VOL. 23, NO. 2, 2024, 81-91
elektrika.utm.my
ISSN 0128-4428

ELEKTRIKA
Journal of Electrical Engineering

Polishing Machine Control System Based on B-Spline Curve Trajectory

Qitao Tan[1,2], **Mohd Ariffanan Mohd Basri**[1*], **Jianbin Wang**[2]

[1]Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310 Johor Bahru, Johor, Malaysia
[2]Faculty of Computer Science and Software, School of Big Data, Zhaoqing University, 55 Zhaoqing Avenue, Duanzhou, Zhaoqing, Guangdong, China

*Correspondence: ariffanan@fke.utm.my

Abstract: In this study, a B-spline trajectory control algorithm is presented, showcasing its ability to achieve smooth trajectory control within an embedded motion control system. Traditional surface machining methods involve converting interpolated trajectories into G-code using numerical control (NC) software, which is then downloaded and executed on the numerical control system. This research introduces a specialized third-degree B-spline interpolation method tailored for curved surface trajectories, aiming to enhance machining efficiency. A significant strength of this project lies in the holistic design of an embedded system, encompassing both hardware circuitry and software logic. Utilizing the cost-effective STM32 architecture, a B-spline discrete velocity interpolation algorithm is implemented, validated through experiments conducted on a polishing machine system. The project involves MATLAB software for data simulations and experiments on a polishing machine using B-spline curve interpolation, confirming the practicality and effectiveness of the third-degree B-spline algorithm within an embedded system.

Keywords: B-spline curve interpolation; motion control; embedded system

Article History: received 24 January 2024; accepted 16 June 2024; published 29 August 2024

1. INTRODUCTION

Surface machining is a broad category in industrial automation, and many manufacturing sectors currently utilize computers or logic controllers for surface processing, incurring high implementation costs[1] [2] [3]. Conventional computer numerical control (CNC) systems for machining free-form surfaces require pre-processing through computer aided design(CAD) and computer-aided manufacturing (CAM) software [4] [5]. The trajectory of the free-form surface is generated by CAM software, converted into G-code, and then imported into the CNC system. However, this method has several drawbacks: 1.Prior to machining, CAM software must be used to convert the motion trajectory, and the accuracy of the machining trajectory depends on the performance of the CAM software [6]. 2.Numerous pre-processing steps are involved, making it challenging to adjust the trajectory during operation. 3.It is categorized as offline programming, and once the execution process begins, the trajectory cannot be modified.

B-spline curves provide a means to achieve surface processing, allowing multi-axis CNC equipment to follow continuous trajectories by setting control points and feed speeds. Control points, which can also be referred to as shape points, can influence the generated curve trajectory. They find extensive applications in specific machining scenarios, such as cases involving the surface polishing of metal vessels [7]. The use of B-spline curves replaces traditional semi-automatic manual labor, enabling the automation of continuous polishing for curved metal vessels, thereby enhancing efficiency and quality [8] [9]. In comparison to other works, this report stands out by proposing a comprehensive embedded system solution[10], encompassing hardware diagram design and software program design. It implements a B-spline discrete velocity interpolation algorithm using the cost-effective STM32 chip, validated on a CNC polishing device. The control board of this system has been successfully applied in industrial settings.

Extensive research has been conducted by numerous scholars focusing on the theoretical research on B-spline curves. In the study by Shuai Ji [11], a non-uniform rational B-spline (NURBS) curve was introduced, along with a forecasting model derived from Newton's interpolating polynomial. The foundation of this equation lies in the interpolated calculation of the correlation between chord length and arc length. The target parameter "u" for each interpolation cycle is determined through Taylor's expansion. Xu Dua's [12] research demonstrated that utilizing the bi-chord error test leads to a substantial decrease in the quantity of control points. In the autonomous adjustment process of both the quantity and positions of control points for the active spline curve, Huaiping Yang [13] incorporated squared distance minimization (SDM). Jean Marie Langeron [14] employed B-spline curves within CAM software to create a tool path. This research focused on the geometric calculation of the tool path. These investigations primarily focused on foundational research, encompassing the analysis of the method for estimating velocity, algorithm based on Taylor interpolation, and various aspects related to spline curves.

Mohammad Mahdi Emami [15] proposed a trajectory prediction generation method. By computing the estimated distance between two trajectory points and performing reverse interpolation within the interpolation cycle, the deceleration moment is determined. In the research conducted by Xianbing Liu [16], a module for lookahead angle prediction was developed, allowing adaptive adjustments to feed rates within a small turning radius range. Simplifying the accelerate (ACC) and decelerate (DEC) algorithms, as proposed by Hepeng Ni [17], ensured compensation for rounding errors. Consequently ensuring the smoothness of the jerk profile. Daoshan Du's [18] study focused on mitigating sudden acceleration/deceleration changes in regions with high curvature. In the proposal by Shingo Tajima [19] , a new interpolation algorithm was suggested, enabling real-time trajectory interpolation within defined intervals and with minimal computational cost. Marco Riboli [20] present a motion planning method for dual-arm Cartesian robots, utilizing fifth-degree B-splines and quadratic programming to minimize trajectory curvature and ensure smooth, collision-free paths. The method also optimizes motion profiles through iso-parametric trajectory planning, demonstrating its application in a laser machine sorting system. Guangwen Yan [21] develop a CNC machining corner rounding technique using asymmetrical B-spline curves to independently adjust transition lengths, effectively minimizing curvature and enhancing feedrate by eliminating overlaps. This method results in smoother and faster machining processes, as demonstrated through extensive simulations and practical experiments. Fengcai Huo [22] propose a smooth path planning method for Ackermann mobile robots using an improved ant colony algorithm and B-spline curves. This approach integrates multi-objective optimization to handle path length and smoothness constraints, incorporates a refined ant colony algorithm with enhanced pheromone updates, and employs B-spline curve adjustments to comply with kinematic limits, significantly improving path efficiency and adherence to curvature constraints. Xiangfei Li [23] developed a Cartesian trajectory planning method for industrial robots utilizing triple NURBS curves to synchronously describe and plan the robot's position and orientation trajectories while ensuring limited linear jerk and continuous bounded angular velocity, even in the absence of an optimization process. Keith Ng [24] introduce a deflection-limited trajectory planning method for robotic milling that employs B-spline curves to ensure smooth and precise curvilinear paths. By integrating real-time feedback and adjusting feed rates based on calculated deflections, this approach enhances machining accuracy and efficiency. In their study, Kang Min [25] and colleagues introduce an innovative force control architecture for robotic abrasive belt grinding of complex curved blades, which integrates smooth trajectories using cubic B-spline and C2 continuous quaternion spline curves. This method significantly improves the grinding process by ensuring continuous and smooth movement, thus enhancing the surface quality of the blades. Hexiong Li [26] developed a Cartesian trajectory planning method for industrial robots using triple NURBS curves, enhancing motion precision by controlling linear jerk and bounded angular velocity, without optimization procedures. Shengqian Li and Xiaofan Zhang [27] introduce a trajectory planning method for underwater vision welding robots using quintic B-spline curves. This method interpolates joint angles to ensure smooth transitions and utilizes sequential quadratic programming to optimize trajectory time, significantly enhancing movement accuracy and efficiency. Yuanjian Lv [28] developed an adaptive trajectory planning algorithm for robotic belt grinding, utilizing a material removal profile model and NURBS curves. This approach dynamically adjusts the grinding paths based on changes in material curvature and elastic deformation, enhancing the grinding efficiency and accuracy as demonstrated through simulations and experiments.

Drawing on the investigations conducted by the aforementioned researchers, this paper conducts unique research, primarily focusing on the foundational theory of B-spline curves and the expansion analysis of constant velocity control.

The contributions of this work encompass the following aspects:

1. Analysis of the B-spline control algorithm, employing control points for the generation of smooth trajectory running curves.
2. Study the constant velocity control strategy for trajectories and validate it through simulation and experimentation.
3. Developed an embedded motion control system, including hardware schematic design, circuit design, software architecture design, algorithm development, and debugging. The system was validated in a physical polishing machine, demonstrating high efficiency and stability.

2. SYSTEM DESIGN

2.1 Principle of B-spline Curve Interpolation

The order of B-spline curves determines the smoothness of the curve, while the degree determines the number of basis functions. Adjusting these parameters allows control over the complexity and smoothness of the curve. This project investigates third-order B-spline curves, and the mathematical derivation of a third-order B-spline curve is as follows:

$$P_i(u) = \sum_{j=0}^{3} B_{j,4}(u)\, P_{i+j} = \frac{1}{6}\begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} P_i \\ P_{i+1} \\ P_{i+2} \\ P_{i+3} \end{bmatrix} \quad (1)$$

The control point P_i exhibits multiple spline feature deformations, where *i* denotes the control point sequence number. The basis function of the cubic spline, denoted as

$B_{j,4}$, is associated with the basis function sequence number *j*. The expansion of $B_{j,4}$ is articulated as follows:

$$B_{j,4}(u) = \begin{cases} B_{0,4}(u) = \frac{1}{6}(1-u)^3 \\ B_{1,4}(u) = \frac{1}{6}(4-6u^2+3u^3) \\ B_{2,4}(u) = \frac{1}{6}(1+3u+3u^2-3u^3) \\ B_{3,4}(u) = \frac{1}{6}u^3 \end{cases} \tag{2}$$

Given the need to apply B-spline curves in real-time embedded systems in this study, it is essential to ensure the system possesses efficient real-time performance. Therefore, the calculation method for B-spline curves must be discretized. Assuming the interpolation period is T_s, the control parameter *u* must be computed within the interpolation period through a simple calculation, as all computations for a third-degree B-spline curve are functions of the parameter *u*, making direct calculation feasible.

The interpolation algorithm for this project is as follows: within each interpolation period T_s, establish a constant-speed increment Δu_k for the parameter, then, using a discrete interpolation formula, calculate the next velocity increment Δu_{k+1}. The function expression for the third-degree B-spline curve associated with control point number *i* can be derived based on the parameter *u*.

$$P(u) = R_3u^3 + R_2u^2 + R_1u + R_0 \tag{3}$$

The parameters R_0, R_1, R_2 and R_3 are defined as follows:

$$\begin{cases} R_3 = (1/6)(-P_i + 3P_{i+1} - 3P_{i+2} + P_{i+3}) \\ R_2 = (1/6)(3P_i - 6P_{i+1} + 3P_{i+2}) \\ R_1 = (1/6)(-3P_i + 3P_{i+2}) \\ R_0 = (1/6)(P_i + 4P_{i+1} + P_{i+2}) \end{cases} \tag{4}$$

To maintain constant velocity in spline interpolation, it is necessary to determine the contour step within a single interpolation cycle based on the given speed and map it into the parameter space to obtain the parameter increment Δu_k . Substituting Δu_k into the spline curve for interpolation calculations allows the computation of the control points $P(u_k)$ for each T_s period.. The calculation for Δu in each constant velocity interpolation cycle is as follows:

$$\Delta u_k = \frac{V \cdot T_s}{\left| \frac{dP_{i(u)}}{du} \right|_{u=u_k}} \tag{5}$$

As the parameter increment *u* is determined by the given speed and interpolation cycle, the chord length between the trajectory space points generated in each cycle remains constant, ensuring a constant feed rate throughout the entire trajectory.

2.2 B-Spline interpolation process

The trajectory of the B-spline curve is fitted through control points $P_i(x, y)$, and the choice of control points influences the overall shape of the curve fit. According to the derived formula, *k* is a variable controlling the accuracy of the curve, and altering this variable will adjust the degree of fitting of the spline curve. The smaller the value of *k*, the higher the fitting accuracy.

The B-spline curve control in this project is fitted using a discrete approach. Before conducting the curve fitting, it is necessary to determine whether it is an open or closed curve. If it is a closed curve, the first two control points are sequentially added after the control sequence. If it is an open curve, the first and last two endpoints are processed and each control point is added, denoted as begin and end. The midpoint between begin and the original first control point is the original second control point, and the midpoint between end and the original penultimate last control point is the original penultimate control point. B-spline interpolation flow chart is shown in Figure 1.

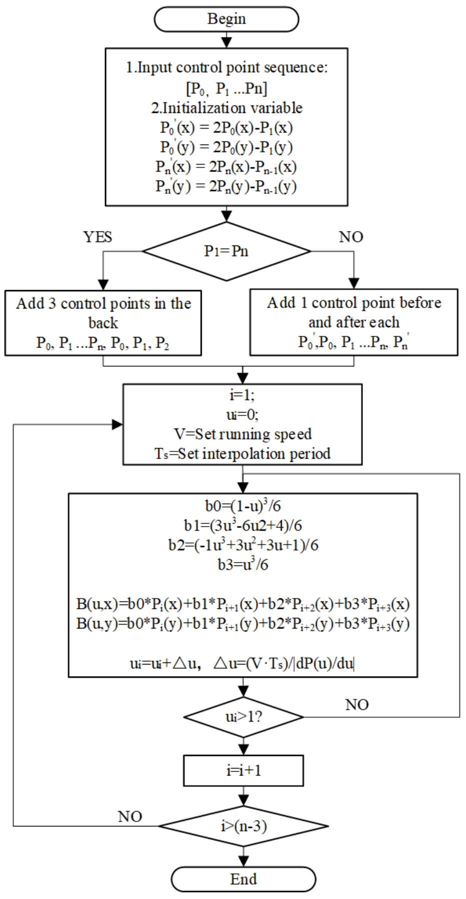
Begin
1.Input control point sequence:
[P0, P1 ...Pn]
2.Initialization variable
P0'(x) = 2P0(x)-P1(x)
P0'(y) = 2P0(y)-P1(y)
Pn'(x) = 2Pn(x)-Pn-1(x)
Pn'(y) = 2Pn(y)-Pn-1(y)
YES
P1=Pn
NO
Add 3 control points in the back
P0, P1 ...Pn, P0, P1, P2
Add 1 control point before and after each
P0',P0, P1 ...Pn, Pn'
i=1;
ui=0;
V=Set running speed
Ts=Set interpolation period
b0=(1-u)^3/6
b1=(3u^3-6u2+4)/6
b2=(-1u^3+3u^2+3u+1)/6
b3=u^3/6
B(u,x)=b0*Pi(x)+b1*Pi+1(x)+b2*Pi+2(x)+b3*Pi+3(x)
B(u,y)=b0*Pi(y)+b1*Pi+1(y)+b2*Pi+2(y)+b3*Pi+3(y)
ui=ui+△u, △u=(V·Ts)/|dP(u)/du|
ui>1?
NO
i=i+1
NO
i>(n-3)
End

Figure 1. B-spline curve interpolation flow chart

2.3 Velocity control strategy

B-spline trajectories are divided into *i* segments based on control points, and different running velocity can be set for each trajectory segment[29]. In this project, velocity variations are achieved using S-shaped curves for velocity rate regulation. The use of S-shaped velocity control enhances the overall smoothness and fluidity of the entire B-spline trajectory. The formula for implementing the S-curve is as follows:

$$s(t) = \begin{cases} v_s\tau_1 + J\tau_1^3/6 & 0 \le t < t_1 \\ s_{01} + v_{01}\tau_2 + JT_1\tau_2^2/2 & t_1 \le t < t_2 \\ s_{02} + v_{02}\tau_3 + JT_1\tau_3^2/2 - J\tau_3^3/6 & t_2 \le t < t_3 \\ s_{03} + v_{03}\tau_4 & t_3 \le t < t_4 \\ s_{04} + v_{04}\tau_5 - J\tau_5^3/6 & t_4 \le t < t_5 \\ s_{05} + v_{05}\tau_6 - JT_5\tau_6^2/2 & t_5 \le t < t_6 \\ s_{06} + v_{06}\tau_7 + JT_5\tau_7^2/2 + J\tau_7^3/6 & t_6 \le t < t_7 \end{cases} \quad (6)$$

S(t) describes the entire acceleration-deceleration process divided into 7 transition stages, where the number of stages is represented by n, and n = 1, 2 ... 7. t_n represents the transition moments for each speed change stage, τ represents the local time coordinate when each stage's starting point is taken as the zero point, and T_n represents the duration of each stage's operation. Assuming the number of control points for the B-spline curve is *i*, the number of segments in the trajectory is *k*, namely *k* = *i*-1. The S-curve acceleration-deceleration algorithm is used to achieve speed transitions for each trajectory segment, as shown in Figure 2.

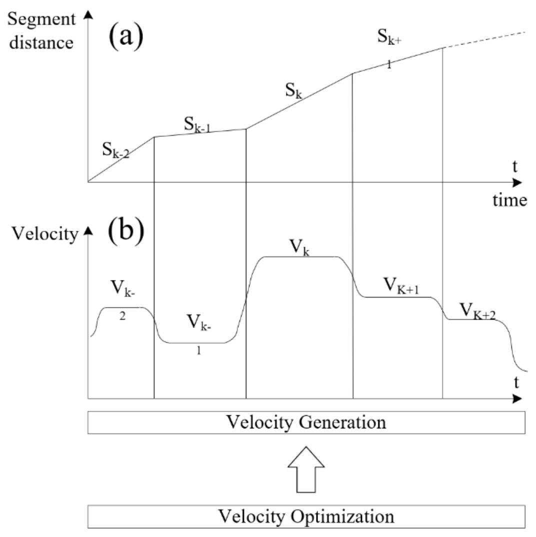

Figure 2. Velocity planning diagram

2.4 Embedded system design

2.4.1 Hardware system design

Based on our assessment of different embedded systems, employing Digital Signal Processor(DSP) for system design guarantees high algorithm operation efficiency but comes with a high implementation cost. If an Field Programmable Gate Array(FPGA) is utilized for system design, achieving algorithm logic design becomes challenging, and the system may experience limited scalability [30, 31]. In order to have a cost advantage in design and ensure a high cost-performance ratio, this study adopts a CPU from the Advanced RISC Machine (ARM) system as the core control chip[32], with the specific model being STM32F407ZGT6.

We selected a Cortex-M4 ARM-based CPU as the main chip, known for its high performance and low power consumption. This chip is widely used in embedded system development, offering robust functionality and flexible configuration options. The peripheral devices in our system design include PWM control, digital input/output, communication interfaces, power output, and other features. The specific hardware design block diagram is illustrated in Figure 3. Figure 4 depicts the hardware circuit board.

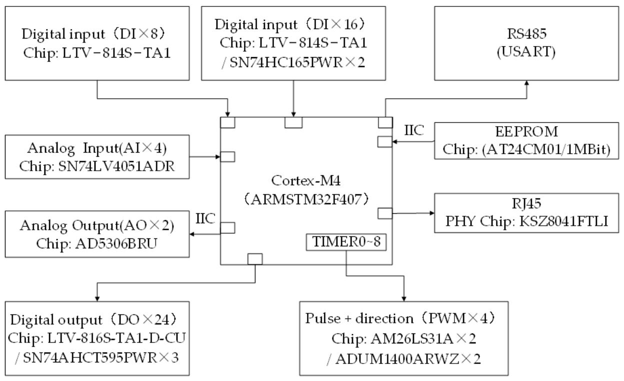
Digital input（DI×8）
Chip: LTV-814S-TA1
Digital input（DI×16）
Chip: LTV-814S-TA1 / SN74HC165PWR×2
RS485 (USART)
IIC
EEPROM Chip: (AT24CM01/1MBit)
Analog Input(AI×4) Chip: SN74LV4051ADR
Cortex-M4（ARMSTM32F407）
RJ45 PHY Chip: KSZ8041FTLI
IIC
Analog Output(AO×2) Chip: AD5306BRU
TIMER0~8
Digital output（DO×24）
Chip: LTV-816S-TA1-D-CU / SN74AHCT595PWR×3
Pulse + direction（PWM×4）
Chip: AM26LS31A×2 / ADUM1400ARWZ×2

Figure 3. Hardware architecture schematic

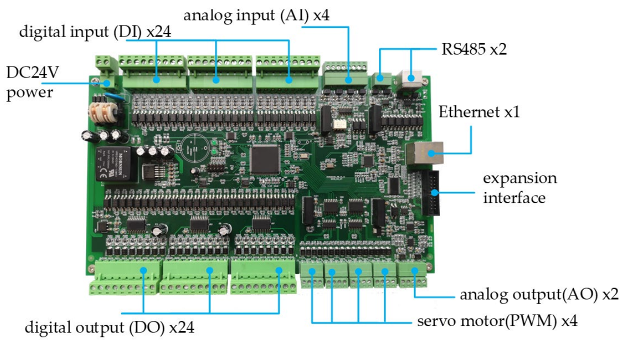

Figure 4. Hardware circuit board

2.4.2 Software system design

The hardware architecture of this project is designed using the STM32 embedded system. The software program needs to be implemented on the STM32 chip. Due to the high real-time requirements of embedded systems and the fact that STM32 is a compact single-threaded chip, there are stringent demands on the software framework design. To achieve high real-time multi-axis interpolation motion control, a FIFO queue is utilized in the software architecture. The software implementation framework is described as follows:

1. Input control points P.
2. A discrete algorithm for B-spline curves, as depicted in Figure 1, is designed. The interpolation period is set to T_s=1ms, meaning the computation is performed every T_s period.
3. A discrete trajectory generation program is designed to calculate the control parameters of the spline curve: R_0, R_1, R_2, R_3.
4. An S-curve velocity program is designed to calculate

the displacement adjustment quantity Δu by setting the velocity variable V.

5. A servo drive function is designed. Due to the real-time requirements of embedded systems, the program architecture employs an advanced First-In-First-Out (FIFO) queue for management. In the fourth step, each servo displacement adjustment quantity Δ_X, Δ_Y, Δ_Z is generated into the queue. Through queue operations, the output pulses ΔP_x, ΔP_y, ΔP_z for each axis are generated in each T_s period.
6. Output to the servo system for execution, and the software framework along with the implementation process are depicted in Figure 5.

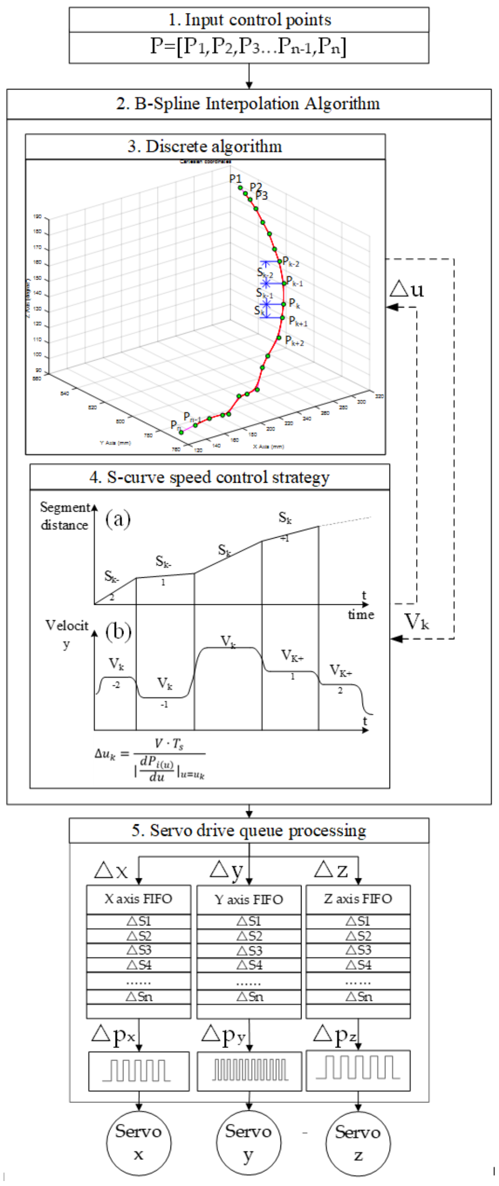

Figure 5. Software Framework and Implementation Process

3. SIMULATION

This study aims to verify the real-time applicability of the third-degree B-spline curve in embedded systems. MATLAB is employed as the analytical tool for theoretical simulations, validating the discrete algorithm of the B-spline curve. The simulation consists of two experiments: first, the simulation of a two-dimensional B-spline curve in the x-y coordinate plane, followed by the simulation of a three-dimensional B-spline curve in the x-y-z Cartesian coordinate plane. During the simulation, the trajectory's operating speed is set at 100 mm/s, and the interpolation period T_s is configured as 1ms. Table 1 presents the parameters for the operation of the two-dimensional B-spline curve, and Figure 6 illustrates its simulated operation. Table 2 provides the parameters for the operation of the three-dimensional B-spline curve, and Figure 7 depicts its simulated operation.

Table 1. B-spline curve parameter configuration

Content	Parameters
Control points	19
Axis numbers	2
x-axis coordinate	(0 20 25 40 55 80 90 100 120 150 165 180 185 190 205 220 247 268 295)
y-axis coordinate	(0 10 15 30 45 50 80 88 90 95 100 102 103 107 108 110 130 145 150)

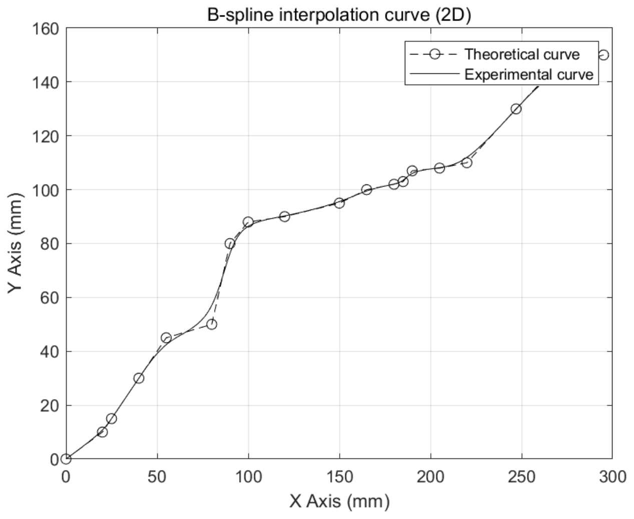

Figure 6. B-spline curve Simulation (2D)

From the simulated curves, it can be observed that the discrete algorithm and process of the B-spline curve effectively track the specified control points. This enables the trajectory to follow the planned path along the control points. Moreover, at the turning positions of the control points, it achieves a smooth transition.

Table 2. B-spline curve parameter configuration

Content	Parameters
Control points	13
Axis numbers	3
x-axis coordinate	(1.5 7.17 9.99 6.75 -0.58 -7.56 -9.96 -6.31 1.16 7.93 9.89 5.84 -1.74)
y-axis coordinate	(10 6.96 -0.29 -7.37 -9.98 -6.53 0.87 7.75 9.93 6.08 -1.45 -8.11 -9.84)
z-axis coordinate	(0.5 0.8 1.6 2.4 3.2 4.0 4.8 5.6 6.4 7.2 8.0 8.8 9.6)

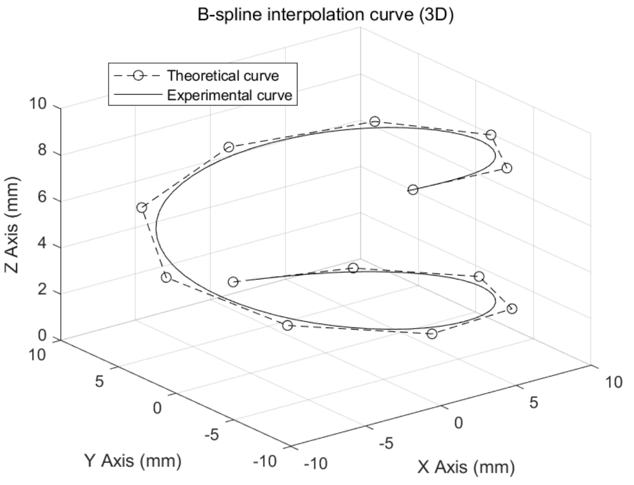

Figure 7. B-spline curve Simulation (3D)

For traditional polishing machines, trajectory programming is typically accomplished using CNC control systems and G-code. For this project, which requires complex surface machining, writing G-code necessitates the use of CAD/CAM software, a standard practice in conventional CNC programming. CAD/CAM software plays a crucial role in precisely mapping the machining paths, allowing programmers to customize and optimize the trajectory in detail to meet specific processing requirements. Using G-code in conjunction with CAD/CAM software not only ensures that the machine can perform high-precision and complex polishing tasks but also enables accurate trajectory interpolation. The interpolation points for the polishing trajectory of this project are generated using MATLAB software and are displayed in Figure 8.

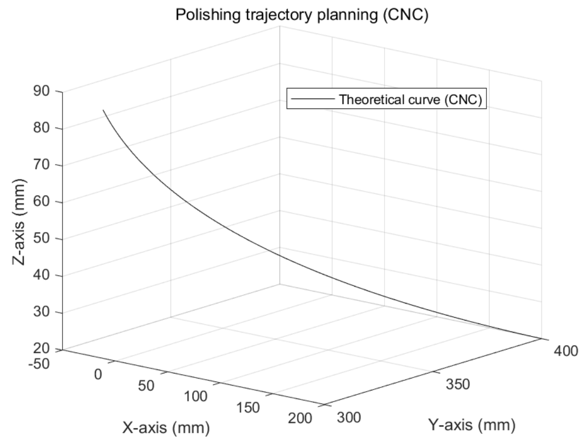

Figure 8. Polishing trajectory planning (CNC)

Taking 13 points on the CNC trajectory curve, these points are used as control points to generate a B-spline trajectory curve. The comparative plot between the generated B-spline curve and the CNC trajectory curve is depicted in Figure 9. The trajectories and trajectory error curves for the x-axis, y-axis, and z-axis are shown in Figure 10, 11 and 12, respectively.

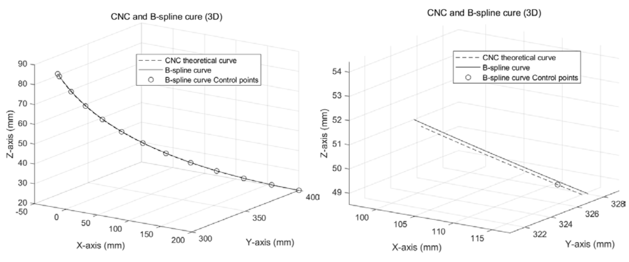

Figure 9. B-spline curve and the CNC trajectory curve (13 points)

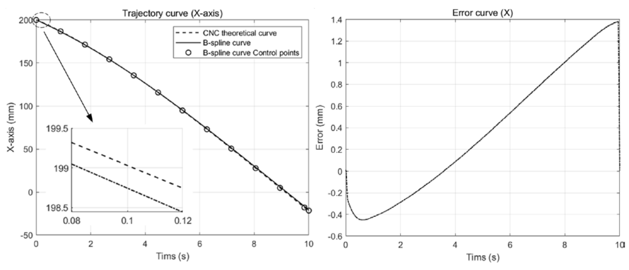

Figure 10. x-axis trajectory and error curve (13 points)

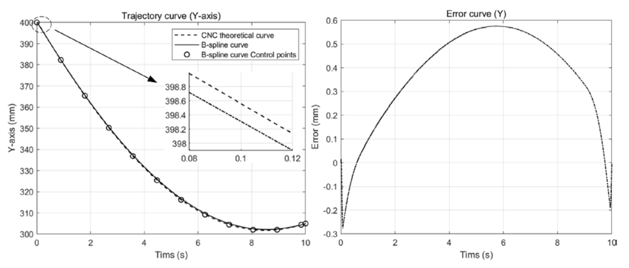

Figure 11. y-axis trajectory and error curve (13 points)

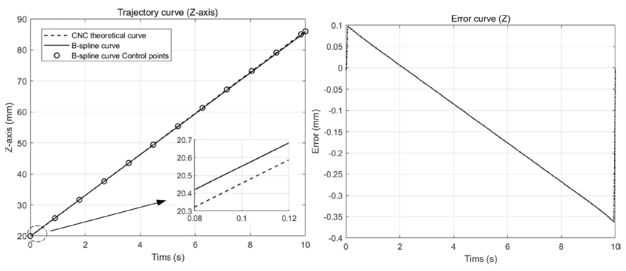

Figure 12. z-axis trajectory and error curve (13 points)

Selecting 29 points from the CNC trajectory curve as control points generated a new B-spline trajectory curve. The comparative plot between the new B-spline curve and the CNC trajectory curve is depicted in Figure 13. The trajectories and trajectory error curves for each axis are shown in Figure 14, 15 and 16, respectively.

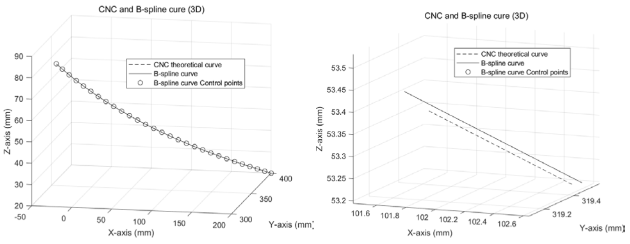

Figure 13. B-spline curve and the CNC trajectory curve (29 points)

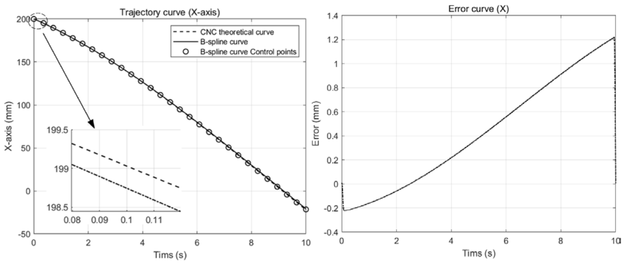

Figure 14. x-axis trajectory and error curve (29 points)

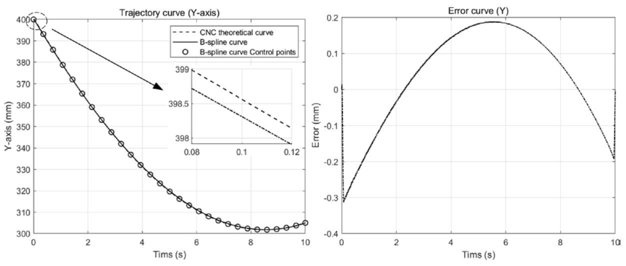

Figure 15. y-axis trajectory and error curve (29 points)

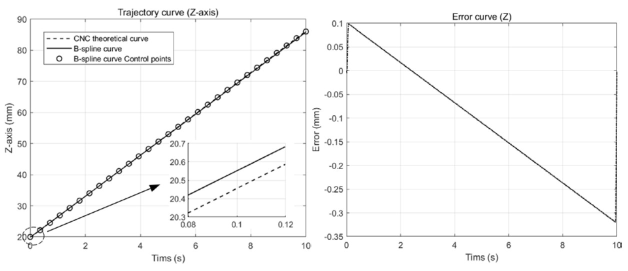

Figure 16. z-axis trajectory and error curve (29 points)

The comparison of the two simulation experiments is depicted in Table 3. From the data comparison, it is evident that the B-spline curve can effectively track the control points, and the curve transitions smoothly and seamlessly, showcasing the high effectiveness of the B-spline control algorithm in this project. In the analysis of the error curves, significant errors are observed in the latter half of the trajectory for each axis. When the B-spline curve is set with 12 control points, the errors are e_x=1.364, e_y =0.582, e_z =0.362. When the B-spline curve is set with 29 control points, the errors are e_x=1.242, e_y =1.191, e_z =0.323. Therefore, it can be observed that the more closely spaced the control points are, the closer the generated trajectory is to the ideal trajectory.

When adjacent control points result in a small turning radius, causing the B-spline curve to deviate from the theoretical curve, it is advisable to add an extra control point at that position to make the trajectory curve closer to the ideal curve.

Table 3. B-spline curve parameter configuration

Content	Simulation 1	Simulation 2
Control points	13	29
Axis numbers	3	3
x-axis coordinate	(200 187 171 154 136 116 95 73 51 28 5 -18 -21)	(200 195 190 184 178 171 165 158 151 143 136 128 120 112 103 95 86 77 69 60 51 41 32 23 14 5 -4 -13 -21)
y-axis coordinate	(400 382 365 350 337 325 316 309 304 302 302 304 305)	(400 393 386 379 372 365 359 353 347 342 337 332 328 323 320 316 313 310 308 306 304 303 302 302 302 302 303 304 305)
z-axis coordinate	(20 26 32 38 44 49 55 61 67 73 79 85 86)	(20 22 25 27 29 32 34 36 39 41 44 46 48 51 53 55 58 60 63 65

		67 70 72 74 77 79 82 84 86)
x-axis max error	1.364mm	1.242mm
y-axis max error	0.582mm	0.191mm
z-axis max error	0.362°	0.323°

4. EXPERIMENT

The polishing machine utilized in this research comprises three axes: x-axis, y-axis, and z-axis (rotation axis). These three axes operate as servo motion systems, collectively forming a Cartesian coordinate motion system in spatial linkage. The tool center point of the coordinate system's movement corresponds to the installation position of the polishing workpiece. In other words, the workpiece can undergo horizontal and rotational movements within the Cartesian space, while the grinding wheel remains fixed on the positioning axis. Specifically, the wax servo device is employed to apply polishing wax to the grinding wheel during the polishing process. This ensures that the metal vessels do not overheat or become discolored during the polishing operation. As shown in Figure 17.

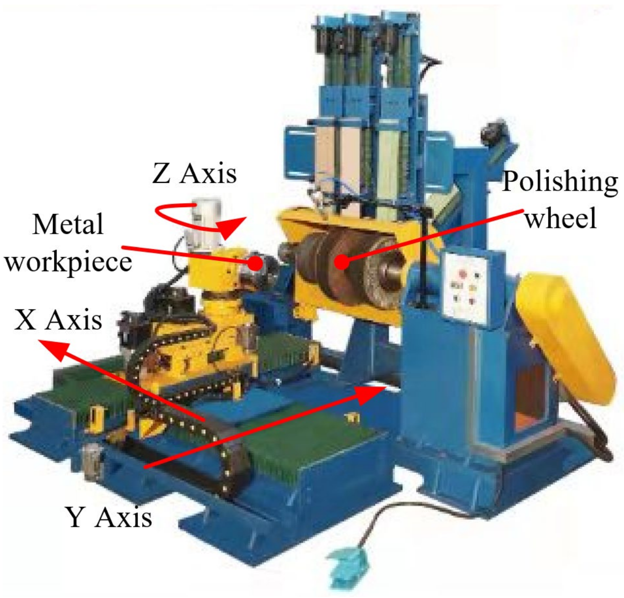

Figure 17. The polishing machine of this project

Figure 18 depicts the schematic diagram of the polishing machine. The x and y axes are capable of spatial movement in a two-dimensional plane, while the Z-axis serves as a rotational axis. The workpiece is mounted on the Z-axis, allowing rotational movement. During the polishing process, it is essential to ensure that the workpiece surface comes into contact with the grinding wheel surface, enabling the polishing of curved surfaces.

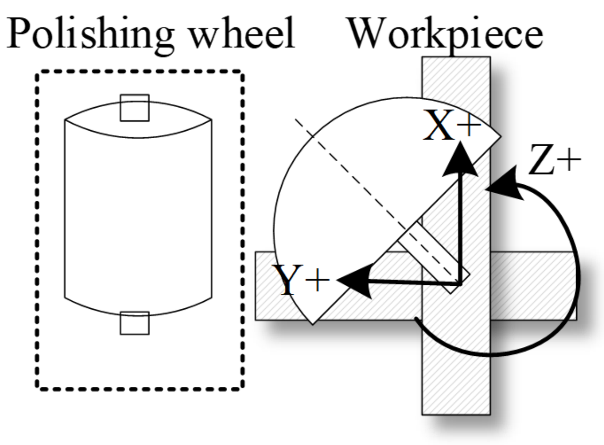

Figure 18. Structure diagram of polishing machine

The system control block diagram is presented in Figure 19. Employing a self-developed embedded board, the controller utilizes the STM32 as the primary control chip within the ARM structure system. For servo control, a pulse-direction differential drive output format is employed, individually linking with the x-axis, y-axis, z-axis, and multiple wax servo axes. The control board card generates a 0~10V analog signal to regulate the frequency converter, driving the workpiece motor based on the prescribed speed. Additionally, the control board oversees the grinding wheel motor, and a current transformer is positioned on the input cable of the grinding wheel motor. The real-time measured current value is dynamically fed back to the control system and serves as a reference for control.

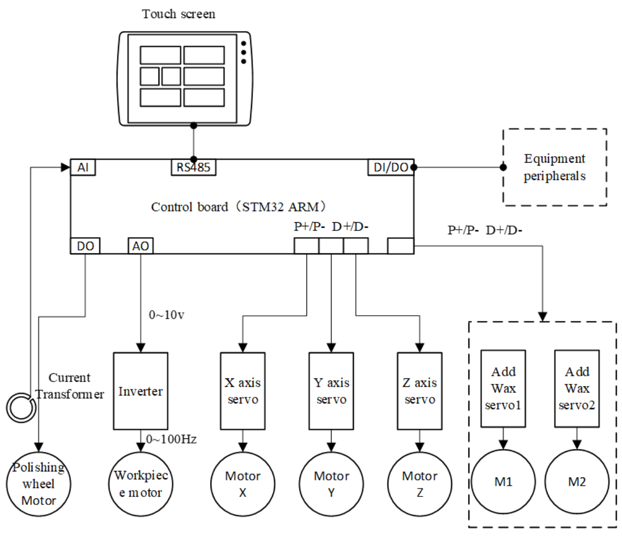

Figure 19. System logic block diagram

Figure 20 provides a schematic representation of the on-site teaching procedure for polishing trajectory, illustrating the relative motion between the polishing wheel and the workpiece surface, along with the step-by-step establishment of trajectory points. The horizontal axes, denoted as x and y, serve as coordinates, while the z-axis functions as the rotational axis, where its value signifies the swing angle of the workpiece. The recorded data during this process [P_1, P_2, P_3, ..., P_{k-1}, P_k, P_{k+1}, ..., P_{n-1}, P_n] represent the actual polishing points, serving as the control points for the B-spline algorithm.

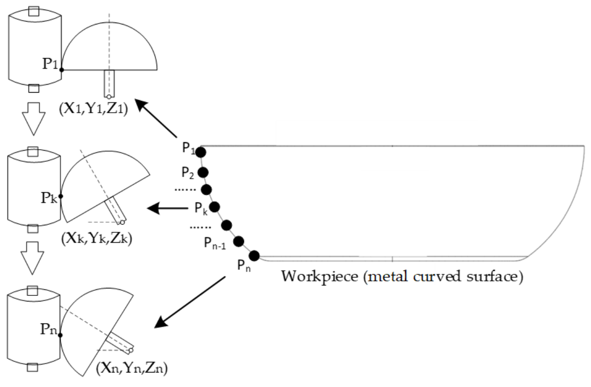

Figure 20. Control point acquisition diagram

The experimental process of this project involves initially obtaining the control points for the B-spline trajectory through on-site teaching and configuring the parameters on the touchscreen. The operating speed of the system in this experiment is set to 100mm/s, with a system interpolation period of 1ms. Specific execution parameters can be found in Table 4.

Upon system startup, the embedded motion controller executes the B-spline interpolation program, driving the servo system with pulse commands. This enables the polishing machine to follow the trajectory based on the teaching points, while concurrently collecting real-time position and encoder feedback information from the drives. The B-spline curve trajectory of this experiment is presented in Figure 21. The real-time position information and error data for the x-axis, y-axis, and z-axis are illustrated in Figures 22, 23, and 24, respectively.

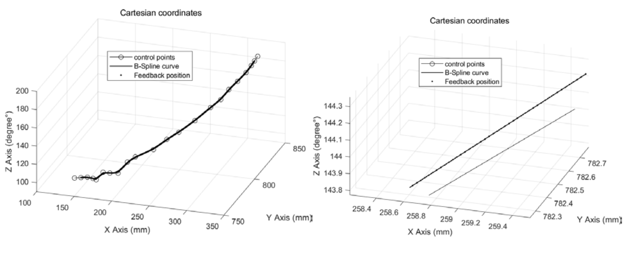

Figure 21. B-spline curve and the CNC trajectory curve (20 points)

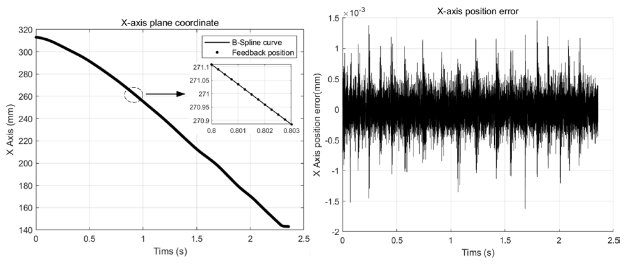

Figure 22. x-axis trajectory and error curve (20 points)

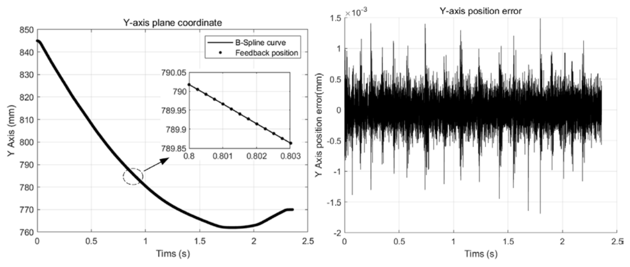

Figure 23. y-axis trajectory and error curve (20 points)

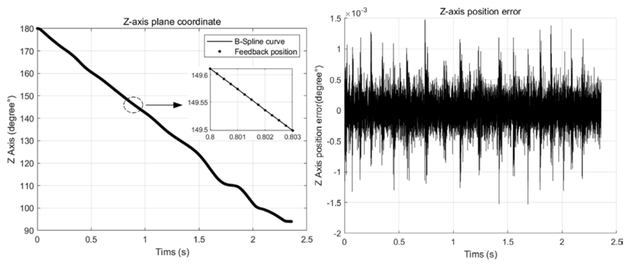

Figure 24. z-axis trajectory and error curve (20 points)

Table 4. B-spline curve parameter configuration

Content	Parameters
Control points	20
Axis numbers	3
x-axis coordinate	(313 312 310 305 300 295 288 276 262 251 238 218 209 199 188 179 169 164 154 143)
y-axis coordinate	(845 841 835 827 819 812 804 793 784 778 772 767 765 762 762 762 763 764 767 770)
z-axis coordinate	(180 178 175 171 168 162 159 152 145 141 133 127 122 111 110 110 100 100 98 94)

In the experiment, we set the running speed to 100 mm/s, with acceleration and deceleration both at 2500 mm/s². The interpolation cycle is 1ms, and the trajectory error update cycle is 0.125ms. Operational effects are illustrated in Figures 22, 23, and 24. Figure 22 displays the x-axis servo's actual trajectory curve, covering a 2.358s run from 180mm to 94mm. The average error for x-axis trajectory is 1.098µm, with a maximum of 1.796µm. Figure 23 showcases the y-axis servo's track curve over 2.358 seconds, ranging from 845mm to 72mm. The average error for track running is 1.284µm, with a maximum of 1.489µm. Figure 24 presents the z-axis servo's trajectory curve, covering a 2.358s run from 180° to 94°. The maximum absolute error for the z-axis is 0.001685°. Figure 22 displays the motion axis's running trajectory in a Cartesian coordinate system, representing the actual system operation. Experimental data are summarized in Table 5.

Table 5. B-spline curve parameter configuration

Content	x-axis	y-axis	z-axis
Average error	1.098μm	1.284μm	0.000876°
Maximum error	1.796μm	1.489μm	0.001685°
Running range	180~94mm	845~72mm	180~94°

In the experiment, the B-spline curve control algorithm was precisely configured and tested for the number of control points, trajectory errors, and system response speeds. Table 4 displays the parameter settings for the B-spline curve in the experimental setup, including the number of control points, coordinates of each axis, as well as the operational speed and interpolation period during the experiment. The purpose of the experiment was to simulate the polishing process through practical operation and to observe how the system executes the predetermined B-spline curve path under real-time conditions.

Table 5 presents detailed error data for the x-axis, y-axis, and z-axis in the experiment, highlighting both average and maximum errors. Specifically, the x-axis exhibited an average error of 1.098 μm and a maximum error of 1.796 μm, the y-axis showed an average error of 1.284 μm with a maximum of 1.489 μm, and the z-axis experienced a maximum absolute error of 0.001685 degrees. These errors were primarily due to the real-time response capabilities of the servo motors and the execution accuracy of the actual equipment. Despite the system achieving high precision in following the B-spline curve, these significant discrepancies across various axes suggest a need for further refinement in the control strategy, aiming to enhance the servo motor responses and reduce execution inaccuracies in the equipment.

Furthermore, the experimental data demonstrated that the system could maintain stable performance at high speeds (100 millimeters per second) and short interpolation periods (1 millisecond). This confirms the applicability of the B-spline curve algorithm in industrial applications requiring high speed and precision. The experiment not only verified the theoretical feasibility of the B-spline algorithm but also showcased its efficiency and reliability in actual industrial applications.

In summary, the combination of experimental results and simulation data clearly demonstrated the advantages of the B-spline curve control algorithm in practical applications, particularly in precisely controlling complex trajectories. This lays a solid foundation for the future application of such algorithms in more high-precision manufacturing scenarios.

5. CONCLUSION

This paper investigates real-time interpolation algorithms for B-spline curves and simulates the algorithms in MATLAB to validate their feasibility. An embedded motion control system based on STM32 is designed, and experiments involving B-spline curve trajectory execution are conducted on a polishing machine, confirming the practical application of the B-spline algorithm in the field of motion control.

Two simulation experiments are performed, involving the generation of B-spline trajectories with 13 and 29 control points, respectively. The results demonstrate the effectiveness of the B-spline curve algorithm, as the trajectory gradually approximates the ideal path with an increasing number of control points. However, in trajectories with very small corner radii of control points, deviations between the actual and ideal trajectories may occur.

The project implements an embedded control system based on STM32 and conducts experiments on a polishing machine. The B-spline curve algorithm is discretized and executed in the embedded system, generating pulse signals to drive servo motors for trajectory planning on the polishing machine. Experimental results show that the B-spline curve can discretely operate in the embedded system, achieving high precision in position calculations. The embedded motion control system operates stably on the polishing machine, enabling the machine to perform specified actions along the given trajectory.

Through simulation and experimentation, this project validates the efficiency of B-spline curves in trajectory control. The concise and clear mathematical description of B-spline curves facilitates their implementation in a computer environment. Their flexibility to adapt to various complex shapes, including curves and surfaces, makes them easily applicable in computational and simulation aspects of industrial production.

ACKNOWLEDGMENT

The authors would like to thank Universiti Teknologi Malaysia under UTMFR (Q.J130000.3823.22H67) for supporting this research.

REFERENCES

[1] Campos, J.G. and L.R. Miguez, Standard process monitoring and traceability programming in collaborative CAD/CAM/CNC manufacturing scenarios. Computers in Industry, 2011. 62(3): p. 311-322.

[2] Dubovska, R., J. Jambor, and J. Majerik, Implementation of CAD/CAM System CATIA V5 in Simulation of CNC Machining Process. Procedia Engineering, 2014. 69: p. 638-645.

[3] Pawan Kumar, N., R. Mangey, and Y. Om Prakash, 7 CNC Machine Programming Codes (G-Codes and M-codes), in Basics of CNC Programming. 2018, River Publishers. p. 73-128.

[4] Xu, X.W., W. Lihui, and R. Yiming, STEP-NC and function blocks for interoperable manufacturing. IEEE Transactions on Automation Science and Engineering, 2006. 3(3): p. 297-308.

[5] Pacheco, N.d.O., Use neutral data interfaces for Numerical Controllers Computerized. IEEE Latin America Transactions, 2017. 15(6): p. 1212-1218.

[6] Cheng, M.Y., M.C. Tsai, and J.C. Kuo, Real-time NURBS command generators for CNC servo

controllers. International Journal of Machine Tools and Manufacture, 2002. 42(7): p. 801-813.

[7] Tsai, M.J., C. Jou-Lung, and H. Jian-Feng, Development of an automatic mold polishing system. IEEE Transactions on Automation Science and Engineering, 2005. 2(4): p. 393-397.

[8] Erwinski, K., et al., Comparison of NURBS trajectory interpolation algorithms for high-speed motion control systems, in 2021 IEEE 19th International Power Electronics and Motion Control Conference (PEMC). 2021. p. 527-533.

[9] Wang, X.D., et al., Global smoothing for five-axis linear paths based on an adaptive NURBS interpolation algorithm. International Journal of Advanced Manufacturing Technology, 2021. 114(7-8): p. 2407-2420.

[10] Fei, J., et al., Research on Embedded CNC Device Based on ARM and FPGA. Procedia Engineering, 2011. 16: p. 818-824.

[11] Ji, S.A., et al., A NURBS curve interpolator with small feedrate fluctuation based on arc length prediction and correction. International Journal of Advanced Manufacturing Technology, 2020. 111(7-8): p. 2095-2104.

[12] Du, X., et al., An error-bounded B-spline curve approximation scheme using dominant points for CNC interpolation of micro-line toolpath. Robotics and Computer-integrated Manufacturing, 2020. 64.

[13] Yang, H.P., W.P. Wang, and J.G. Sun, Control point adjustment for B-spline curve approximation. Computer-aided Design, 2004. 36(7): p. 639-652.

[14] Langeron, J.M., et al., A new format for 5-axis tool path computation, using Bspline curves. Computer-Aided Design, 2004. 36(12): p. 1219-1229.

[15] Emami, M.M. and B. Arezoo, A look-ahead command generator with control over trajectory and chord error for NURBS curve with unknown arc length. Computer-aided design, 2010. 42(7): p. 625-632.

[16] Liu, X.B., et al., Adaptive interpolation scheme for NURBS curves with the integration of machining dynamics. International Journal of Machine Tools & Manufacture, 2005. 45(4-5): p. 433-444.

[17] Ni, H.P., et al., Feedrate Scheduling of NURBS Interpolation Based on a Novel Jerk-Continuous ACC/DEC Algorithm. IEEE Access 2018. 6: p. 66403-66417.

[18] Du, D.S., et al., An accurate adaptive parametric curve interpolator for NURBS curve interpolation. International Journal of Advanced Manufacturing Technology, 2007. 32(9-10): p. 999-1008.

[19] Tajima, S. and B. Sencer, Online interpolation of 5-axis machining toolpaths with global blending. International Journal of Machine Tools and Manufacture, 2022. 175: p. 103862.

[20] Riboli, M., et al., Collision-free and smooth motion planning of dual-arm Cartesian robot based on B-spline representation. Robotics and Autonomous Systems, 2023. **170**: p. 104534.

[21] Yan, G., et al., Asymmetrical transition-based corner rounding method driven by overlap elimination for CNC machining of short-segmented tool path. Journal of Manufacturing Processes, 2022. **76**: p. 624-637.

[22] Yang, J., et al., An analytical tool path smoothing algorithm for robotic machining with the consideration of redundant kinematics. Robotics and Computer-Integrated Manufacturing, 2024. **89**: p. 102768.

[23] Li, X., et al., A novel cartesian trajectory planning method by using triple NURBS curves for industrial robots. Robotics and Computer-Integrated Manufacturing, 2023. **83**: p. 102576.

[24] Ng, K., et al., Deflection-limited trajectory planning in robotic milling. Journal of Manufacturing Processes, 2024. **120**: p. 1180-1191.

[25] Min, K., F. Ni, and H. Liu, Robotic abrasive belt grinding of complex curved blades based on a novel force control architecture integrating smooth trajectories. Journal of Manufacturing Processes, 2023. **107**: p. 447-458.

[26] Li, H., et al., An integrated trajectory smoothing method for lines and arcs mixed toolpath based on motion overlapping strategy. Journal of Manufacturing Processes, 2023. **95**: p. 242-265.

[27] Li, S. and X. Zhang, Research on planning and optimization of trajectory for underwater vision welding robot. Array, 2022. **16**: p. 100253.

[28] Lv, Y., et al., An adaptive trajectory planning algorithm for robotic belt grinding of blade leading and trailing edges based on material removal profile model. Robotics and Computer-Integrated Manufacturing, 2020. **66**: p. 101987.

[29] Heng, M. and K. Erkorkmaz, Design of a NURBS interpolator with minimal feed fluctuation and continuous feed modulation capability. International Journal of Machine Tools & Manufacture, 2010. 50(3): p. 281-293.

[30] Ponce, P., et al., Experimental study for FPGA PID position controller in CNC micro-machines. IFAC-PapersOnLine, 2015. 48(3): p. 2203-2207.

[31] 22. De Santiago-Perez, J.J., et al., FPGA-based hardware CNC interpolator of Bezier, splines, B-splines and NURBS curves for industrial applications. Computers & Industrial Engineering, 2013. 66(4): p. 925-932.

[32] Zhang, H.-f. and W. Kang, Design of the Data Acquisition System Based on STM32. Procedia Computer Science, 2013. 17: p. 222-228.