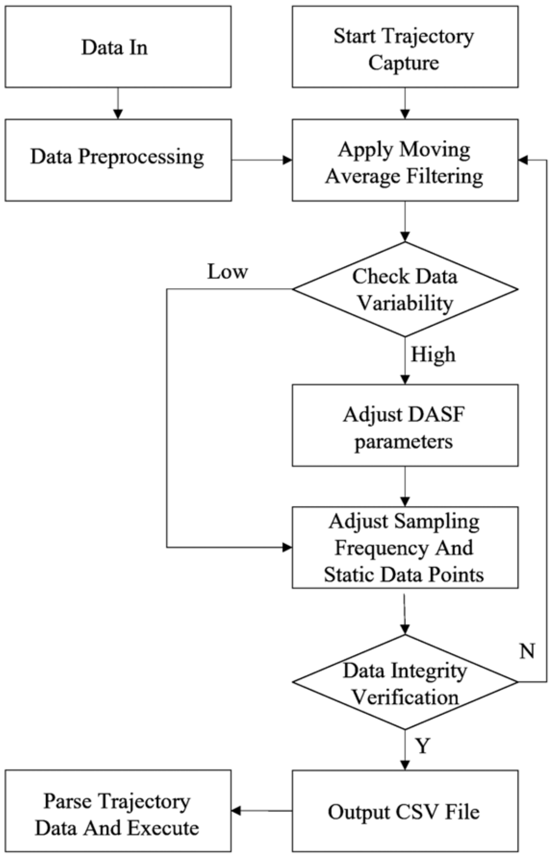

Data In
Start Trajectory Capture
Data Preprocessing
Apply Moving Average Filtering
Check Data Variability
Low
High
Adjust DASF parameters
Adjust Sampling Frequency And Static Data Points
Data Integrity Verification
N
Y
Output CSV File
Parse Trajectory Data And Execute

Figure 3. *Flow chart of end-effector trajectory capture and playback.*

where γ is used in the adjustment of weights, $w_i(t)$, it modulates the sensitivity of the weight adjustments to changes in the rate of data change, represented by the difference in derivatives, $|\dot{x}_t - \dot{x}_{t-1}|$ [25, 26].

Additionally, static data points at the trajectory's start and end are removed, focusing analysis on the most dynamic and relevant segments. Details of the complete data processing method are depicted in Fig. 3.

By integrating DASF, automatic sampling rate adjustment, and removal of static data points, this study achieved a high-precision trajectory capture and playback mechanism for collaborative robots. Further details can be found in Appendix.

4. Experiments

The experiments were conducted using the Robot Operating System (ROS) [8] platform with MoveIt [8] and Gazebo [17] for simulation, as well as the KUKA Sunrise for controlling the real robot. The aim was to validate the effectiveness of the high-precision trajectory capture and playback mechanism. The experimental setup and results are detailed below.

The experimental setup is shown in Fig.4: a Kuka LBR iiwa 7 R800 robot, a laptop running ROS with MoveIt and Gazebo for simulation, and an Ethernet cable for communication between the robot and the laptop.

The control system was divided into two main parts: ROSCORE and SUNRISE. The ROSCORE part included C++ and Python nodes that handled the robot's state and command data. The SUNRISE part included the robotic application and KUKA Sunrise OS [27]. Figure 5 outlines the robotic system setup

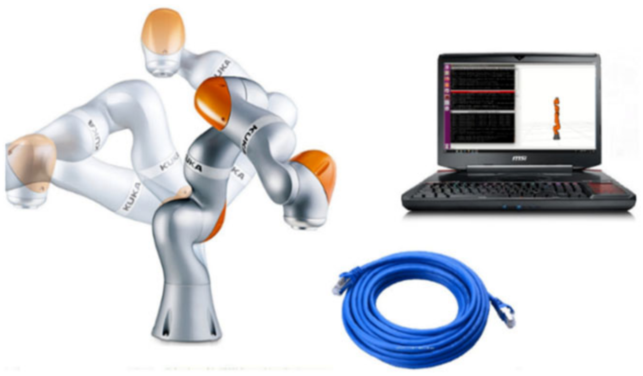


Figure 4. Experimental setup for high-precision trajectory capture and playback [14].

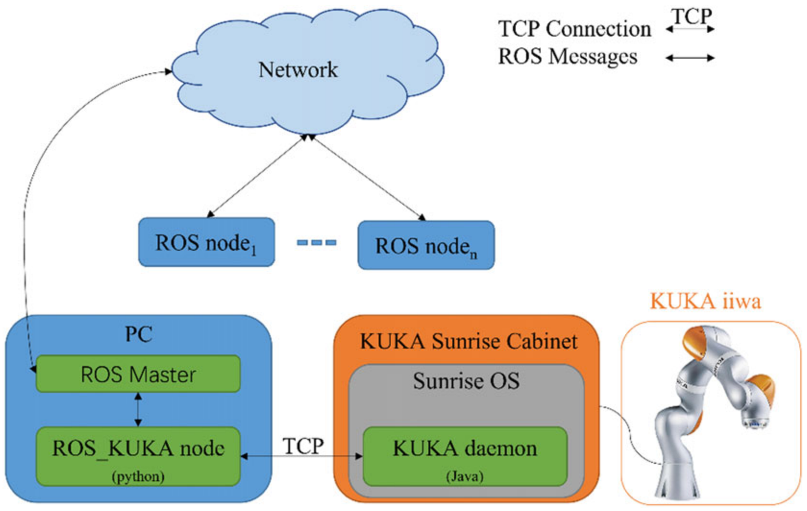


Figure 5. Schematic diagram of system signal transmission.

using ROS to control a KUKA robot. At its core, the setup includes a ROS Master on a PC coordinating the communication between various ROS nodes, which may be spread across different devices within a network. One of these nodes, developed in Python, interfaces directly with the KUKA Sunrise Cabinet that controls the robot through a dedicated Java daemon [28]. This architecture not only enables the robot to perform tasks based on commands sent from the ROS nodes but also ensures smooth and reliable communication over Transmission Control Protocol connections.

4.1. Filtering effect experiments

Two sets of experiments were conducted using a sinusoidal trajectory, starting from the initial position (−348.5686, −2.7714, 697.1189) and ending at the final position (255.3793, 493.7929, 466.1564), as modeled in Fig. 6, to quantitatively assess the impact of data smoothing and adaptive sampling rate adjustments on the quality of trajectory data. The first set was performed without the implementation of a sliding average filter and without automatic adjustments in the sampling rate. This experiment provided a baseline dataset, saved in a CSV file format, capturing the raw sensor outputs as the robot executed predefined tasks. Subsequently, a second experiment incorporated both a sliding average filter and an adaptive sampling rate mechanism.

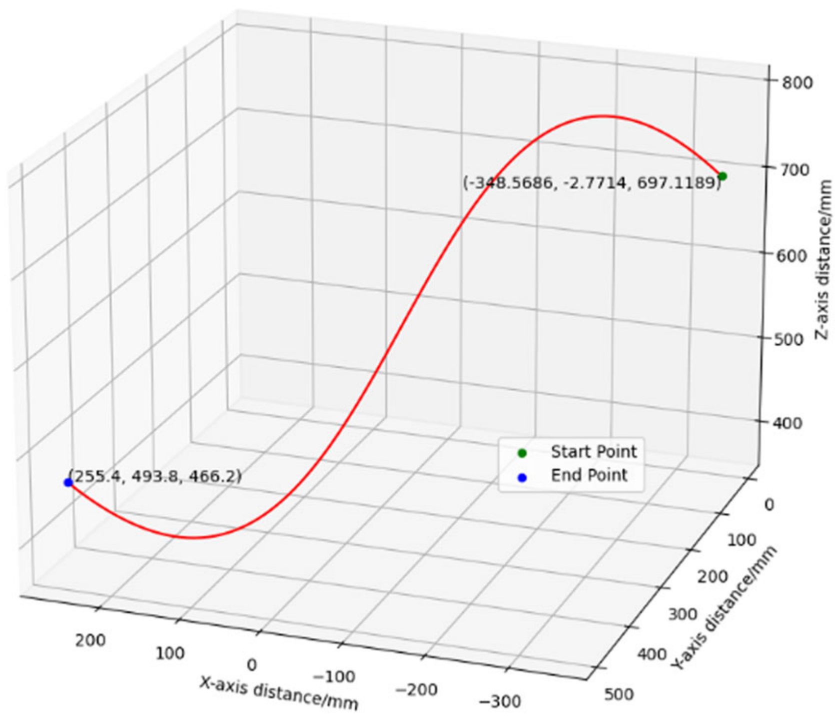


Figure 6. *Sinusoidal trajectory model from initial to final position.*

The experimental data from this experiment was recorded and processed into curve plots for comparative analysis, as shown in Figs. 7–9. The comparative analysis focused on the smoothness and noise levels in the trajectory data from both experiments. It was observed that the introduction of the sliding average filter and adaptive sampling frequency significantly smoothed the data curves. This effect was evident as it effectively reduced random fluctuations and noise, providing a clearer and more consistent representation of the end-effector's trajectory.

It was noted that the errors were primarily concentrated in the early to mid-phases of the motion. In the initial phase of motion, the KUKA iiwa robot's end-effector transitions from rest to active motion, striving to adhere to the predefined trajectory. This transitional period is characterized by significant dynamic changes. The end-effector's inertia and the immediate adjustments required by the control system often result in initial overshooting or oscillatory movements [31]. Such dynamics are prevalent in robotic systems, where initial discrepancies predominantly stem from the control system's delay in effectively responding to rapid changes in the trajectory. This adjustment period can induce pronounced errors at the beginning of the movement, which tend to stabilize as the system reaches a dynamic equilibrium.

In this study, the developed approach was successfully applied to both Gazebo simulations and the physical KUKA iiwa robot, demonstrating the ability to accurately reproduce smooth sinusoidal trajectories. These trajectories are visually represented by blue lines in the MoveIt visualization interface, as shown in Fig. 10. The consistent results across both simulated and real-world environments confirm the effectiveness of the proposed method.

4.2. Forward and inverse kinematics calculations

In this study, the end-effector of the robot completes the spatial trajectory depicted in Fig. 11, starting from the initial position of (−348.5686, −2.7714, 697.1189) and concluding at the final position

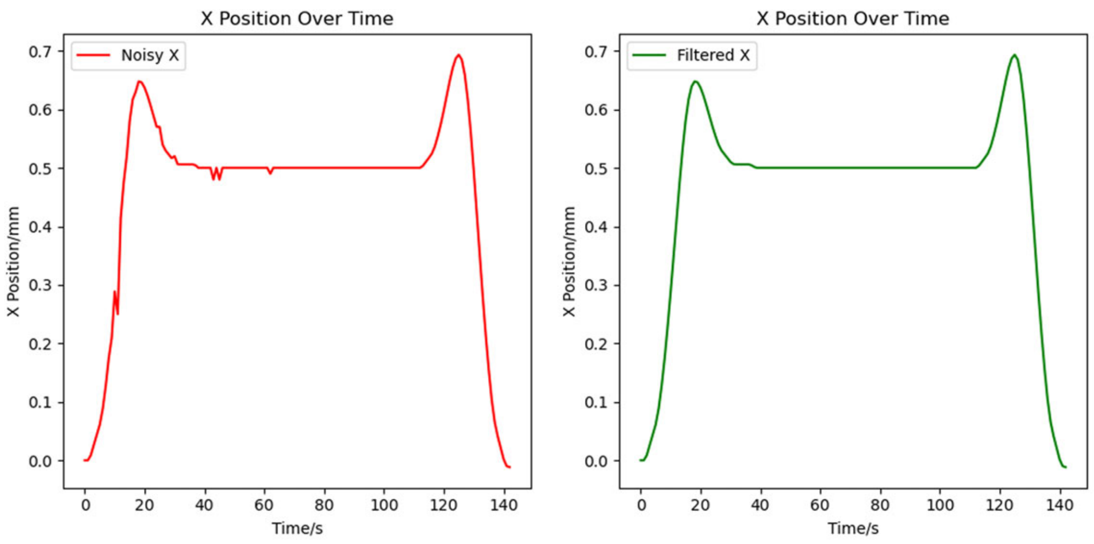


Figure 7. *X position over time (noisy vs. filtered).*

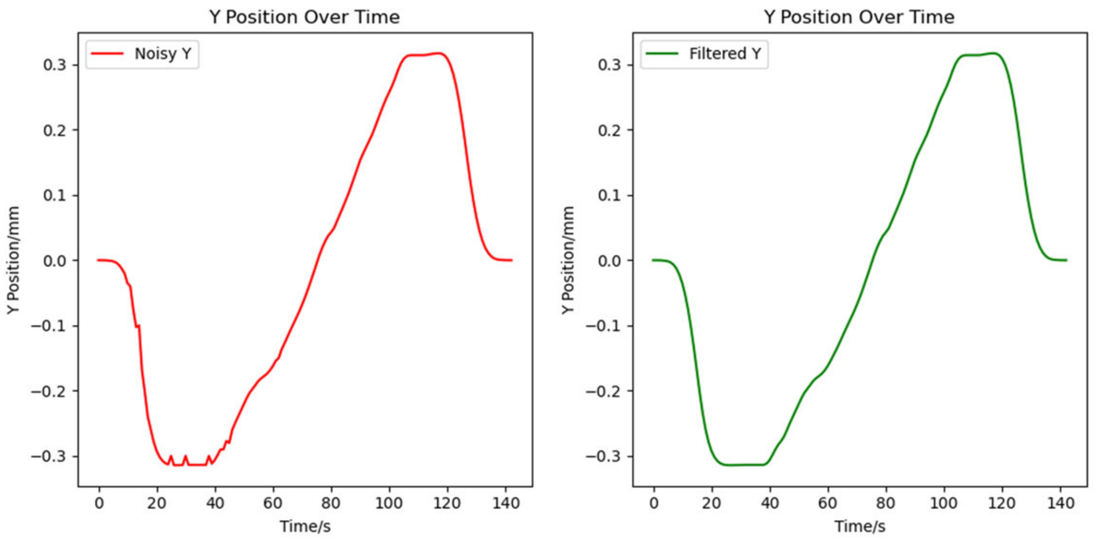


Figure 8. *Y position over time (noisy vs. filtered).*

of (255.3793, 493.7929, 466.1564). The trajectory is divided into 50 segments, with a step duration of 0.05 s per segment, culminating in a total duration of 2.5 s for the robot to complete the arc trajectory.

The inverse kinematics of a seven DOF robotic arm is addressed by decomposing the solution into two main parts within a unified approach. Initially, the first six DOF are treated as a standard six DOF mechanism, ensuring that the end-effector reaches the desired position and orientation. Subsequently, the seventh DOF, typically an additional rotational joint near the end-effector, is utilized to optimize the solution or meet specific criteria. This approach is particularly effective for simple motion trajectories, where the focus is primarily on the angular displacements of the six primary joints.

Since the positions of two fixed points on rotation axis ξ_1 remain unchanged, the second-type subproblems are first used to solve for angles θ_2 and θ_3. Following this, the first-type subproblems are

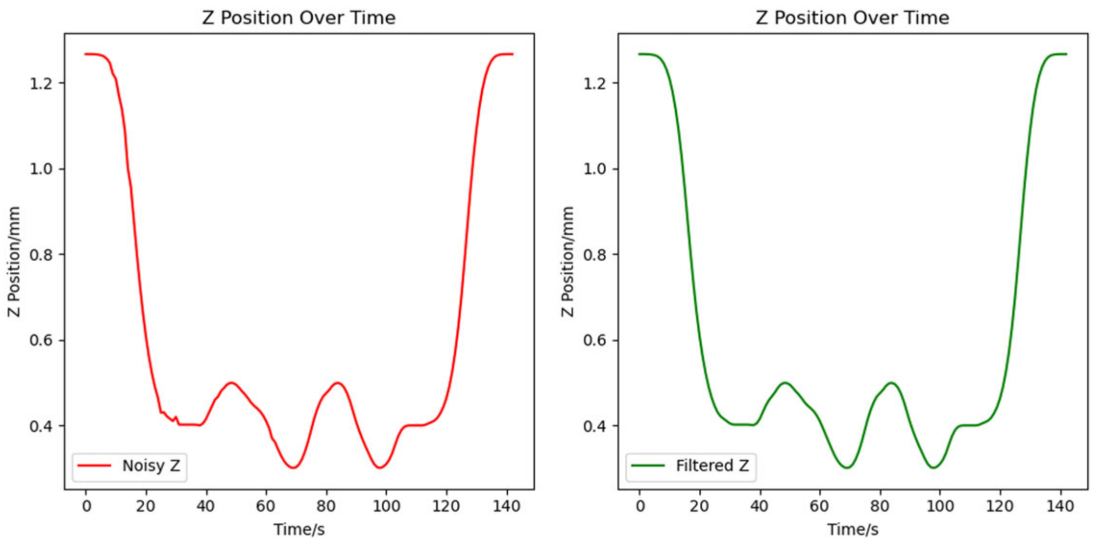


Figure 9. *Z position over time (noisy vs. filtered).*

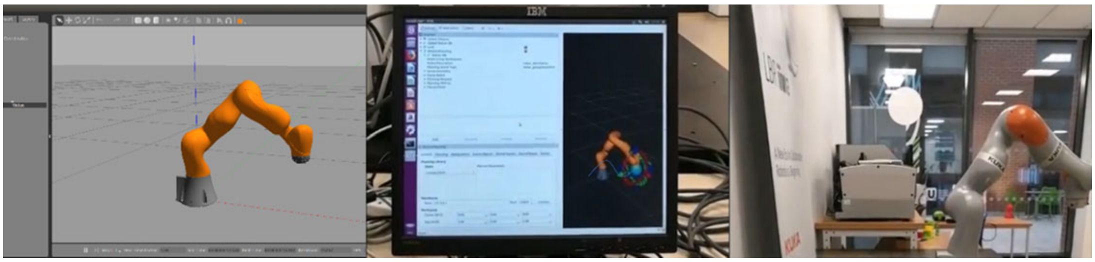


Figure 10. *Trajectory display of end-effector capture and playback.*

employed to determine angle θ_1. After acquiring angles θ_1, θ_2, and θ_3, the initial position of the robot arm's end-effector changes. The process continues with the second-type subproblems to solve for angles θ_4 and θ_5. Once angles θ_1, θ_2, θ_3, θ_4, and θ_5 have been obtained, a point outside the axis is located, and the first-type subproblems are used to solve for angle θ_6. This procedure is repeated to yield a series of 50 corresponding outputs. As shown in Fig. 12, the experimentally obtained angular displacement curves for each joint are smooth, indicating that the inverse solutions derived from the subproblems are continuous.

The computed joint angles from the inverse kinematic process are analyzed using Ju-Gibbs quaternions and the exponential product method. To efficiently compare computation times, calculations for both methods are performed alternately, as illustrated in Fig. 13. This comparison reveals that Ju-Gibbs quaternion computations are more efficient than the exponential product method, offering better real-time performance. Additionally, the Ju-Gibbs quaternions exhibit minimal time fluctuation across different data sets, enhancing the stability of robotic operations.

Finally, the joint angles obtained from the inverse kinematic process are used to compute forward kinematics. The spatial positions derived are then compared with the desired positions. As depicted in Fig. 14, the discrepancies in the end-effector positions determined by the subproblem solutions are within $\pm 5 \times 10^{-2}$ mm, meeting the accuracy requirements for practical robotic applications.

From these experimental results, it is evident that the utilization of the Ju-Gibbs quaternion provides substantial advantages, including the maintenance of precision within an acceptable error margin, which is essential for practical implementations in robotic systems.

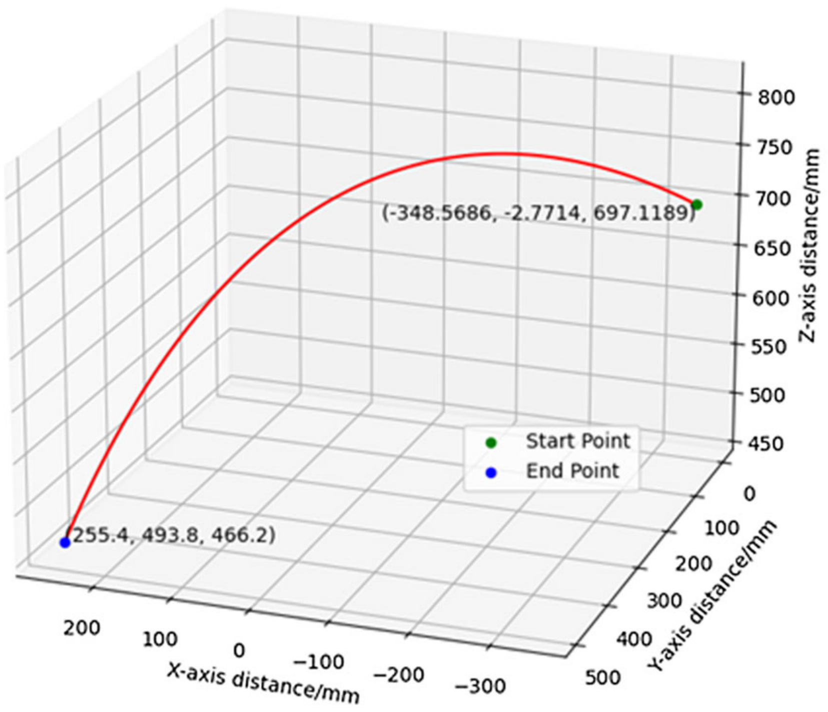


Figure 11. Arc trajectory model from initial to final position.

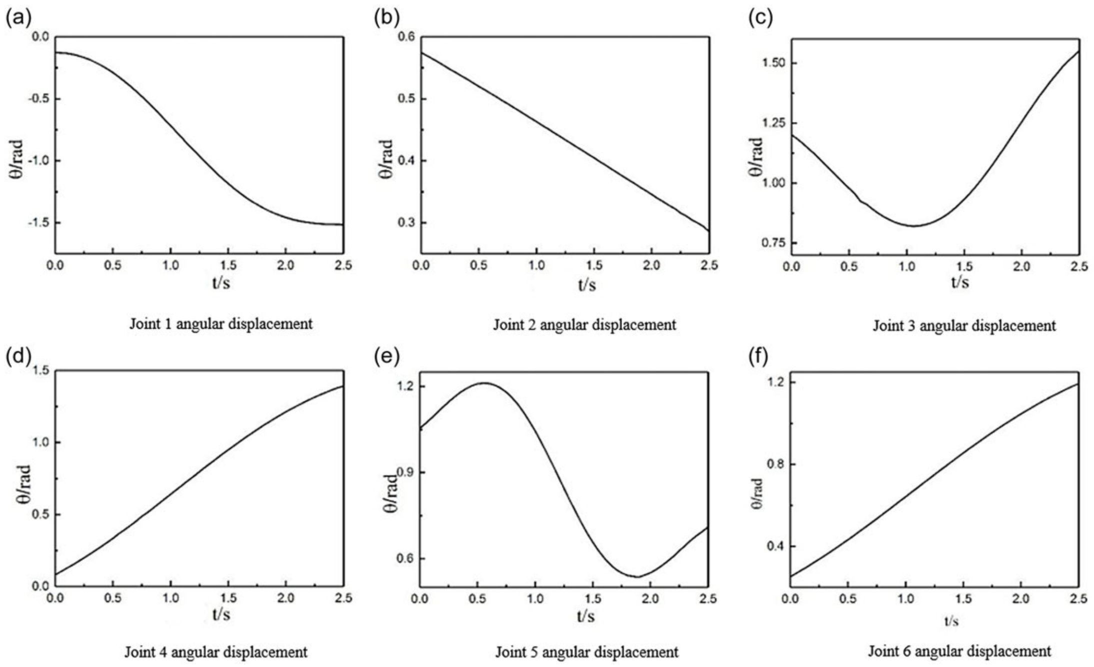


Figure 12. Changes in the angle of the individual joints.

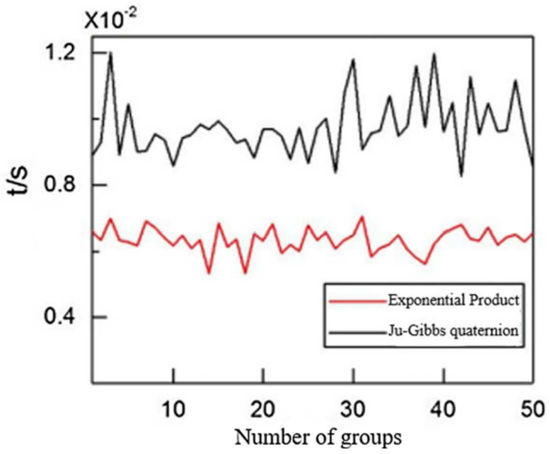


Figure 13. *Comparison of positive kinematics simulation times.*

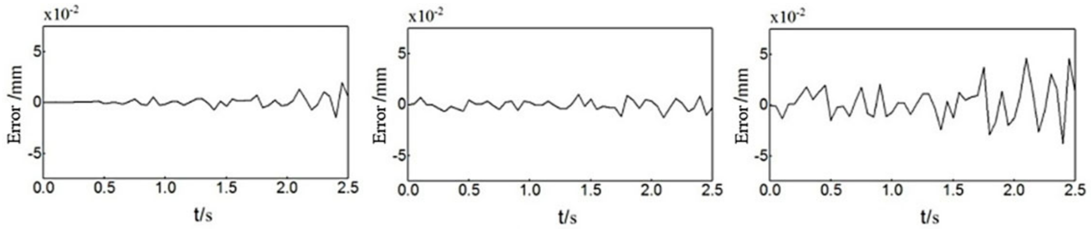


Figure 14. *Inverse kinematic error analysis of X, Y, and Z axes.*

5. Conclusion

The advancements presented in this paper significantly elevate the precision and efficiency of robotic trajectory planning. The new sliding filter ensures more reliable data processing, while the application of the Ju-Gibbs attitude quaternion provides a robust framework for kinematic analysis in complex systems.

This research encompassed two pivotal experiments utilizing the KUKA iiwa robotic platform. The first experiment demonstrated the effectiveness of the DASF across four distinct trajectories, highlighting its superior performance in enhancing trajectory accuracy and noise reduction. The second experiment compared the computational techniques of screw theory exponential product and the Ju-Gibbs quaternion for kinematics calculations. The results distinctly favored the Ju-Gibbs quaternion approach, proving its enhanced efficiency and accuracy in handling complex rotational movements of robotic joints. These findings validate the integration of the DASF and Ju-Gibbs quaternion as substantial advancements in robotic trajectory precision and control.

The experimental results demonstrate the effectiveness of these improvements in achieving high-precision trajectory capture and playback. The end-effector position error analysis, alongside a comparison of position data before and after the application of the DASF, underscores the enhanced accuracy and reliability of the system. The implementation of these advancements has been successfully replicated on the KUKA iiwa robot, demonstrating the practical applicability of these solutions in real-world settings.

In conclusion, the proposed trajectory capture and playback mechanism offers significant improvements in data quality and system performance for collaborative robots. These enhancements render the system a valuable tool for a variety of robotic applications, encompassing fields such as manufacturing, medical robotics, and research [29]. Future work will focus on further refining the algorithms, exploring additional filtering techniques, and extending the system to other robotic platforms to validate its robustness and versatility [30].

Author contributions. Z.R.L.: writing—original draft, writing—review and editing, conceptualization, methodology, software, validation, formal analysis, investigation, data curation, visualization. J.H.Y.: writing—review and editing, validation, conceptualization, methodology. A.F.: conceptualization, validation. C.X.L.: conceptualization, validation, resources, supervision, writing—review and editing.

Financial support. This research received no specific grant from any funding agency, commercial or not-for-profit sectors.

Competing interests. The authors declare no conflicts of interest exist.

Ethical approval. Not applicable.

References

[1] B. Paden, M. Čáp, S. Z. Yong, D. Yershov and E. Frazzoli, “A survey of motion planning and control techniques for self-driving urban vehicles,” *IEEE Trans. Intell. Veh.* **1**(1), 33–55 (2016).
[2] J. Sun, H. Yu, G. Zhong, J. Dong, S. Zhang and H. Yu, “Random shapley forests: Cooperative game-based random forests with consistency,” *IEEE Trans. Cybernetics.* **52**(1), 205–214 (2020).
[3] K. Biswas, I. Kar and E. Feron, “Intent-aware optimal collision avoidance and trajectory planning for a pursuit vehicle,” *Robotica.* **40**(8), 2505–2526 (2022). doi: 10.1017/S0263574721001764.
[4] C. Fu, K. Chen and D. Sun, “Adaptive trajectory planning for autonomous robotic capture of non-cooperative targets in space,” *Acta. Astronaut.* **158**, 244–255 (2019). doi: 10.1016/j.actaastro.2019.01.036.
[5] S. Zhang, A. Li, J. Ren and R. Ren, “Kinematics inverse solution of assembly robot based on improved particle swarm optimization,” *Robotica.* **42**(3), 833–845 (2024). doi: 10.1017/S0263574723001789.
[6] J. Sun, Z. Wang, H. Yu, S. Zhang, J. Dong and P. Gao, “Two-stage deep regression enhanced depth estimation from a single RGB image,” *IEEE Trans. Emerg. Top. Comp.* **10**(2), 719–727 (2020).
[7] J. Sun, G. Zhong, K. Huang and J. Dong, “Banzhaf random forests: Cooperative game theory based random forests with consistency,” *Neural Netw.* **106**, 20–29 (2018).
[8] M. Dong and J. Zhang, “A review of robotic grasp detection technology,” *Robotica.* **41**(12), 3846–3885 (2023). doi: 10.1017/S0263574723001285.
[9] W. Hamilton, Lectures on Quaternions: Containing a Systematic Statement of a New Mathematical Method (1853). (https://l1nq.com/ClrIR) Accessed 30 March 2023.
[10] J. Sun, H. Yu, J. J. Zhang, J. Dong, H. Yu and G. Zhong, “Face image-sketch synthesis via generative adversarial fusion,” *Neural Netw.* **154**, 179–189 (2022).
[11] R. B. Rusu and S. Cousins, “3D is here: Point Cloud Library (PCL),” In: IEEE International Conference on Robotics and Automation (ICRA), Sacramento, California, USA, from April 9–April 11 (2011).
[12] Clifford, “Preliminary sketch of biquaternions,” *Proc. London Math. Soc.* **1**(1), 381–395 (1871).
[13] A. Cohen and M. Shoham, “Hyper dual quaternions representation of rigid bodies kinematics,” *Mech. Mach. Theory.* **150**, 103861 (2020). doi: 10.1016/j.mechmachtheory.2020.103861.
[14] A. Ahmed, H. Ju, Y. Yang and H. Xu, “An improved unit quaternion for attitude alignment and inverse kinematic solution of the robot arm Wrist,” *Machines* **11**(7), 669 (2023). doi: 10.3390/machines11070669.
[15] C. Faria, F. Ferreira, W. Erlhagen, Sérgio Monteiro and E. Bicho, “Position-based kinematics for 7-DoF serial manipulators with global configuration control, joint limit and singularity avoidance,” *Mech. Mach. Theory* **121**, 317–334 (2018).
[16] S. Thrun, W. Burgard and D. Fox. *Probabilistic Robotics* (MIT Press, Cambridge, MA, 2005).
[17] R. Cui, C. Yang and H. Wang, “A review of optimization algorithms for trajectory planning in robotics,” *Robotics* **10**(2), 55 (2021).
[18] X. Liu, L. Wang and L. Jin, “Enhanced trajectory optimization for mobile robots with environmental interaction,” *Actuators* **11**(1), 22 (2022).
[19] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” *Int. J. Robot. Res.* **5**(1), 90–98 (1986).
[20] J. J. Craig. *Introduction to Robotics: Mechanics and Control* (Pearson/Prentice Hall, Upper Saddle River, NJ, 2005).
[21] S. Lee, J. Kim and S. Park, “Development and application of a real-time trajectory planning system for industrial robots,” *Robotics* **10**(3), 99 (2021).
[22] C. Samson and K. Ait-Abderrahim, “Feedback Control of a Nonholonomic Wheeled Cart in Cartesian Space,” Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Shanghai, China, from May 9-May 13 (1991).
[23] R. M. Murray, Z. Li and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation* (CRC Press, Boca Raton, FL, 1994).
[24] Z. Jiang and T. Mei, “Strategy and implementation of smooth trajectory planning for robotic systems,” *Actuators* **11**(2), 50 (2022).

[25] L. E. Kavraki, P. Švestka, J. C. Latombe and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE T. Robot. Autom.* **12**(4), 566–580 (1996).

[26] Z. Liu, Y. Huang, D. Liu, X. Guo, K. Wang and J. Tan, "Trajectory planning of large redundant manipulator considering kinematic constraints and energy efficiency," *Robotica.* **41**(11), 3524–3540 (2023). doi: 10.1017/S0263574723001157.

[27] F. Ardiani, M. Benoussaad and A. Janot, "On the dynamic parameter identification of collaborative manipulators: Application to a KUKA iiwa, **2022**, 468–473 (2022). doi: 10.1109/ICARCV57592.2022.10004294.

[28] T. Xu, Dynamic Identification of the KUKA LBR iiwa Robot With Retrieval of Physical Parameters Using Global Optimization, *IEEE Access* **8**, 108018–108031 (2020).

[29] F. Pomerleau, F. Colas and R. Siegwart, "A review of point cloud registration algorithms for mobile robotics," *Found. Trends® Robot.* **4**(1), 1–104 (2013).

[30] D. Fox, W. Burgard and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.* **4**(1), 23–33 (1997).

[31] N. Trawny and S. Roumeliotis, "Indirect kalman filter for 3D attitude estimation[J], 1–6 (2005). doi: 10.2514/6.2005-6052.

[32] H. Ju, "An axis-invariant based inverse kinematics modeling and solution method for general 7R manipulators," China National Intellectual Property Administration, China Patent, CN109033688B (2020).

Appendix

The specific implementation of DASF is as follows:

Algorithm 1

```
Input: data sequence x, initial window size n, sensitivity parameter γ
Output: filtered sequence S
1: Initialize window_half_width n
2: Initialize weights w to Gaussian distribution based on n
3: Initialize S as empty list
4: for t from n to length(x) - n do
5:     Calculate local standard deviation σ_t over window [t-n, t+n]
6:     Adjust window_half_width dynamically: n = floor(α * σ_t + β)
7:     Recalculate weights w for new window size
8:     Compute weighted average for S_t:
9:         numerator = 0
10:        denominator = 0
11:        for i from -n to n do
12:            weight_adjustment = 1 + abs(derivative(x_t) - derivative(x_{t-1})) / γ
13:            w_i = w_i * weight_adjustment
14:            numerator += w_i * x_{t+i}
15:            denominator += w_i
16:        S_t = numerator / denominator
17:        Append S_t to S
18: end for
19: return S
```