Enhancing Motion Planning for Industrial Robots Using Hybrid Methods and Quaternion Representation

Ali Joodi Aalhasan, Bao Song.

National Engineering Research Center of Numerical Control System, School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan 430073 P.R.China (e-mail:ali.judy@atu.edu.iq, songbao@hust.edu.cn)

Abstract: This research addresses the limitations of relying on single trajectory types, which often resulting jerky and inefficient movements, especially in complex scenarios . This research present an enhanced motion planning methodology for industrial robots that integrates hybrid trajectory methods with quaternion representation . Our approach combines Linear-SLERP, B-Spline, and Bézier curves to achieve smooth, adaptive and efficient path planning suitable for diverse industrial environments. By leveraging the strengths of the trajectory method the hybrid approach ensures continuous and stable robot manipulations. Quaternion representation is utilized to avoid gimbal lock and to provide a robust orientation framework enhancing the motion smoothness of industrial robots . The research implemented proposed method using CoppeliaSim with a 7-DoF Franka Emika Panda robotic arm performing a standard pick-and-place task. The results demonstrate that the proposed hybrid method significantly reduces acceleration and jerk compared to traditional trajectory methods thereby minimizing mechanical stress and enhancing overall motion efficiency . This research offers a novel and effective solution for complex robotic applications ensuring precise and stable operations across various industrial settings.

Keywords: Industrial robots, motion planning, hybrid trajectory, Quaternion representation, linear-SLERP, B-Spline, Bézier .

1. INTRODUCTION

Optimizing motion planning for robotic manipulators is crucial in environments where hazardous and fragile materials are handled. Over the past decades, extensive research has provided various approaches to achieving objectives such as minimizing time, acceleration and jerk, increasing velocity, and enhancing overall manipulation. High advancements in motion planning optimization have explored integrating parametric curves and optimization techniques (Y. Chen et al., 2014; Jiao et al., 2013; Van Loock et al., 2015; Wang HF et al., 2012; Y. Zhang et al., 2013). In (Kuo et al., 2020), a dual-optimization trajectory planning method using B-splines and Bézier curves was proposed for robot manipulators, combining optimal path and motion profile planning. Another study formulated motion planning as convex spline optimization, offering fast and reliable solutions with global optimum guarantees(Xu et al., 2021). Bernstein polynomials have been utilized to approximate solutions in a discretized setting, enabling efficient computation and enforcement of constraints for multivehicle operations (Cichella et al., 2021). (Yi et al., 2022) utilized rational Bézier and NURBS algorithms for path planning in environments with known static obstacles, demonstrating that the rational Bézier algorithm generated shorter paths with reduced planning time. (Galcík & Duchon, 2024), Research on mobile robot navigation within known environments has examined various algorithms for real-time path determination. The study also emphasizes different methods of representing known environments, including basic layouts and simplified or transformed graph structures. For autonomous vehicles, spatio-temporal motion planning using trapezoidal prism corridors and Bézier curves has been developed to generate collision-free trajectories in complex scenarios(Deolasee et al., 2023).

These collective efforts contribute to advancing trajectory planning methodologies promoting synchronized, continuous, and safe robotic manipulations across various applications. Planning manipulator trajectories has traditionally involved synthesizing time-jerk optimal paths using diverse approaches (Wang et al., 2022), often employing B-splines of varying orders for joint space interpolation (Gasparetto & Zanotto, 2007). However, relying on a singular trajectory type for individual movements, particularly over short distances, presents computational inefficiencies and challenges. Additionally, obstacles can compromise trajectory effectiveness and disrupt movement smoothness (D. Chen et al., 2020; Chettibi, 2019). The research demonstrates that integrating quaternion representation with hybrid trajectory methods so far improves motion smoothness in industrial robots by minimizing jerk and leveraging the strengths of linear-SLERP, combined with B-Spline and Bézier curves. This approach provides adaptive and efficient path planning tailored to varying environmental conditions. The findings emphasize the importance of effective control strategies (Ren et al., 2023) in achieving stable and predictable robot movements, offering a novel and robust solution for complex robotic applications. In this context, it is crucial to address the following issues:

1. Relying on a single trajectory for motion may resulting a jerky and dangerous trajectory during complex and lengthy approaches.

2. Identifying best points for the best trajectory poses significant challenges in robotic course layouts, even with advanced RRT*(Schulman et al., 2014; Tahir et al., 2018) and A*(Al-Ansarry & Al-Darraji, 2021) algorithms. RRT* is practical for high-dimensional spaces but suffers from high computational complexity and slow convergence. A*, while comprehensive in austere environments, struggles with excessive time complexity, high memory intake, and inefficiency in dynamic situations.

3. Traditional methods for planning robot orientation, such as Euler angles and rotation matrices, can introduce inefficiencies. New approaches are required to improve these methods (Lee, 2023).

4. Methods like CHOMP (Zucker et al., 2013) and TrajOpt (Schulman et al., 2014) involve solving non-linear optimization problems, which can be computationally expensive, particularly for high-dimensional systems. This complexity can limit their real-time performance, making them unsuitable for applications requiring fast response times.

By addressing these gaps, the research proposes the following solutions:

1. Create smooth, unique trajectories by omitting unnecessary points when no obstacles are detected, assuming a collision-free, well-known, rigid environment.

2. Remove the constraint of choosing a single trajectory by adopting multiple trajectory types and segmenting the total travel mission.

3. Provide the robot and the operator the flexibility to choose the best trajectory for each segment, describing each gripper's status at work and offering reliability and adaptability in diverse operations.

4. Utilize a robot with seven degrees of freedom to enhance motion flexibility, smoothness, and manipulation accuracy.

While previous approaches, such as those in (C. Wang et al., 2023), have focused on handling more than two trajectory types and accommodating seven degrees of freedom, this study emphasizes addressing spherical interpolation trajectories for a comprehensive solution in industrial applications. The core of this investigation revolves around using quaternion orientation over Euler angles to mitigate gimbal-locking errors. Our main objective is to combine Linear-SLERP with Bézier and B-Spline trajectories in simple scenarios, comparing the outcomes like velocity, acceleration, jerk (The kinematic constraints" C^2 *continuity* ") and total trajectory smoothness with a singular trajectory attitude type by utilizing CoppeliaSim as a leading platform to achieve this target. The flowchart in (Fig.1) will depict this method procedure.

The forthcoming sections outline the organizational framework of the research. The second section introduces the research methodology for trajectory optimization integration. A detailed presentation of the simulation setup and procedures follows this. The fourth section presents the simulation setup, while the fifth section presents the results and discusses them, offering insights into their implications. Finally, the last section encapsulates this conclusion, summarizes vital outcomes, and outlines potential avenues for future research.

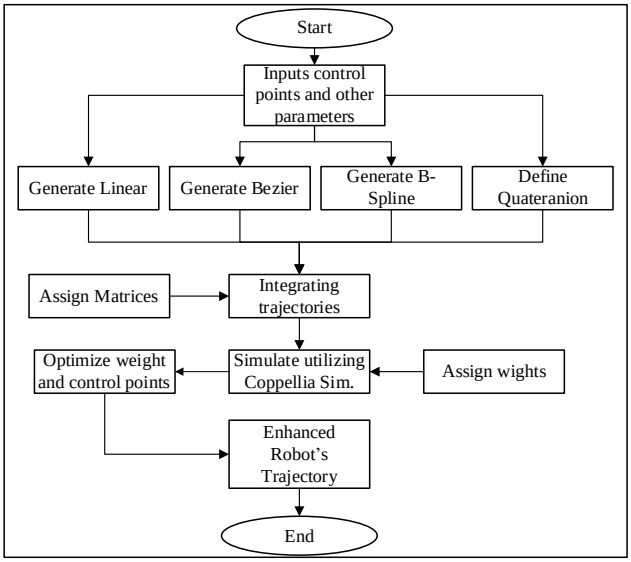

Figure 1. Peoposed method flowchart.

2. RESEARCH METHODOLOGY

2.1 Defining quaternions representation:

The quaternion orientation is essential in traditional trajectory motion for describing oreintation in industrial robotics, where precise and smooth motion control is required. The quadrant consists of four components: one real component and three imaginary components, represented by(Joldes & Muller, 2020):

$$q = (q_x, q_y, q_z, q_m) \tag{1}$$

Where *x*, *y*, and *z*, components of a quaternion, define the axis of rotation in 3D space, and the *m* component represents the magnitude of rotation about that axis.

Normalization of the quaternion is necessary to maintain alignment, stability, and efficiency in rotational activities. The quaternion q is normalized by dividing each quaternion component by its value. The value of the quaternion q is calculated as follows.

$$q = \sqrt{q_x^2 + q_y^2 + q_z^2 + q_m^2} \tag{2}$$

The normalized quaternion is:

$$q' = \left(q_x', q_y', q_z', q_m'\right) \tag{3}$$

Given by:

$$q_x' = \frac{q_x}{\|q\|}\ ,\ q_y' = \frac{q_y}{\|q\|}\ , q_z' = \frac{q_z}{\|q\|}\ , q_m' = \frac{q_m}{\|q\|}$$

If q is zero, the function returns the default quaternion, typically (*0,0,0,1*), to avoid division by zero.

Quaternion helps to properly initiate between directions, an essential process for producing fluid and natural motion in robots.

In industrial robots, SLERP is a technique used to project between two rotational orientations, usually represented by a Quaternion (Jiang et al., 2021). The quaternion interpolation process using SLERP bridges the rotational gap between two orientations, represented by the quadratics q and p. The equation governs the interpolations (Legnani et al., 2021):

$$SLERP(q,p,t) = q(q^{-1}p)^t \quad (4)$$

For parameter t ranging from *0* (corresponding to orientation q) to *1* (corresponding to orientation p), the process is called SLERP. The gain of the quaternion $d = q-1p = \{cos(t\phi/2), Sx\ sin(t\phi/2), Sy\ sin(-t\phi/2), Sz\ sin(-t\phi/2)\}$ is analogous to the power of t, at the angle ϕ, to rotate the angle t times, so the quadratic d^t:

$$d^t = \{cos(\tfrac{tj}{2}), S_x sin(\tfrac{tj}{2}), S_y sin(\tfrac{-tj}{2}), S_z sin(\tfrac{-tj}{2})\} \quad (5)$$

Where:

ϕ is the total angle of rotation between the initial and final quaternions.

j the interpolation index for steps varying from *0* to *1*.

t the interpolation parameter, varying from *0* to *1*.

So, if $t = 0, 1/n, 2/n..., 1$, SLERP essentially involves applying a rotation of ϕ/n at each discretization step, where n - *1* represents the number of intermediate orientations. Ensuring that ϕ does not exceed 180°; otherwise, must invert one of the two quaternions and perform the calculations again. The SLERP equation (5) can be expressed equivalently using the following equation, which is computationally more efficient:

$$SLERP(q,p,t) = \frac{sin((1-t)\phi/2)}{sin(\phi/2)} q + \frac{sin(t\phi/2)}{sin(\phi/2)} p \quad (6)$$

Where ϕ represents the rotation angle between vectors q and p, both SLERP functions are identical, ensuring the outcome is a unit quaternion for any given t. An expeditious method for interpolating between orientations along an identical rotation path about the equivalent axis involves applying the following straightforward formula:

$$LERP(q,p,t) = (1-t)q + tp \quad (7)$$

An alternative to SLERP for small rotation angles (less than 90°) is Linear Interpolation (LERP), which is less computationally intensive but lacks the constant speed of SLERP and requires normalization. However, for rotations up to 180° apart, SLERP is the preferred method in industrial robotics due to its smoother reorientation capabilities (M. Dong et al., 2020).

2.2 Defining B-Spline trajectory:

The B-spline (Basis Spline) curves are piecewise-defined polynomial functions that ensure smoothness and flexibility in trajectory planning. The B-spline curve is defined using control points and a degree parameter. A B-spline curve *q(t)* is defined as (Li et al., 2022):

$$q(t) = \sum_{i=0}^{t} N_{i,p}(t) P_i, \quad t_0 < t < t_e \quad (8)$$

Where: $N_{i,p}(t)$ is the degree p of the B-spline basis function defined over a knot vector T. P_i are the control points. t is the parameter. The knot vector T for a clamped B-spline curve is given by:

$$T = \{t_0, \dots t_0, t_{p+1}, \dots, t_{n+1}, \dots, t_{n+p+1}\} \quad (9)$$

The first $p+1$ and the last $p+1$ values are repeated to ensure the curve starts and ends at the first and last control points. Cubic B-splines produce soft interpolated curves(L. Zhang et al., 2021). A series of $n+1$ control points determine a cubic B-spline curve: $p_0, p_1, \dots, p_n$. The section of the B-spline curve connecting any two adjacent control points takes the form of a cubic polynomial. The cubic polynomial between the control points P_i and P_{i+1} is (Riboli et al., 2023):

$$B(t) = A_i(1-t)^3 + B_i(1-t)^2 t + C_i(1-t)t^2 + D_i t^3 \quad (10)$$

Where t goes from *0* to *1*, the parameters A_i, B_i, C_i, and D_i are determined based on position, slope, and continuity limits between components. This leads to the following formulas:

$$A_i = (1/6)P_i \quad (11)$$
$$B_i = (3/6)P_i + (3/6)P_{i+1} \quad (12)$$
$$C_i = (-3/6)P_i + (3/6)P_{i+1} + (1/6)P_{i+2} \quad (13)$$
$$D_i = (1/6)P_{i+1} \quad (14)$$

Substituting the coefficients into the cubic polynomial in (10) will get (15) as follows:

$$B(t) = \frac{(1-t)^3}{6} P_i + \left(\frac{3t^3 - 6t^2 + 4}{6}\right) P_{i+1} + \frac{-3t^2+3t^2+3t}{6} P_{i+2} + \frac{t^3}{6} P_{i+3} \quad (15)$$

Well, smoothness results from B-Spline curves sharing standard endpoints and having the same slope at those endpoints. The first, second and third derivatives for (15) Will give us the velocity, acceleration and jerk equations, respectively, as follows:

$$B'(t) = \left(-\frac{(1-t)^2}{2}\right) \cdot P_i + \left(\frac{3t^2-4t}{2}\right) \cdot P_{i+1} + \left(\frac{-3t^2+2t+1}{2}\right) \cdot P_{i+2} + \left(\frac{t^2}{2}\right) \cdot P_{i+3} \quad (16)$$

$$B''(t) = (1-t).P_i + (3t-2).P_{i+1} + (-3t+1).P_{i+2} + t.P_{i+3} \quad (17)$$

$$B''' = (-1).P_i + 3P_{i+1} + (-3)P_{i+2} + 1P_{i+3} \quad (18)$$

Cubic B-splines are essential in robotic trajectory planning for their smoothness and C^2 continuity, ensuring the curve and its first and second derivatives are continuous. These functions prevent jerky motions and reduce mechanical stress. Their local control property allows precise adjustments to a limited curve region without affecting the overall trajectory, which is vital for real-time applications. Cubic B-splines, Known for their numerical stability, are suitable for computationally intensive tasks and can be efficiently evaluated using recurrence relations. The cubic polynomial degree balances

flexibility and efficiency, while their non-interpolating nature results in smoother, more desirable trajectories.

2.3 Defining Bézier trajectory type

Béziers method is ideal for applications requiring precision and simplicity, such as automation or assembly. Any interval of the generated B-spline trajectory is also a Beizer trajectory with the same degree. The mathematical equations for a Bézier curve with *n* degrees are given (Deolasee et al., 2023; W. Dong et al., 2019):

$$b(t) = \sum_{i=0}^{n} P_i \binom{n}{i} (1-t)^{n-i}\, t^i \quad (19)$$

Where, *b(t)* area on the Bézier curve over a given parameter *t* (where $0 \leq t \leq 1$). P_i control points of the curve. $\binom{n}{i}$ Binomial coefficient *i,n* is the degree of the curve (number of points minus one).

In this equation, the curve starts at P_0 and ends at P_n, with central control points directing the curve. The flexibility of the trajectory enables the development of robust and accurate methods tailored to the needs of dynamic robotic systems. The Bézier curve is defined by a series of control points, $P_0, P_1, \ldots, P_n$. The mathematical formula for a cubic Bézier curve with four control points is:

$$b(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t)t^2 P_2 + t^3 P_3, \quad t \in [0,1] \quad (20)$$

Where: *t* ranges from *0* to *1*. Each method generally has its strengths and applications. Bézier curves are flexible and easy to avoid obstacles, and B-spline curves are simple, efficient, and can handle complex curves. The choice of method depends on the robot application's specific needs, including the trajectory complexity and the desired flexibility and efficiency. The velocity, acceleration and jerk equations for the Bézier curve derivative will be as follows:

$$b'(t) = -3(1-t)^2 P_0 + 3(1-t)(1-3t)P_1 + 3t(2-3t)P_2 + 3t^2 P_3 \quad (21)$$

$$b''(t) = 6(1-t)P_0 + 3(6t-4)P_1 + 3(2-6t)P_2 + 6tP_3 \quad (22)$$

$$b''' = -6P_0 + 18P_1 - 18P_2 + 6P_3 \quad (23)$$

Invariably, the derivative of B-Spline is in a lower order than the main B-Spline.

2.5 Integration trajectory weights

To combine multiple planning methods to propose a comprehensive and smooth trajectory, such as linear -SLERP, B-Spline, and Bézier curves based on quaternion. Each method contributes to the final trajectory with a specific weight. For example, the weights of each control point in the B-spline segment are determined based on the *B-spline basis functions*. These weights are a time-varying factor influenced by the local control points and the normalized time parameter. The mathematical relation can be expressed as follows (Tang et al., 2021):

$$x(t) = \sum_{i=0}^{n} p_i\, B_{i,k}(t) \quad (24)$$

Bi,k(t) is the B-spline basis function, always not zero at time *t*, and it is called Local Control Property (LCP), and *pi* is the control point, which is always four points in this scenario since using Cubic-B-Spline so that each segment will be equal to *n-3*. The weights are chosen based on their respective specific task requirements. The combined position trajectory is formulated as follows:

$$P_{combined}(t) = w_1 . p_{linear}(t) + w_2 . p_{bspline}(T) + w_3 . p_{bezier}(t) \quad (25)$$

Where $P_{bspline}, P_{bezier}$ and P_{linear} are the positions of the segments derived from the respective methods. (w_1,w_2 and w_3) are the weights assigned to Linear-SLERP, B-Spline, and Bézier curves, respectively. The normalized weights are such that $w_1+w_2+w_3=1$. Adjusting the weights can control the influence of each trajectory planning method, ensuring that the final trajectory is smooth, efficient, and suitable for the specific robotic task (Likhachev M, 2003; Nantabut & Abel, 2023). The choice of weights is critical and can be optimized based on empirical tuning or specific optimization criteria like velocity, acceleration and jerk in the research. The knot vector *T* for a clamped B-spline curve was previously mentioned in (9). The knot vector is needed for the B-spline knot vector *T* like $T=\{p_0, p_0, p_0, p_1, p_2, p_3, p_3, p_3\}$ between p_0 *and* p_3. While *t* always represents the period relies (0~1). For example, in calculation, suppose the following parameters:

Control Points: p_0,p_1,p_2,p_3p_4.

Knot Vector for B-Spline: $T=\{p_0, p_0, p_0, p_1, p_2, p_3, p_4, p_4, p_4\}$

Quaternions: q_1,q_4.

Weights: $w_1=0.3,w_2=0.25,w_3=0.30,w_4=0.15$.

Interpolation Parameter: $t \in [0,1]$.

Calculate each component:

Linear Translation: $p_{linear}(t)$.

SLERP: $q_{slerp}(t)$.

B-Spline: $p_{bspline}(T)$.

Bézier : $p_{bezier}(t)$.

Combine them using (25):

$$P_{combined}(t) = 0.3 . p_{bezier}(t) + 0.25 . p_{bspline}(T) + 0.3 . p_{bspline}(T) + 0.15 . p_{linear}(t) \quad (26)$$

Note that linear translation is a conventional transition from point to point, and SLERP is well-controlled to direct it smoothly for orientation directing.

2.5 Problem Description

Our approach prioritizes C^2 continuity to ensure that the planned trajectories are not only geometrically smooth but also feasible, supporting the precise and stable operation of the robot. It can describe the problem and formulate it to minimize the jerk, but jerk is also related to velocity and acceleration. Also, time consumption will change according to these. During the total trajectory, the length will be ensured to show different attitudes according to trajectory type and settings, and

feasible motion will be the desired target. For that, the mathematical representation of the problem can be as follows (F. Wang et al., 2022; Wu et al., 2022):

$$\text{Minimize } J = \int_{t_0}^{t_f} \left(\dddot{k}(t)^2 + \alpha \dot{k}(t)^2 + \beta \ddot{\theta}(t)^2\right) dt \qquad (27)$$

Where:

k(t) represents the position of the robot joints over time.

$\dot{k}(t)$ is the velocity of the robot joints.

$\ddot{k}(t)$ is the acceleration of the robot joints.

$\ddot{\theta}$ (t) Angular acceleration of the end-effector.

α and β are weighting factors.

That will give the main Kinematic Constraints to ensure the robot's movements are feasible will be as follows:

$$k_{min} \le k_i(t) \le k_{max}$$
$$\dot{k}_{min} \le \dot{k}(t) \le \dot{k}_{max}$$
$$\ddot{k}_{min} \le \ddot{k}_i(t) \le \ddot{k}_{max}$$

Where $k_{min}, k_{max}, \dot{k}_{min}, \dot{k}_{min}, \ddot{k}_{min}, \ddot{k}_{min}$ are the minimum and maximum values for joint angles, velocities and accelerations, respectively.

3. SIMULATION SETUPS

3.1 Preparing simulation environment

The simulation was conducted on a Windows 10 with Intel i5 processors and a 16 GB RAM system using CoppeliaSim. EDU. v 4.0 (Coppelia Robotics, 2019). The robot used was the Franka Emika Panda, a 7-DoF robotic arm, with the Robotiq 2-Finger-85 gripper as the end-effector. The simulation platform is listed in (Fig.2) below.

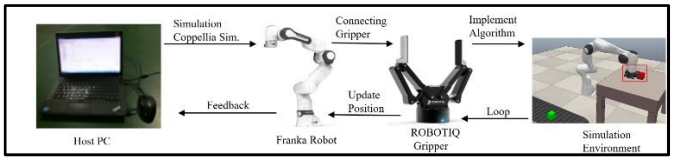

Figure 2. Simulation platform.

The research developed a Lua script to manage object manipulation, specifically handling a cubic object, within the CoppeliaSim application framework to orchestrate the robot's operations. The primary objective of this research is to minimize jerky motions during the robot's operations. To simplify the operating scenario, assuming the workspace is free of obstacles, thus avoiding the need for collision detection calculations. The proposed scenario involves a pick-and-place operation using a cubic object to demonstrate the manipulation process. The robot will pick up the cube from a start point p_i on a conveyor and place it on the other side at p_e on the table. During this process, assuming the robot's gripper will pass through several control points in the workspace to ensure a smooth trajectory. (Fig.3) illustrates the complete sequence of steps in this scenario.

To enhance the overall efficiency of the pick-and-place task, it has divided the process into four distinct actions:

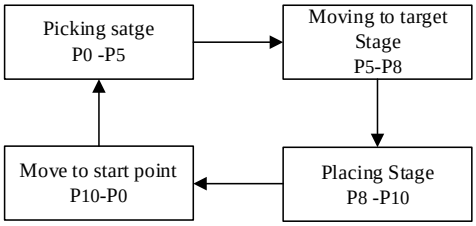

Figure 3. Pick-and-place scenario.

Action 1: Move from P_0 to P_5 and pick the object.
Action 2: Move from P_5 to P_8 to reach the target.
Action 3: Move from P_9 to P_{10} and place the object.
Action 4: Move from P_{10} to P_0.
The workspace environment established in Coppellia sim. is shown below in (Fig.4).

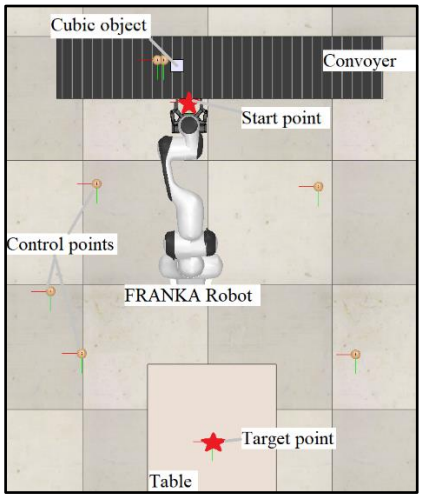

Figure 4. The robot workspace.

3.2 Trajectory generating

In terms of State-of-Art, it conducted simulations to make a comparison of the performance of the proposed hybrid trajectory planning method with individual trajectory methods: Linear—SLERP (Jiang et al., 2021), B-Spline in (Li et al., 2022), and Bézier curves with(Nguyen et al., 2023). The performance metrics considered were velocity, acceleration, and jerk. The simulations were performed using the Franka Emika Panda robot executing a standard pick-and-place task.

3.2.1 Generating linear-SLERP trajectory

First, the research employed conventional translation from point to point (Aalhasan et al., 2016), a hybrid approach for positional interpolation with (6) of SLERP for orientation representation to achieve simple trajectory planning. The trajectory of traditional implementation persists, passing through all points for all previous 4 actions of the pick-and-place task, producing acute angles during the transition from point to point, which leads to jerky motion. This hybrid approach ensures coordinated position and orientation changes. The total trajectory attitude is shown in (Fig.5) below.

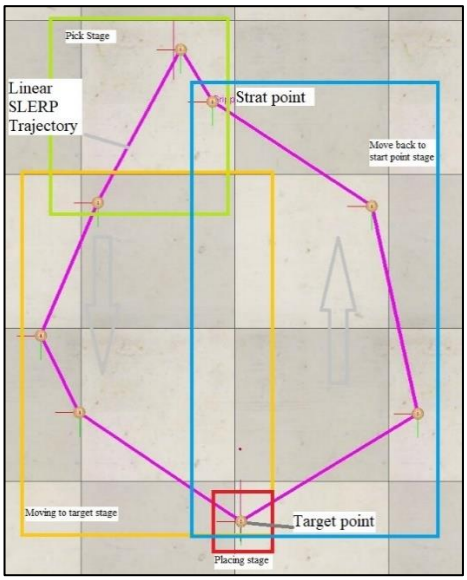

Figure 5. Linear interpolation combined with SLERP.

In the simulation, the `moveTo` function handles linear translation and SLERP for smooth positional and rotational transitions. For example, moving from `p1` to `p2` with constant orientation `orientation5_q` is implemented using Lua code as:

```
moveTo(p1, p2, orientation5_q, orientation5_q)
```

3.2.2 Generating B-Spline trajectory

Recalling (10), implement (15), generate a B-Spline trajectory, and utilize quaternion representation. The pick-and-place task repeats the same 4 actions, resulting in the trajectory shown in (Fig. 6). Notably, the trajectory does not pass through all points directly, creating a smooth curve passing beside some of these points since (t) duration between points is divided into several parts.

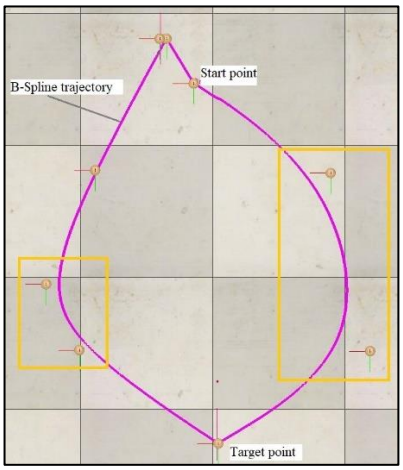

Figure 6. B-spline combined with quaternion.

The B-Spline trajectory is generated by defining the control points and calculating the cubic- B-Spline basis functions. The simulation implements the trajectory using the Lua script, which interpolates the positions and orientations. Snippet lua code will be like this:

```
local bSplinePoints = {p1, p1, p1, p2, p3, p4, p4, p4}
for t = 0, 1, stepSize do
    local position = calculateBSplinePosition(t, bSplinePoints)
    local orientation = calculateBSplineOrientation(t, orientation1_q, orientation2_q)
    setGripperPositionAndOrientation(position, orientation)
    sim.wait(sim.getSimulationTimeStep())
end
```

3.2.3 Generating Bézier Trajectory

Invoking (20) combined with quaternion to generate the desired Bézier trajectory. (Fig.7) illustrates the Bézier trajectory, characterized by its ability to generate smooth curves that pass close to the defined control points. However, it is essential to note that while the Bézier trajectory attempts to create a smooth path, it tends to form curves that approximate a straight line between control points. This behaviour can lead to potential self-collisions for the robot, particularly in scenarios where the trajectory closely follows a linear path.

Another significant observation from the Bézier trajectory is the varying distance between the control points and the actual trajectory. This distance can increase, leading to deviations from the intended path.

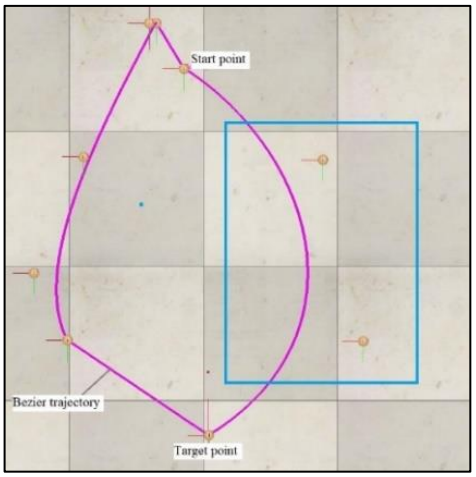

Figure 7. Bezier combined with quaternion.

The simulation implements the trajectory using the Lua script, which interpolates the positions and orientations. Snippet lua code will be like this:

```
local Bézier Points = {p1, p2, p3, p4}
for t = 0, 1, stepSize do
    local position = calculateBézier Position(t, Bézier Points)
    local orientation = calculateBézier Orientation(t, orientation1_q, orientation2_q)
    setGripperPositionAndOrientation(position, orientation)
    sim.wait(sim.getSimulationTimeStep())
end
```

3.2.4 Generating Hybrid Trajectory

Since it can transform a linear trajectory into other types of curves by defining new control points that approximate the linear motion and utilize the conversion matrix between the B-spline to Bézier and vice versa, the matrix detailed methodologies in (Nguyen et al., 2023)and (Romani & Sabin, 2004) are as follows:

$$P_j = P_j A(d, n, j) \tag{28}$$

Where, P_j are the B-spline control points, and *A(d,n,j)* is the conversion matrix. It can propose a hybrid method that integrates multiple techniques to enhance trajectory planning for robotic motion in workspace environments. This trajectory must ensure that i) reaching the number of control points is feasible, ii) the first and final point and interval are available, and iii) the orientation transformation-based quaternion is applicable. This approach ensures smooth and efficient motion by leveraging the strengths of these techniques. Since it is four actions for the pick-and-place task and three trajectory methods: Linear Translation, B-Spline, and Bézier, the combined position trajectory can be represented according to the sequence of the method in the trajectory, as (*B-spline, Bézier, B-spline, Liner*), (25) will constantly dividing weight according to the diverse of trajectory to be equal *1*, the mathematical equation will be such as:

$$P_{combined}(t) = w_1.p_{bezier}(t) + w_2.p_{bspline}(T) + w_3.p_{bspline}(T) + w_4.p_{linear}(t) \quad (29)$$

The research is already performing a pick-and-place task with the following considerations: Smoothness is prioritized during the *placing* stage to avoid sudden movements. Speed is prioritized when *moving* to the target stage to minimize operation time. Accuracy is crucial during the *picking* stage to ensure precise object handling. Based on these requirements and extensive tests, it is assigned weights as follows:

- *Picking Stage (Bézier Curve): w_1=0.30*

Prioritizes smoothness and precision to ensure accurate object handling and minimize sudden movements.

- *Moving to Target (B-Spline Curve): w_2=0.25.*

Balances speed and smoothness, allowing for rapid movement while maintaining control.

- *Placing Stage (B-Spline Curve): w_3=0.30.*

Emphasizes smoothness and stability to ensure precise and controlled placement of the object.

- *Moving Back to initial (Linear-SLERP): w_4=0.15.*

It focuses on stability and coordinated motion to return accurately to the start point without unnecessary complexity.

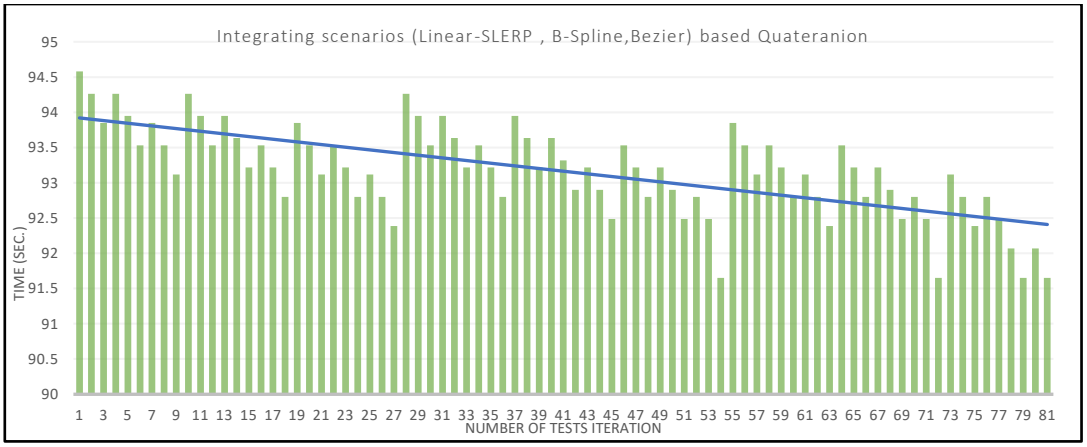

Figure 8. Time consumption for 81 scenarios integrating different types of trajectories.

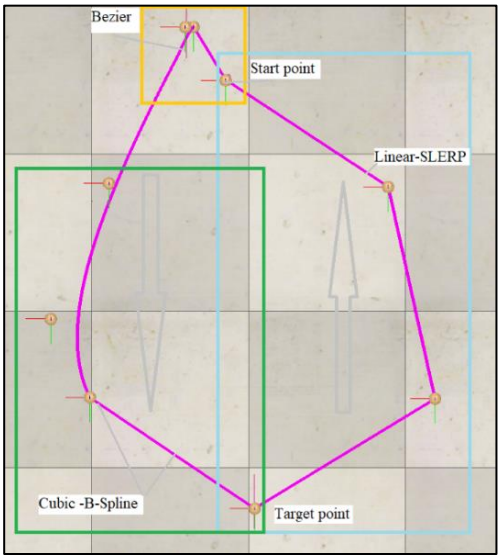

Figure 9. The proposed trajectory curve test no.67.

Employed a complete factorial design for practical experiments involving the three types of trajectories. Implemented a pick-and-place task scenario for each trajectory type to comprehensively compare the results by implementing different weight values for each trajectory segment. This resulted in a total of (3^4 =81) trajectory representations for each of the four actions. It measured the time taken for each trajectory segment to assess the temporal performance of different planning methods. These trajectories are comprehensively illustrated in (Fig.8). During the testing phase, the mean duration was recorded at 93.16 seconds.

A representative subset of scenario, precisely scenario 67, was selected for detailed analysis due to its proximity to the average temporal metric and diversity of trajectory types. The trajectories generated using test no.67 adopted Bézier for action 1(picking stage) and B-spline for actions 2 and 3 (move to target and placing stages). Finally, the last action, 4 (moving to the start point), is implemented using Linear-SLERP trajectory. (Fig.9) showing the proposed trajectory curve. By assigning weights, the hybrid trajectory planning method

ensures that each stage of the pick-and-place task is optimized for its specific requirements, effectively leveraging each trajectory planning method's strengths.

4. RESULTS AND DISCUSSION

4.1 Simulation results

The research implemented the traditional technique (Jiang et al., 2021) to generate a Liner trajectory transfer directly from one point to another, forming a stand-alone pick-and-place task, making the robot's gripper translate from the (P_i) start point to the (P_e) target point along this trajectory utilizing SLERP to control orientation. The shape of the trajectory is noticed in (Fig.5) before. The joint jerks graphs will be shown in (Fig.10) below. Table 1. will list the result of this motion in terms of kinematics constraints (velocity, acceleration and jerk).

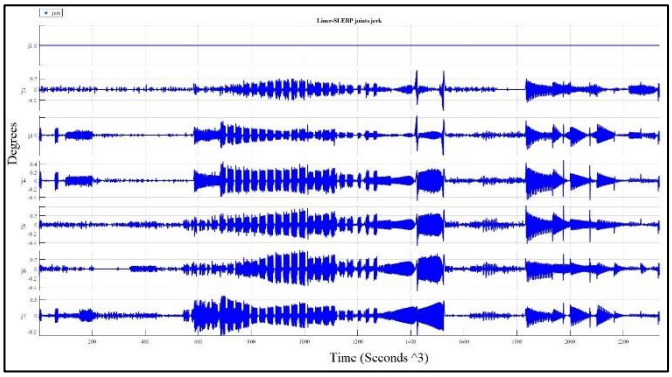

Figure 10. Joints jerk of Linear-SLERP trajectory.

Table 1. Linear-SLERP motion results

Joint	Average Velocity (°/Sec.)	Average Acceleration (°/Sec.^{2})	Average Jerk (°/Sec.^{3})
Joint1	0	0.00	0.00
Joint2	6.99	0.22	-0.05
Joint3	11.47	0.16	-0.06
Joint4	14.90	0.14	-0.06
Joint5	14.01	0.20	-0.05
Joint6	14.02	0.21	-0.04
Joint7	17.62	0.20	-0.05
Average Deviation	8.876	2.35	42.9

Then, continue with B-spline generating the traditional trajectory(Li et al., 2022), referring to (Fig.6). The robot gripper is trying to make a smooth curve passing by control points. (Fig.11) will depict the joint jerk graphs, while Table 2. will introduce the results of joint motion.

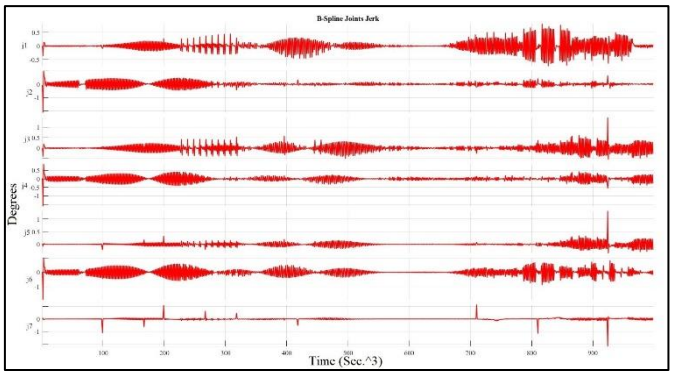

Figure 11. Joints jerk of B-Spline trajectory.

Table 2. B-Spline motion results

Joint	Average Velocity (°/Sec.)	Average Acceleration (°/Sec.^{2})	Average Jerk (°/Sec.^{3})
Joint1	13.72	−0.172	−0.00058
Joint2	−5.48	−0.172	0.098
Joint3	11.12	−0.801	−0.250
Joint4	5.91	−0.801	−0.336
Joint5	3.49	0.224	−0.129
Joint6	−6.18	0.264	−0.009
Joint7	0.01	0.232	−0.080
Average Deviation	13.20	5.92	76.58

Utilizing the Bézier method in (Nguyen et al., 2023) to obtain the results of its trajectory, as shown in (Fig.7). Repeat the same parameters for the workspace environment and task. (Fig.12) shows the performance of joint jerks related to the trajectory, and Table 3. shows the results of the gripper trajectory passing near the control points mentioned before.

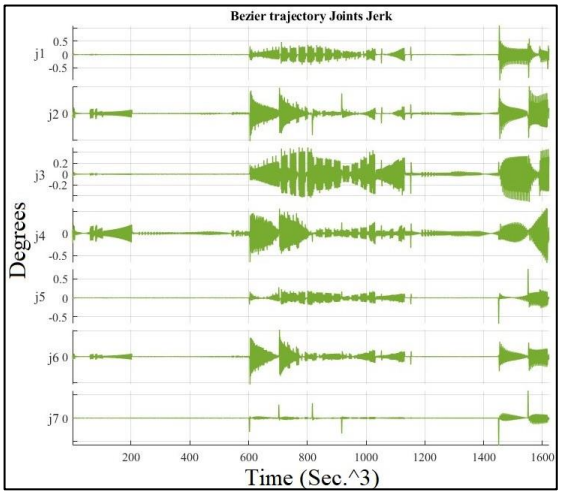

Figure 12. Joints jerk of Bezier trajectory.

Table 3. Bézier motion results

Joint	Average Velocity (°/Sec.)	Average Acceleration (°/Sec.^{2})	Average Jerk (°/Sec.^{3})
Joint1	6.64	0.59	0.052
Joint2	-0.0039	-0.25	0.063
Joint3	8.047	0.42	0.077
Joint4	-0.720	-0.088	-0.060
Joint5	0.0798	0.11	0.000168
Joint6	-0.51	0.034	-0.067
Joint7	-4.53	0.56	-0.030
Average Deviation	9.24	96.822	122.85

Finally, the integrating test of the method's different trajectories is implemented, as shown in (Fig.9), the jerk graph in (Fig. 13) and the results in Table 4

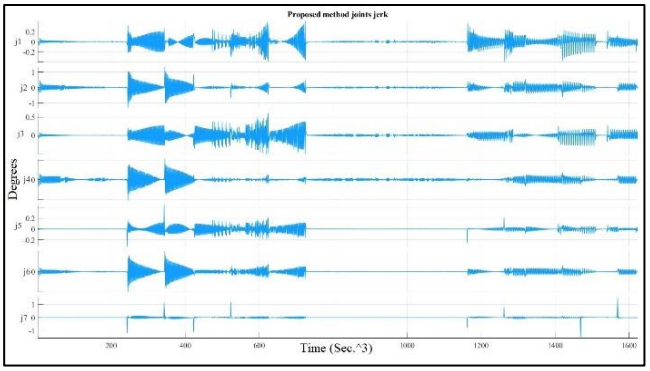

Figure 13. Joints jerk of Proposed trajectory.

Table 4. Proposed method motion results

Joint	Average Velocity (°/Sec.)	Average Acceleration (°/Sec.^{2})	Average Jerk (°/Sec.^{3})
Joint1	7.75	0.275	-0.032
Joint2	-0.598	-0.383	-0.013
Joint3	8.71	0.009	-0.038
Joint4	-0.72	-0.188	0.022
Joint5	0.16	0.012	-0.0085
Joint6	-1.23	0.025	-0.008
Joint7	-1.002	-0.085	0.13
Average Deviation	7.72	3.00	35.79

The Linear-SLERP method exhibited an average deviation velocity of 8.876 °/sec, average deviation acceleration of 2.35 °/sec², and average deviation jerk of 42.9 °/sec³. While the method maintained controlled motion with relatively low velocity and acceleration, the higher jerk indicated less smooth transitions, leading to potential mechanical stress. The B-Spline trajectory showed an average deviation velocity of 13.20 °/sec, average deviation acceleration of 5.92 °/sec², and average deviation jerk of 76.58 °/sec³. This method provided smooth paths by not strictly passing through all control points, resulting in higher velocity and acceleration. The higher jerk values suggested rough transitions and less controlled motion. The Bézier trajectory recorded an average deviation velocity of 9.24 °/sec, average deviation acceleration of 96.822 °/sec², and average deviation jerk of 122.85 °/sec³. Despite attempting to create smooth paths, significant deviations from the intended path led to higher acceleration and jerk, indicating jerky motions. The proposed hybrid method achieved the best overall performance by combining linear-SLERP, B-Spline, and Bézier curves. It had the lowest average deviation velocity of 7.72 °/sec, average deviation acceleration of 3.00 °/sec², and average deviation jerk of 35.79 °/sec³. This demonstrated that the hybrid approach effectively balanced smoothness and control, resulting in minimal mechanical stress and efficient motion.

4.2 Comparative analysis

Smoothness calculation (Gasparetto & Zanotto, 2007) is utilized to achieve the target as follows:

$$S = w_v.\bar{V} + w_a.\bar{A} + w_j.\bar{J} \quad (30)$$

Where:

S = Smoothness score.

$\bar{V}$= Average velocity.

$\bar{A}$ = Average acceleration.

$\bar{J}$= Average jerk.

w_v, w_a, w_j= Weights for velocity, acceleration, and jerk, respectively, assuming equal weights (1/3) for simplicity.

The comparative analysis of the trajectory planning methods—Linear-SLERP, B-Spline, Bézier, and the proposed hybrid method—was conducted using key performance metrics: trajectory duration, length, and smoothness. The Linear-SLERP method, with a trajectory duration of 118.8 seconds and length of 4.25 meters, achieved a trajectory smoothness average of 18.04, indicating a balance between smoothness and time efficiency. The B-Spline method, which had the shortest trajectory duration at 51.6 seconds and a length of 4.228 meters, recorded a higher smoothness average of 31.90, suggesting less smooth motion despite efficient path generation. The Bézier method showed a trajectory duration of 85.75 seconds and a length of 4.068 meters. However, it had the highest smoothness average of 76.30, reflecting its struggle to maintain control and resulting in jerky motions. The proposed hybrid method demonstrated superior performance, with a trajectory duration of 93.1 seconds, a length of 4.238 meters, and the lowest smoothness average of 15.50, indicating

the smoothest trajectory. This method effectively integrates linear-SLERP, B-Spline, and Bézier trajectories, optimizing for smoothness and control. The hybrid approach's ability to balance these metrics highlights its potential for applications requiring precise and smooth robotic motion. Table 5. shows the enhancement results of this approach.

Table 5. Comparative results

Trajectory method	Trajectory duration (sec)	Trajectory length (m)	Trajectory smoothness Score
Linear - SLERP (Jiang et al., 2021)	118.8	4.25	18.04
B-Spline (Li et al., 2022)	51.6	4.228	31.90
Bézier (Nguyen et al., 2023)	85.75	4.068	76.30
Our proposed Method	93.1	4.238	15.50

5. CONCLUSION

This study proposed a comprehensive comparative analysis of trajectory planning methods specifically Linear-SLERP, B-Spline, and Bézier, and a proposed hybrid method to enhance robotic motion smoothness and efficiency . The performance of each method was meticulously evaluated based on trajectory duration length and smoothness . The Linear-SLERP method showed controlled motion with low average velocity and acceleration but higher jerk values indicated less smooth transitions potentially leading to mechanical stress. The B-Spline method demonstrated the shortest trajectory duration reflecting efficient path generation yet higher jerk values suggested rough transitions and reduced smoothness . Although designed for smooth paths, the Bézier method struggled with control resulting in the highest smoothness average and significant deviations, leading to jerky motions.

In contrast, the proposed hybrid method integrating Linear-SLERP, B-Spline and Bézier trajectories achieved superior performance with the lowest average jerk indicating the smoothest trajectory while maintaining balanced trajectory duration and length . This hybrid approach effectively combines the strengths of each trajectory type optimizing for smoothness and control thereby minimizing mechanical stress and enhancing overall motion efficiency . Future research should incorporate dynamic constraints to ensure practical applicability and safety and enhance obstacle avoidance capabilities for deployment in complex environments .

ACKNOWLEDGMENT

China's National Key R&D Program supported the work under the grant（2023YFB4704000).

REFERENCES

Aalhasan, A. J. J., XiaoQi, T., & Bao, S. (2016). Collision Detection and Trajectory Planning for Palletizing Robots Based OBB. *Indonesian Journal of Electrical Engineering and Computer Science*, *1*(1), 109. https://doi.org/10.11591/ijeecs.v1.i1.pp109-118

Al-Ansarry, S., & Al-Darraji, S. (2021). Hybrid RRT-A*: An Improved Path Planning Method for an Autonomous Mobile Robots. *Iraqi Journal for Electrical and Electronic Engineering*, *17*(1), 1–9. https://doi.org/10.37917/ijeee.17.1.13

Chen, D., Li, S., Li, W., & Wu, Q. (2020). A Multi-Level Simultaneous Minimization Scheme Applied to Jerk-Bounded Redundant Robot Manipulators. *IEEE Transactions on Automation Science and Engineering*, *17*(1), 463–474. https://doi.org/10.1109/TASE.2019.2931810

Chen, Y., Li, L., & Ji, X. (2014). Smooth and Accurate Trajectory Planning for Industrial Robots. *Advances in Mechanical Engineering*, *6*, 342137. https://doi.org/10.1155/2014/342137

Chettibi, T. (2019). Smooth point-to-point trajectory planning for robot manipulators by using radial basis functions. *Robotica*, *37*(3), 539–559. https://doi.org/10.1017/S0263574718001169

Cichella, V., Kaminer, I., Walton, C., Hovakimyan, N., & Pascoal, A. M. (2021). Optimal Multivehicle Motion Planning Using Bernstein Approximants. *IEEE Transactions on Automatic Control*, *66*(4), 1453–1467. https://doi.org/10.1109/TAC.2020.2999329

Coppelia Robotics. (2019). *CoppeliaSim* (Edu Version 4.0.). Coppelia Robotics.

Deolasee, S., Lin, Q., Li, J., & Dolan, J. M. (2023). Spatio-temporal Motion Planning for Autonomous Vehicles with Trapezoidal Prism Corridors and Bézier Curves. *2023 American Control Conference (ACC)*, 3207–3214. https://doi.org/10.23919/ACC55779.2023.10155930

Dong, M., Yao, G., Li, J., & Zhang, L. (2020). Research on Attitude Interpolation and Tracking Control Based on Improved Orientation Vector SLERP Method. *Robotica*, *38*(4), 719–731. https://doi.org/10.1017/S0263574719001000

Dong, W., Ding, Y., Huang, J., Yang, L., & Zhu, X. (2019). An optimal curvature smoothing method and the associated real-time interpolation for the trajectory generation of flying robots. *Robotics and Autonomous Systems*, *115*, 73–82. https://doi.org/10.1016/j.robot.2019.02.004

Galcík, A., & Duchon, F. (2024). Mobile Robot Navigation in Known Environment. *Journal of Control Engineering and Applied Informatics*, *26*(2). https://doi.org/10.61416/ceai.v26i2.8969

Gasparetto, A., & Zanotto, V. (2007). A new method for smooth trajectory planning of robot manipulators.

Mechanism and Machine Theory, *42*(4), 455–471. https://doi.org/10.1016/j.mechmachtheory.2006.04.002

Jiang, B., Li, J., Du, X., Duan, P., & Wang, J. (2021). Attitude Interpolation Algorithm for Industrial Robot Based on Quaternion Method. *Journal of Physics: Conference Series*, *1820*(1), 012159. https://doi.org/10.1088/1742-6596/1820/1/012159

Jiao, J., Cao, Z., Zhao, P., Liu, X., & Tan, M. (2013). Bezier curve based path planning for a mobile manipulator in unknown environments. *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 1864–1868. https://doi.org/10.1109/ROBIO.2013.6739739

Joldes, M., & Muller, J.-M. (2020). Algorithms for Manipulating Quaternions in Floating-Point Arithmetic. *2020 IEEE 27th Symposium on Computer Arithmetic (ARITH)*, 48–55. https://doi.org/10.1109/ARITH48897.2020.00016

Kuo, Y.-L., Lin, C.-C., & Lin, Z.-T. (2020). Dual-optimization trajectory planning based on parametric curves for a robot manipulator. *International Journal of Advanced Robotic Systems*, *17*(3), 172988142092004. https://doi.org/10.1177/1729881420920046

Lee, S. (2023). Quaternion-based 3D Attitude Tracking Algorithm Using Virtual Roll Angle for Missile Systems. *International Journal of Control, Automation and Systems*, *21*(10), 3391–3404. https://doi.org/10.1007/s12555-022-0587-5

Legnani, G., Fassi, I., Tasora, A., & Fusai, D. (2021). A practical algorithm for smooth interpolation between different angular positions. *Mechanism and Machine Theory*, *162*, 104341. https://doi.org/10.1016/j.mechmachtheory.2021.104341

Li, X., Gao, X., Zhang, W., & Hao, L. (2022). Smooth and collision-free trajectory generation in cluttered environments using cubic B-spline form. *Mechanism and Machine Theory*, *169*, 104606. https://doi.org/10.1016/j.mechmachtheory.2021.104606

Likhachev M, G. G. T. S. (2003). Anytime A* with provable bounds on sub-optimality. *Advances in Neural Information Processing Systems*, *16*.

Nantabut, C., & Abel, D. (2023). Weighted Hybrid A*-Based Trajectory Planning for Dynamic Environments. *SN Computer Science*, *4*(4), 390. https://doi.org/10.1007/s42979-023-01811-3

Nguyen, N. T., Gangavarapu, P. T., Kompe, N. F., Schildbach, G., & Ernst, F. (2023). Navigation with Polytopes: A Toolbox for Optimal Path Planning with Polytope Maps and B-spline Curves. *Sensors*, *23*(7), 3532. https://doi.org/10.3390/s23073532

Ren, Z., Hu, B., Wang, Z., Sun, L., & Zhu, Q. (2023). Knowledge Database-Based Multiobjective Trajectory Planning of 7-DOF Manipulator With Rapid and Continuous Response to Uncertain Fast-Flying Objects. *IEEE Transactions on Robotics*, *39*(2), 1012–1028. https://doi.org/10.1109/TRO.2022.3207616

Riboli, M., Corradini, F., Silvestri, M., & Aimi, A. (2023). A New Framework for Joint Trajectory Planning Based on Time-Parameterized B-Splines. *Computer-Aided Design*, *154*, 103421. https://doi.org/10.1016/j.cad.2022.103421

Romani, L., & Sabin, M. A. (2004). The conversion matrix between uniform B-spline and Bézier representations. *Computer Aided Geometric Design*, *21*(6), 549–560. https://doi.org/10.1016/j.cagd.2004.04.002

Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., & Abbeel, P. (2014). Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, *33*(9), 1251–1270. https://doi.org/10.1177/0278364914528132

Tahir, Z., Qureshi, A. H., Ayaz, Y., & Nawaz, R. (2018). Potentially guided bidirectionalized RRT* for fast optimal path planning in cluttered environments. *Robotics and Autonomous Systems*, *108*, 13–27. https://doi.org/10.1016/j.robot.2018.06.013

Tang, L., Wang, H., Liu, Z., & Wang, Y. (2021). A real-time quadrotor trajectory planning framework based on B-spline and nonuniform kinodynamic search. *Journal of Field Robotics*, *38*(3), 452–475. https://doi.org/10.1002/rob.21997

Van Loock, W., Pipeleers, G., & Swevers, J. (2015). B-spline parameterized optimal motion trajectories for robotic systems with guaranteed constraint satisfaction. *Mechanical Sciences*, *6*(2), 163–171. https://doi.org/10.5194/ms-6-163-2015

Wang, C., Chen, X., Li, C., Song, R., Li, Y., & Meng, M. Q.-H. (2023). Chase and Track: Toward Safe and Smooth Trajectory Planning for Robotic Navigation in Dynamic Environments. *IEEE Transactions on Industrial Electronics*, *70*(1), 604–613. https://doi.org/10.1109/TIE.2022.3148753

Wang, F., Wu, Z., & Bao, T. (2022). Time-Jerk optimal Trajectory Planning of Industrial Robots Based on a Hybrid WOA-GA Algorithm. *Processes*, *10*(5), 1014. https://doi.org/10.3390/pr10051014

Wang HF, Zhu SQ, & Wu WX. (2012). INSGA-II based multi-objective trajectory planning for manipulators. . *Journal of Zhejiang University. Engineering Science.*, *46*(4), 622-8.

Wang, Z., Li, Y., Shuai, K., Zhu, W., Chen, B., & Chen, K. (2022). Multi-objective Trajectory Planning Method based on the Improved Elitist Non-dominated Sorting Genetic Algorithm. *Chinese Journal of Mechanical Engineering*, *35*(1), 7. https://doi.org/10.1186/s10033-021-00669-x

Wu, Z., Chen, J., Bao, T., Wang, J., Zhang, L., & Xu, F. (2022). A Novel Point-to-Point Trajectory Planning Algorithm for Industrial Robots Based on a Locally Asymmetrical Jerk Motion Profile. *Processes*, *10*(4), 728. https://doi.org/10.3390/pr10040728

Xu, W., Wang, Q., & Dolan, J. M. (2021). Autonomous Vehicle Motion Planning via Recurrent Spline Optimization. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 7730–7736. https://doi.org/10.1109/ICRA48506.2021.9560867

Yi, J., Yuan, Q., Sun, R., & Bai, H. (2022). Path planning of a manipulator based on an improved P_RRT*

algorithm. *Complex & Intelligent Systems*, *8*(3), 2227–2245. https://doi.org/10.1007/s40747-021-00628-y

Zhang, L., Wang, Y., Zhao, X., Zhao, P., & He, L. (2021). Time-optimal trajectory planning of serial manipulator based on adaptive cuckoo search algorithm. *Journal of Mechanical Science and Technology*, *35*(7), 3171–3181. https://doi.org/10.1007/s12206-021-0638-5

Zhang, Y., Gong, D., & Zhang, J. (2013). Robot path planning in uncertain environment using multi-objective particle swarm optimization. *Neurocomputing*, *103*, 172–185. https://doi.org/10.1016/j.neucom.2012.09.019

Zucker, M., Ratliff, N., Dragan, A. D., Pivtoraiko, M., Klingensmith, M., Dellin, C. M., Bagnell, J. A., & Srinivasa, S. S. (2013). CHOMP: Covariant Hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, *32*(9–10), 1164–1193. https://doi.org/10.1177/0278364913488805

Appendix A. PSEUDO CODE OF PROPOSED METHODE

START

1. **Initialize Quaternions Representation**
2. **Generate Linear-SLERP Trajectory**
3. **Generate B-Spline Trajectory**
4. **Generate Bézier Trajectory**
5. **Integrate Trajectories**
6. **Simulate in CoppeliaSim**
7. **Optimize Weights and Control Points**
8. **Main Program:**

— *initial_orientation, target_orientation = getInitialAndTargetOrientations()*
— *start_point, end_point = getStartAndEndPoints()*
— *control_points = getControlPoints()*

— *quaternion_initial, quaternion_target = initializeQuaternions(initial_orientation, target_orientation)*

— *linear_path, slerp_orientations = generateLinearSLERP(start_point, end_point, quaternion_initial, quaternion_target)*
— *b_spline_path = generateBSpline(control_points)*
— *Bézier _path = generateBézier (control_points)*

— *initial_weights = [0.25, 0.20, 0.20, 0.35]*
— *combined_path = integrateTrajectories(linear_path, b_spline_path, Bézier _path, initial_weights)*

— *simulation_parameters = getSimulationParameters()*
— *simulation_results = simulateInCoppeliaSim(combined_path, simulation_parameters)*

— *optimized_weights, optimized_control_points = optimizeWeightsAndControlPoints(simulation_results, initial_weights, control_points, maxIterations)*
— *return optimized_weights, optimized control_points*

END