Available online at www.sciencedirect.com

Manufacturing Letters

Manufacturing Letters 41 (2024) 395–405

52nd SME North American Manufacturing Research Conference (NAMRC 52, 2024)

Jerk Limited Continuous Tool Path Generation for Flexible Systems in Non-Cartesian Coordinate Systems

Mehrdad Sadeghieh[a], Sayyed Mohammadreza Mofidi[b], Ali Hosseini[a], Behnam Moetakef-Imani[b]*

[a] *Machining Research Laboratory,Faculty of Engineering and Applied Science,Ontario Tech University, Oshawa, ON, L1G0C5, Canada*
[b] *CAD/CAM Laboratory, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran*

* Corresponding author. Tel.: +98-51-38 81 5100 ext.:416; fax: +98-51-38 43 6433. *E-mail address:* imani@um.ac.ir

Abstract

Inspired by computer numerical control (CNC) technology and drastic changes in the world of electronics and microcontrollers, the idea of using non-cartesian coordinate system for CNC machines became a topic of interest. Non-cartesian coordinate systems provide freedom of movement in systems with large working spaces where high dimensional accuracy is not a prime concern. Applications of such systems can be found in murals, where a painting is applied to a large wall of any arbitrary build material. In the present paper, a new design was introduced to automate the process of drawing murals on large surfaces. The proposed mechanism utilized chain and sprocket to overcome the size limitations of traditionally available x-y tables. In addition to that, the developed mechanism does not utilize the common robust structures used in the conventional CNC systems and the tool motion is created using flexible arms that are not heavily constrained. The effect of systems unwanted vibration has been eliminated using continuous trajectory and kinematic profiles. Parametric curve interpolation algorithms for trajectory planning were implemented to increase the flexibility of drawing complex curves. Parametric curves such as B-spline and non-uniform rational B-spline (NURBS) were employed to generate a jerk limited trajectory for motion control and extraction of kinematic profiles. Such trajectory constrains the jerk of the system and guarantees a smooth motion free of feed-rate fluctuations. Compared to other methods available for extracting the kinematic profiles, jerk limited method decreases the travel time and trajectory error. Ultimately, motion commands were produced by using kinematic profiles and the second order Taylor interpolator. STM32746ZG card was used as the main processor and two stepper motors controlled by a Bit Pattern interpolation algorithm were utilized to drive the proposed mechanism. The input pulses of stepper motors were stored as binary bits and transmitted to drivers in real-time. The feedback was also evaluated by two rotary encoders, placed at the end of the motors' shafts. Encoder data were acquired and transmitted using a TCP/IP protocol to guarantee zero data loss during the transmission.

Peer-review under responsibility of the scientific committee of the NAMRI/SME.
Keywords: Jerk limited, NURBS, continous toolpath, non-cartesian coordinate systems

1. Introduction

With the advancement of technology, computer numerical control (CNC) machines have made their way into the fields of artistic creation. However, conventional systems are typically confined to the dimensions of their workspace. This limitation prevents the creation of artistic designs on a larger scale such as exterior walls. High costs associated with hardware in larger scales along with software limitations in dealing with curves of higher order, for complex patterns, further contribute to these issues.

The construction of a portable two-dimensional CNC device that can be installed on various surfaces constitutes one of the primary motivations of this research in the hardware domain. The project aims to replace traditional CNC system axes with a more flexible chain and sprocket mechanism, simplifying the system's setup on different platforms. However, using less rigid axes may lead to unwanted vibrations, diminishing the

Peer-review under responsibility of the scientific committee of the NAMRI/SME.

machine's precision in executing predefined trajectory. To address this challenge within the software realm, a continuous jerk-limited trajectory planning algorithm is utilized. This approach effectively reduces system fluctuations, enhancing stability. Additionally, the use of NURBS curves in the design phase was selected for their versatility, offering greater flexibility in the machine's operational capabilities.

Nomenclature

A	Acceleration
A_n	Normal acceleration
A_t	Tangential acceleration
$C(u)$	NURBS curve
$C'(u)$	Derivative of NURBS curve
D	Deceleration
J	Jerk
J_n	Normal Jerk
J_t	Tangential Jerk
$N_{i,p}(u)$	B-spline basis function
P_i	NURBS control points
T_s	Sampling time
U	Knot vector
e_i	Arc length error
f	Feed-rate
p	Degree of the NURBS curve
s_a	Arc length in acceleration section
s_c	Arc length in constant velocity section
s_d	Arc length in deceleration section
s_{cri}^{l}	Arc length criterion for long blocks
s_{cri}^{s}	Arc length criterion for short blocks
u	Curve parameter
u_i	Knot in NURBS curve
${}^{b}v$	Velocity at backward scanning
${}^{f}v$	Velocity at forward scanning
w_i	Weight of NURBS control points
Δe_i	Distributed arclength error
Δs_j	Arclength increment at i^{th} step
$\Delta s_i'$	Modified Arclength increment at i^{th} step
ζ	Tolerance
κ_{cr}	Curvature threshold
$\kappa(u)$	Curvature of NURBS curve

1.1. Previous studies

The pioneering work of Harold Cohen in the development of AARON, an artist robot, stands out as one of the most renowned applications of industrial machineries in creating artistic designs [1]. The initial prototype of AARON was later simplified to put the focus on functionality rather that appearance. Another example of a device capable of artistic drawing was a robot called Paul which was introduced by Tresset and Leymarie [1]. This device was able to draw portraits using image processing. It utilized a camera to capture the desired image and subsequently employed image processing algorithms to transform the captured image into continuous lines. These lines served as the path for the robotic arm to draw the sketch.

Numerous efforts have been made to model profiles of arbitrary shapes using higher order parametric curves. Efforts have also been made to extract kinematic profiles for generating the desired tool trajectory. One of the earliest works in this field was published by Bazaz and Tundo [2] who implemented 3rd order Bezier curves to generate kinematic profiles. In their approach, optimization goal was to achieve the shortest possible time while considering acceleration and velocity constraints. Another work pertaining to this topic was presented by Erkorkmaz and Altintas [3]. They proposed an algorithm for toolpath generation that not only ensures precise trajectory profile creation but also utilizes kinematic profiles to guarantee accurate path tracking. In their approach, a smooth acceleration profile was obtained by imposing constraints on acceleration and jerk. Subsequently, interpolation points which were computed with varying time intervals, were resampled using a fifth order polynomial with a constant sampling time. Another interpolation method using parametric curves was proposed by Sekar et. al. [4]. In this work, a NURBS-based adaptive speed control algorithm for obtaining jerk limited velocity profiles in high-speed machining was presented. The proposed algorithm effectively reduced the speed fluctuations in sharp corners. Parameters curves can also be locally implemented to increase the continuity at the connection point between the non-parametric curves. In this context, Zhao et.al. [5] introduced a rapid interpolation method using predictive control with B-spline curves. In this work, connection points between two curves were linked using a B-spline curve with five control points, ensuring G2 continuity. Considering that calculating the derivatives of parametric curves is normally a computationally expensive task, Li et al. [6] proposed a rapid implicit interpolation method for NURBS curves. The integration interval was divided into several sub-intervals and the inverse length function (ILF) was calculated for each of them. The ILF function was then utilized to calculate the curve parameters eliminating the need for calculating the parametric curve derivatives. NURBS curves were also employed by Du et al. [7] who proposed the complete s-shape feed rate scheduling algorithm (CSFA). In their method, the main path was initially divided into several NURBS sub-curves based on critical points. This step was followed by finding the velocity at each connection point. To enhance velocity profiles, NURBS sub-curves were further subdivided into short, medium, and long blocks based on their lengths. Another work employing NURBS curves was published by Erkorkmaz and Heng [8] where a robust and computationally efficient interpolation process for NURBS curves was introduced. The presented algorithm mitigated unintended speed fluctuations and rounding errors ensuring that final kinematic profiles were uniformly bounded and continuous across all axes.

In the context of NURBS interpolators, Wang et al. [9] proposed a method which focuses on improving machining efficiency and quality in modern manufacturing through advanced parametric interpolation techniques. Specifically, it delved into NURBS interpolation, which offers smoother motion for complex part geometries compared to traditional linear and circular interpolations. The study introduced a feed rate scheduling method that considers both geometric and dynamic constraints to ensure smoother and more efficient

machining. It also presents a double interpolation algorithm based on the cosine theorem to minimize feed rate fluctuation, improving tool motion smoothness and machining accuracy. The effectiveness of their methods was validated through simulations and experiments on typical NURBS curves, demonstrating significant improvements in machining time, surface roughness, and contour accuracy compared to conventional interpolation methods. In 2020, Ji et al. [10] presented a novel interpolation method for NURBS curves, aiming to enhance contour accuracy and dynamic performance in manufacturing systems without heavily increasing computational load. The method comprises three stages: segmenting a NURBS curve into high-order continuous parts, predicting the target arc length for the next cycle using Newton's divided differences based on past chord and arc lengths, and calculating the target parameter for interpolation with Taylor's expansion, iteratively corrected for precise target point positioning. Comparative simulations demonstrate that this approach achieves lower velocity fluctuations with reduced computational demands, offering a balanced solution for efficient and accurate NURBS curve interpolation in manufacturing. Recently, Nie et al. [11] presented a jerk-continuous feed rate optimization method for NURBS interpolation in CNC machining, focusing on improving machining accuracy, surface quality, and efficiency. It introduced a novel approach to segment NURBS curves, classify segments by length, and compute maximum achievable feed rate for each segment. Implementation of this method shows that chord and chord error could be limited to a defined span. They incorporated feed rate scheduling using trigonometric-function-based jerk continuity to enhance motion smoothness and computational efficiency. Simulation results on two curves demonstrated its effectiveness in keeping errors within tolerance and reducing computational and interpolation times, offering better performance compared to existing methods.

1.2. NURBS curves

Given $n+1$ control points $P_0, P_1, \dots, P_n$ and a knot vector $U = \{u_0, u_1, \dots, u_m\}$ containing $m+1$ knots, the mathematical expression of a NURBS curve is defined as follows [12]:

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \quad 0 \le u \le 1 \tag{1}$$

In this context, P_i represents the set of control points that define the control polygon, w_i denotes the weights of the control points, and $N_{i,p}(u)$ are the basis functions of the B-spline curve of degree p. The knot vector of NURBS curves investigated in this study is formulated as follows [12]:

$$U = \{\underbrace{a, \dots, a}_{p+1}, u_{p+1}, \dots, u_{m-p-1}, \underbrace{b, \dots, b}_{p+1}\} \tag{2}$$

Basis function of the curve is defined as follows [12]:

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \le u \le u_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+q} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{4}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+q} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{5}$$

Curvature in NURBS curve is defined as [7]:

$$\kappa(u) = \frac{\left\| C'(u) \times C''(u) \right\|}{\left\| C'(u) \right\|^3} \tag{6}$$

In Eq. (6), $C'(u)$ and $C''(u)$ denote the first and second derivative of the NURBS curves.

In this paper length of the NURBS curve has been calculated using an adaptive quadrature composite Simpson rule. The arc length function in NURBS parametric curves $s(u)$ is defined within the interval $[u_i, u_{i+1}]$ as follows [7]:

$$s(u) = \int_{u_i}^{u_{i+1}} \left\| C'(u) \right\| du \tag{7}$$

Considering the Eq. (7), the first-order derivative of the arc length function can be defined as follows:

$$s'(u) = \left\| C'(u) \right\| > 0 \tag{8}$$

To determine the length of the curve within an arbitrary interval $[u_i, u_{i+1}]$, an adaptive fourth-order method using Simpson's rule is employed, as indicated by the Eq. (9) [7]:

$$s(u_i, u_{i+1}) = \frac{(u_{i+1} - u_i)}{6} \left[s'(u_i) + 4s'\left(\frac{u_{i+1} + u_i}{2}\right) + s'(u_{i+1}) \right] \tag{9}$$

By dividing $[u_i, u_{i+1}]$ into two equal sub-intervals $\left[u_i, \left(\frac{u_i+u_{i+1}}{2}\right)\right]$ and $\left[\left(\frac{u_i+u_{i+1}}{2}\right), u_{i+1}\right]$, a composite Simpson's rule can be applied. In this case, Eq. (9) can be rewritten as:

$$s\left[u_i, \left(\frac{u_i + u_{i+1}}{2}\right), u_{i+1}\right] = s\left[u_i, \left(\frac{u_i + u_{i+1}}{2}\right)\right] + s\left[\left(\frac{u_i + u_{i+1}}{2}\right), u_{i+1}\right] \tag{10}$$

Where:

$$s\left[u_i, \left(\frac{u_i + u_{i+1}}{2}\right), u_{i+1}\right] = s\left[u_i, \left(\frac{u_i + u_{i+1}}{2}\right)\right] + s\left[\left(\frac{u_i + u_{i+1}}{2}\right), u_{i+1}\right] \tag{11}$$

$$s\left[\left(\frac{u_i+u_{i+1}}{2}\right),u_{i+1}\right] = \frac{(u_{i+1}-u_i)}{12}\left[s'\left(\frac{u_i+u_{i+1}}{2}\right) + 4s'\left(\frac{3u_{i+1}+u_i}{4}\right) + s'(u_{i+1})\right]$$

If the condition stated in Eq. (12) is satisfied , the parameter tolerance ζ can be defined over the examined interval as follows [7]:

$$\frac{1}{10}\left[s\left[u_i,\left(\frac{u_i+u_{i+1}}{2}\right),u_{i+1}\right]-s(u_i,u_{i+1})\right]<\zeta \quad (12)$$

It can be proved that:

$$\left|\int_{u_i}^{u_{i+1}} s'(u)du - s\left[u_i,\left(\frac{u_i+u_{i+1}}{2}\right),u_{i+1}\right]\right|<\zeta \quad (13)$$

Therefore, the length of the NURBS parametric curve, considering the given tolerance, is equal to $s\left[u_i,\left(\frac{u_i+u_{i+1}}{2}\right),u_{i+1}\right]$. If Eq. (12) is not satisfied, each of the sub-intervals $\left[u_i,\left(\frac{u_i+u_{i+1}}{2}\right)\right]$ and $\left[\left(\frac{u_i+u_{i+1}}{2}\right),u_{i+1}\right]$ should be further divided into two equal sub-intervals. This process must be continued until the condition stated by Eq. (12) is satisfied. Subsequently, the total length of the divided sub-intervals is calculated using Eq. (10) and the obtained curve length is then used for further interpolation along the path [7].

2. CSFA Algorithm

Process of extracting kinematic profiles for NURBS curves consists of three general steps:

- Preprocessing: in this step, critical points along the toolpath are identified using curvature thresholding. Subsequently, based on these critical points, desired trajectory is divided into multiple blocks.
- Feed rate scheduling algorithm: in this step, curve length of each block is adaptively computed using the Simpson's method. To ensure speed continuity at the connection points of the NURBS blocks, a bidirectional scanning approach is employed, considering constraints on jerk, acceleration, and velocity.
- Interpolation: finally, the obtained speeds from the bidirectional scanning method are utilized in the feed rate scheduling algorithm [7].

2.1. Critical points and curve splitting

Candidate points for curve splitting are points where the curvature exceeds a certain threshold. Among these points, those with a local maximum curvature are referred to as critical points. To find these critical points, the curvature threshold, presented by Eq. (14), must first be determined based on δ, A_n, and J_n.

$$\kappa_{cr} = min\left(\frac{8\delta}{v_{max}T_s+4\delta^2}, \frac{A_n}{V_{max}^2}, \sqrt{\frac{J_n}{v_{max}^3}}\right) \quad (14)$$

In Eq. (14), δ is the chord error, v_{max} is the feedrate, A_n is the normal acceleration, and J_n is the normal jerk. The parameter u of the curve, which represents a point on the curve with a local maximum curvature, is calculated as follows:

$$\kappa'(u) = 0 \quad (15)$$

After obtaining the critical points, the length of the curve between each pair of them is determined using composite Simpson's rule [7].

2.2. Bidirectional Scanning

The goal of bidirectional scanning, involving two consecutive steps namely backward and forward scanning, is to determine velocity values at both ends of each NURBS block, ensuring 2nd order continuity (position, velocity, and acceleration) at transition points.

2.2.1. Backward scanning

Backward scanning initiates from the last NURBS block and concludes at the first. At the start of the backward scanning phase, the final velocity in the last NURBS block is considered to be zero which means the motion is terminated after the last block. The S-shaped motion profile is used to constrain the velocity at the connection points of blocks. During the backward scanning, δ, A_n, J_n, and v_{max} are simultaneously considered. The algorithm for the backward scanning process is as follows, where N is the number of NURBS blocks, ${}^b v_{i+1}$ is the final velocity in the i^{th} NURBS block obtained by backward scanning, and A_t , J_t represent the tangential acceleration and jerk:

I. Initialize $i = N$ and set the final velocity to zero.

II. If $i = 1$, go to step III; otherwise, proceed as follows:

II. a. Perform jerk limited algorithm from final point to the initial point of the block. Taking A_t constraints into consideration, a third-order equation is obtained as follows:

$$({}^b v_i^T)^3 + {}^b v_{i+1}({}^b v_i^T)^2 - ({}^b v_{i+1})^2\,{}^b v_i^T - ({}^b v_{i+1})^3 - J_t s_i^2 = 0 \quad (16)$$

To calculate ${}^b v_i^T$ from Eq. (16), the Newton-Raphson method is used with the starting value ${}^b v_{i+1}$.

If the condition in Eq. (17) holds, the tangential acceleration constraint is satisfied and the algorithm can proceed to step b. Otherwise, if the condition in Eq. (17) is not met, then Eq. (16) can no longer be used for backward velocity calculation. Instead, Eq. (18) must be used.

$${}^b v_i^T \le \frac{A_t^2}{J_t} + {}^b v_{i+1} \quad (17)$$

$$J_t({}^{b}v_i^T)^2 + A_t^2({}^{b}v_i^T) - J_t({}^{b}v_{i+1})^2 + A_t^2({}^{b}v_{i+1}) - 2A_tJ_ts_i = 0 \quad (18)$$

Similar to Eq. (16), Eq. (18) is also solved using the Newton-Raphson method.

II. b. In this step, δ, A_n, and J_n are taken into account. Final value of the speed at backward scanning is calculated as follows:

$${}^{b}v_i = min\left\{ {}^{b}v_i^T, \frac{2}{T_s}\sqrt{\frac{2\delta}{\kappa_i} - \delta^2}, \sqrt{\frac{A_n}{\kappa_i}}, \sqrt[3]{\frac{J_n}{\kappa_i^2}}, v_{max} \right\} \quad (19)$$

II. c. $i = i - 1$ return to step II.

III. Store the velocity values [7].

2.2.2. Forward Scanning

Forward scanning on a NURBS curve begins with the initial block and ends with the final block. The presented algorithm below illustrates the process of forward scanning [9]:

I. Initialize $i = 1$ and set the staring velocity to zero.

II. If $i = N$, proceed to step III; otherwise, perform the following steps:

II. a. Perform jerk limited algorithm from the initial point to the final point of the block. Taking A_t constraints into consideration, a third-order equation is obtained as follows:

$$({}^{f}v_{i+1}^T)^3 + {}^{f}v_i({}^{f}v_{i+1}^T)^2 - ({}^{f}v_i)^2\,{}^{f}v_{i+1}^T - ({}^{f}v_i)^3 - J_ts_i^2 = 0 \quad (20)$$

To obtain ${}^{f}v_{i+1}^T$ in Eq. (20), the Newton-Raphson method is used with a starting value of ${}^{f}v_i$.

If the condition in Eq. (21) holds, the tangential acceleration constraint is satisfied and the algorithm can proceed to step b. Otherwise, if the condition in Eq. (21) is not met, then Eq. (20) is not applicable anymore for forward velocity calculation. Therefore, Eq. (22) must be utilized.

$${}^{f}v_{i+1}^T \le \frac{A_t^2}{J_t} + {}^{f}v_i \quad (21)$$

$$J_t({}^{f}v_{i+1}^T)^2 + A_t^2({}^{f}v_{i+1}^T) - J_t({}^{f}v_i)^2 + A_t^2({}^{f}v_i) - 2A_tJ_ts_i = 0 \quad (22)$$

To find ${}^{f}v_{i+1}^T$, the Newton-Raphson approach is implemented.

II. b. In this step, the forward scanning process calculates the feed-rate at the final point of the block as follows:

$$v_{i+1} = min\{ {}^{b}v_{i+1}, {}^{f}v_{i+1} \} \quad (23)$$

II. c. $i = i + 1$ and return to step II.

III. Store the final velocity values.

Finally, the obtained velocity values are used in feed-rate scheduling algorithm [7].

2.3. Jerk limited feed rate scheduling of blocks

After completing the bidirectional scanning phase, the values $\{s_i, v_s, v_e\}$ (where $v_s = v_i$ and $v_e = v_{i+1}$) are calculated for each of the NURBS blocks to ensure continuous speed at critical points. In feed rate scheduling, the primary task is to generate the S-shaped profile, see Fig. 1, along the NURBS blocks.

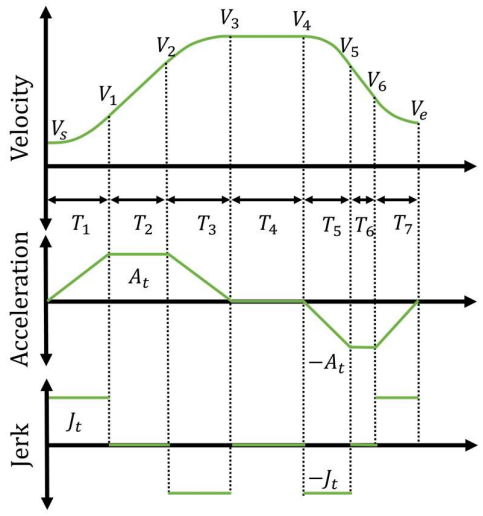

Fig. 1 Jerk limited kinematic profiles [7]

Table 1 shows the kinematic equations governing each section of a seven segment jerk limited motion.

Table 1Jerk limited kinematic equations.

T	$J(t)$	$A(t)$	$v(t)$	$s(t)$
T_1	J_t	J_tT_1	$v_s + \frac{1}{2}J_tT_1^2$	$v_sT_1 + \frac{1}{6}J_tT_1^3$
T_2	0	A_t	$v_1 + A_tT_2$	$s_{a_1} + v_1T_2 + \frac{1}{2}A_tT_2^2$
T_3	$-J_t$	$A_t - J_tT_3$	$v_2 + A_tT_3 - \frac{1}{2}J_tT_3^2$	$s_{a_2} + v_2T_3 + \frac{1}{2}A_tT_3^2 - \frac{1}{6}J_tT_3^3$
T_4	0	0	v_3	$s_a + v_3T_4$
T_5	$-J_t$	$-J_tT_5$	$v_4 - \frac{1}{2}J_tT_5^2$	$s_a + s_c + v_4T_5 - \frac{1}{6}J_tT_5^3$
T_6	0	$-A_t$	$v_5 - A_tT_6$	$s_a + s_c + s_{d_1} + v_5T_6 - \frac{1}{2}A_tT_6^2$
T_7	J_t	$-A_t + J_tT_7$	$v_6 - A_tT_7 + \frac{1}{2}J_tT_7^2$	$s_a + s_c + s_{d_2} + v_6T_7 - \frac{1}{2}A_tT_7^2 + \frac{1}{6}J_tT_7^3$

As mentioned before in CSFA algorithm, each NURBS block is categorized in three sections namely, short, medium, and long blocks. All NURBS blocks could also be categorized into 17 subsections depending on their length, initial, and final velocity. Fig. 2 shows the 17 general cases that are applied to each NURBS block. To schedule the S-shaped velocity profile, it is necessary to initially determine the category of each NURBS block based on the aforementioned 17 scenarios. For this purpose, the characteristic length of the curve, s_{cri}^s, for the NURBS blocks is defined as follows [7]:

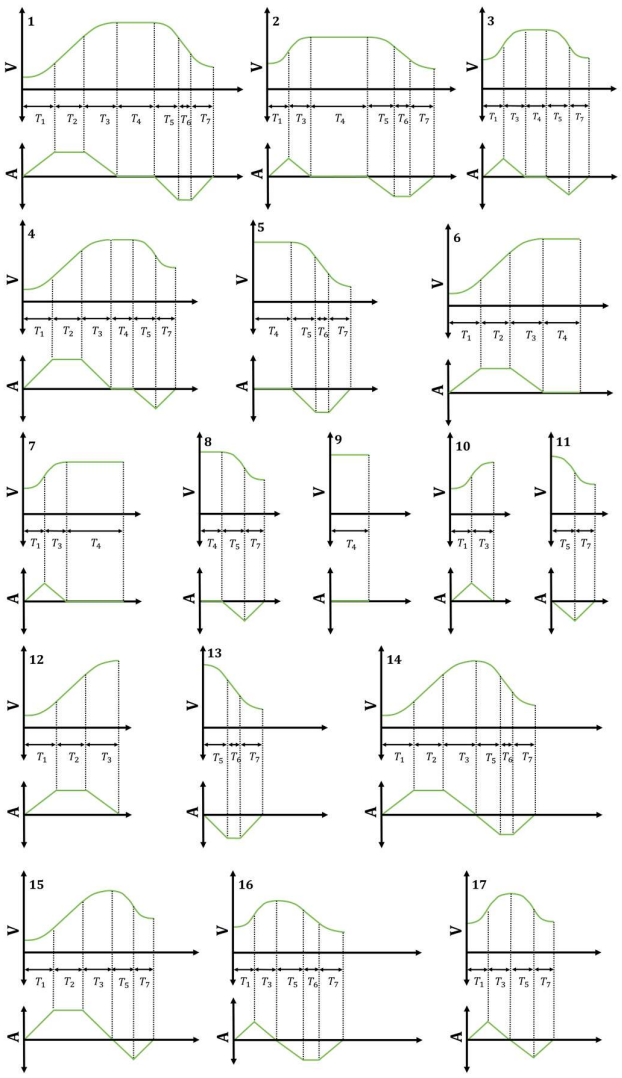

Fig. 2 General NURBS block cases [7]

If $v_s < v_e$, the value of s_{cri}^s is equal to the length of the acceleration section of the curve, $s_a(v_s, v_e)$, and can be calculated as follows:

$$s_a(v_s, v_e) = \begin{cases} (v_s + v_e)\sqrt{\dfrac{v_e - v_s}{J_t}} & v_e - v_s \le \dfrac{A_t^2}{J_t} \\ \dfrac{1}{2(v_s + v_e)\left[\dfrac{A_t}{J_t} + \dfrac{(v_e - v_s)}{A_t}\right]} v_e - v_s > \dfrac{A_t^2}{J_t} \end{cases} \quad (24)$$

If $v_s > v_e$, the value of s_{cri}^s becomes equal to the length of the deceleration section of the curve, $s_d(v_s, v_e)$, see Eq. (25).

$$s_d(v_s, v_e) = \begin{cases} (v_s + v_e)\sqrt{\dfrac{v_s - v_e}{J_t}} & v_s - v_e \le \dfrac{A_t^2}{J_t} \\ \dfrac{1}{2}(v_s + v_e)\left[\dfrac{A_t}{J_t} + \dfrac{(v_s - v_e)}{A_t}\right] v_s - v_e > \dfrac{A_t^2}{J_t} \end{cases} \quad (25)$$

If the desirable curve length (s_i) matches the characteristic length of the curve, the kinematic profiles fall into one of the scenarios labeled between 10 to 13 in Fig. 2, known as short NURBS blocks. Otherwise, the NURBS block falls into either the medium or long block scenarios. To precisely investigate the exact type of the block, another characteristic curve length, s_{cri}^l, is defined, see Eq. (26) [7]:

$$s_{cri}^l = s_a(v_s, v_{max}) + s_d(v_{max}, v_e) \quad (26)$$

Here, the lengths of the acceleration and deceleration phase $s_a(v_s, v_e)$ and $s_d(v_s, v_e)$ are calculated using Eq. (24) and Eq. (25), respectively.

If the desirable length of the curve exceeds the characteristic length, s_{cri}^l, the block is categorized as a long one, falling into one of the scenarios 1 to 9 in Fig. 2. In cases where the block does not belong to any of the short or long scenarios, it is categorized as a medium block, corresponding to one of the scenarios 14 to 17 in Fig. 2.

To calculate the T_k $(k = 1\ to\ 7)$ values based on the type of the block, one of the algorithms for short, long, and medium NURBS blocks presented in Fig. (3) to (5) must be implemented [7].

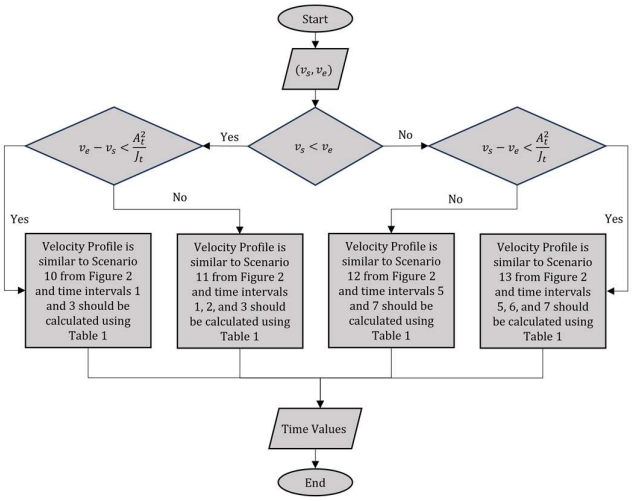

Fig 3 Time interval calculation flowchart for short blocks

2.4. Variable jerk compensation

After calculating the time intervals of the kinematic profiles, each interval must be defined as an integer multiple of the sampling time T_s. Since the magnitude $|J_t|$ in section one is set equal to the magnitude $|-J_t|$ in section three, time interval T_1 and T_3 are assumed to be equal. The number of intermediate time steps for sections one and three is calculated by Eq. (27):

$$\begin{cases} N_{a1} = round\left(\dfrac{T_1}{T_s}\right) \\ N_{a3} = round\left(\dfrac{T_3}{T_s}\right) \end{cases} \quad (27)$$

If the values of N_{a1} and N_{a3} are zero for non-zero values of T_1 and T_3, the timing of the kinematic profile for the block of interest is not accurate. Consequently, a variable jerk

compensation algorithm, depicted in Fig. 6 and defined in Eq. (28), has been introduced, in which the values of T_1 and T_3 are set to T_s. The rational is to ensure the jerk continuity at connection points. It should be noted that the modified jerk, J_t^v, is only used as the jerk within that block.[7].

$$J_t^v = \frac{v_e - v_s}{T_s^2} \tag{28}$$

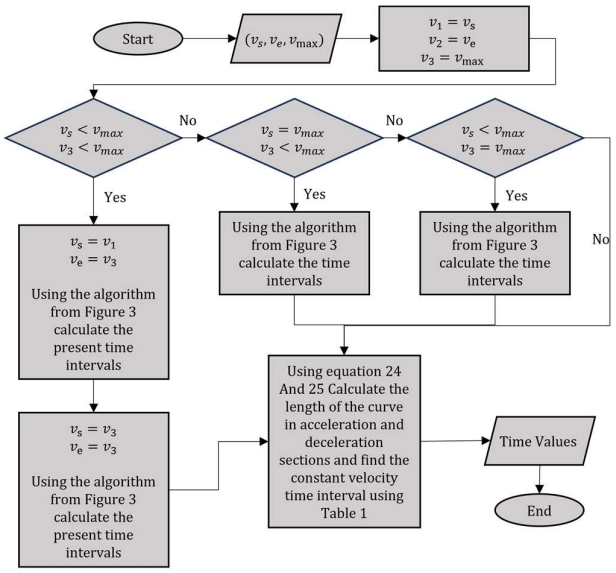

Fig. 4 Time interval calculation flowchart for long blocks

2.5. Arc length compensation

After variable jerk compensation, a rounding error e_i arises, represented as the difference between the desirable path length and the calculated path lengths, see Eq. (29).

$$e_i = s_i - (s_a + s_c + s_d) \tag{29}$$

The values of s_a, s_d, and s_c respectively represent the length in the acceleration, deceleration, and constant speed phases. The arc length during the acceleration phase is calculated as:

$$\begin{cases} s_{a1} = v_s(N_{a1}T_s) + \frac{1}{6}J_t(N_{a1}T_s)^3 \\ s_{a2} = s_{a1} + v_1(N_{a2}T_s) + \frac{1}{2}A_t(N_{a2}T_s)^2 \\ s_a = s_{a2} + v_2(N_{a3}T_s) + \frac{1}{2}A_t(N_{a3}T_s)^2 - \frac{1}{6}J_t(N_{a3}T_s)^3 \end{cases} \tag{30}$$

In this context, N_{a1}, N_{a2}, and N_{a3} represent the count of intermediate time steps [9]. Arc length in the deceleration phase could be found by substituting v_s, N_{a1}, N_{a2}, and $N_{a3}$with v_e, N_{d3}, N_{d2}, and N_{d1} in Eq. (30). The arc length of the constant-speed motion section is calculated as follows:

$$s_c = N_c v_{max} \tag{31}$$

Here, N_c represents the number of time steps in constant-speed motion. To enhance the precision of interpolation, an algorithm for error compensation is introduced. If there is no section with constant acceleration, the arc length error is distributed in the first and third segments as follows:

$$\underbrace{0\ \Delta e_i\ 2\Delta e_i \dots\ (N_{a1}-1)\Delta e_i}_{N_{a1}}\ \underbrace{(N_{a3}-1)\Delta e_i \dots\ 2\Delta e_i\ 0}_{N_{a3}} \tag{32}$$

Since $T_1 = T_3$, then $N_{a1} = N_{a3}$, and the value of Δe_i is calculated as:

$$\Delta e_i = \frac{e_i}{N_{a1}(N_{a1}-1)} \tag{33}$$

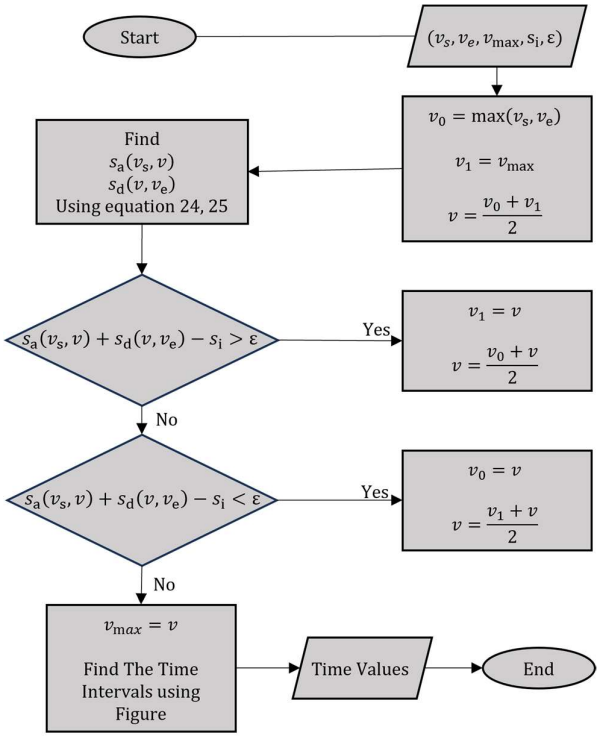

Fig. 5 Time interval calculation flowchart for medium blocks

The increment in arc length at each interpolation step (Δs_j) is modified as:

$$\Delta s_j^{'} = \begin{cases} \Delta s_j + (j-1)\Delta e_i if\ \mathrm{j} \le N_{a1} \\ \Delta s_j + (2N_{a1} - j)\Delta e_i if\ \mathrm{N}_{a1} + 1 \le j < 2N_{a1} \end{cases} \tag{34}$$

In case of having the constant acceleration motion within the motion profile, the compensation strategy is expanded, distributing the length error among N_{a1}, N_{a2}, and N_{a3}:

$$\begin{matrix} \underbrace{0\ \Delta e_i\ 2\Delta e_i \dots\ (N_{a1}-1)\Delta e_i}_{N_{a1}} \\ \underbrace{(N_{a1})\Delta e_i\ (N_{a1})\Delta e_i \dots (N_{a1})\Delta e_i\ (N_{a1})\Delta e_i}_{N_{a2}} \\ \underbrace{(N_{a3}-1)\Delta e_i \dots\ 2\Delta e_i\ 0}_{N_{a3}} \end{matrix} \tag{35}$$

Therefore, the value of Δe_i is calculated as:

$$\Delta e_i = \frac{e_i}{N_{a1}(N_{a1}-1) + N_{a1}N_{a2}} \tag{36}$$

The increment in arc length at each interpolation step (Δs_j)in this case is modified as:

$$\Delta s_j' = \begin{cases} \Delta s_j + (j-1)\Delta e_i \ if \ \ j \le N_{a1} \\ \Delta s_j + N_{a1}\Delta e_i \ if \ \ N_{a1}+1 \le j < N_{a1}+N_{a2} \\ \Delta s_j + (2N_{a1}+N_{a2}-j)\Delta e_i \ if \ \ N_{a1}+N_{a2}+1 \le j < 2N_{a1}+N_{a2} \end{cases} \quad (37)$$

Since in the section with constant-speed motion, the desired velocity matches the desired feed rate, error compensation for arc length is only applied to the acceleration and deceleration sections to prevent speed jumps at high-speed [7].

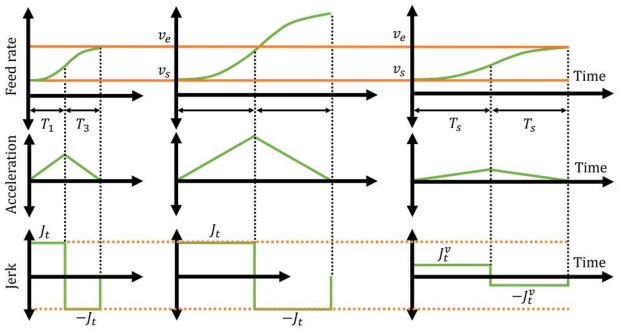

Fig. 6 Variable jerk compensation [7]

2.6. Second order interpolation

In this step, the NURBS curve is interpolated to find the accurate location after each time step. To achieve this, a second order Taylor expansion with respect to the curve length is utilized:

$$u(s_j+1) = u(s_j) + \frac{du}{ds}|_{s=s_j}\Delta s_j + \frac{1}{2}\frac{d^2u}{ds^2}|_{s=s_j}\Delta s_j^2 \quad (38)$$

In order to generate accurate values, $\Delta s_j'$ must be used in Eq. (39) instead of Δs_j [7]. In order to increase the interpolation accuracy, higher order Taylor expansions could be used. Moreover, the interpolation accuracy could be further increased by decreasing the sampling time and chord error.

3. Experimental setup

Experimental setup, shown in Fig. 7, was specifically designed, and built for this research. Its design provides extensive freedom of motion in applications which require a large workspace while dimensional accuracy is not the primary concern. This device could be utilized in mural designs where a large surface of an arbitrary build material is used as the workspace. The experimental setup was comprised of two flexible motion axes equipped with a sprocket and a chain attached to an assembly carrying a marker whose motion visualized the trajectory. These two axes were installed on a vertical bed and driven by stepper motors. Control commands were transmitted to the motors through STM32F746ZGT6 card and drivers. Additionally, two rotary encoders were installed at the end of the motor shafts to acquire precise rotation angle to calculate movement error.

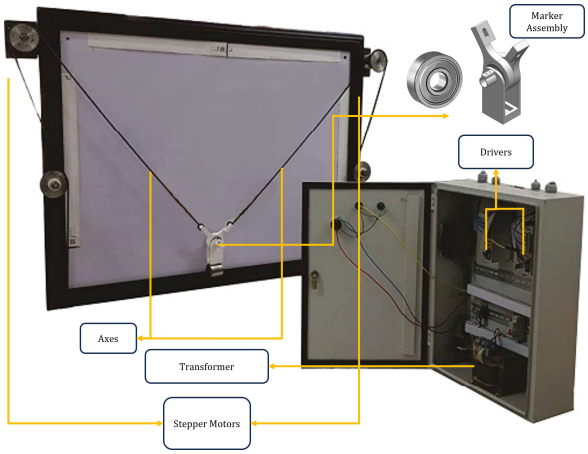

Fig. 7 Experimental setup

3.1. Inverse kinematics

Interpolated points which were derived from second order Taylor expansion are in cartesian coordinate system. To generate the desired trajectory, it is necessary to transfer the respective points from the Cartesian coordinate system to the non-Cartesian (experimental setup) coordinate system L_1 and L_2, see Fig. 8.

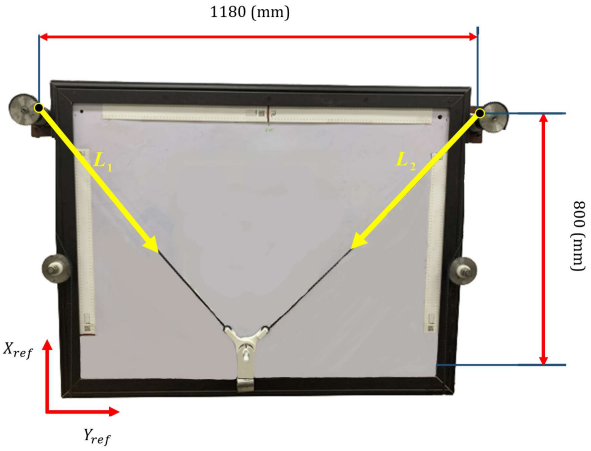

Fig. 8 Experimental setup's coordinate system

To transform the calculated points from Cartesian coordinates into the L_1 and L_2, the inverse kinematics equations were employed and the values of L_1 and L_2 at each point were calculated based on their corresponding (x_{ref}, y_{ref}). In Eq. (39), width and length of the platform are 800 mm and 1180 mm respectively.

$$L_1 = \sqrt{x_{ref}^2 + (width - y_{ref})^2}$$
$$L_2 = \sqrt{(length - x_{ref})^2 + (width - y_{ref})^2} \quad (39)$$

3.2. Error calculation

In order to calculate the kinematic profiles error, two rotary encoders connected to the end of the motors were used. Given that the encoder outputs represent the motor rotation angle, it was necessary to first convert the encoder output values to $(L_{1_{actual}}, L_{2_{actual}})$ coordinates using Eq. 40:

$$L_{1_{actual}} = \frac{encoder_1\ Pulse \times BLU}{motor\ encoder\ ratio}$$
$$L_{2actual} = \frac{encoder_2\ Pulse * BLU}{motor\ encoder\ ratio} \quad (40)$$

In Eq. (40), BLU is basic length unit which in this case represents $L_{1_{actual}}$ and $L_{2_{actual}}$ increments for one stepper motor pulse. Motor encoder ratio indicates the number of encoder pulses for one motor step. Eq. (41) outlines how BLU and motor encoder ratio are calculated:

$$BLU = \frac{D \times \pi}{steps\ per\ revoloution * gearbox\ ratio}$$
$$motor\ encoder\ ratio = \left(\frac{encoder\ pulses\ per\ revoloution}{steps\ per\ revoloution}\right) \quad (41)$$

In Eq. (41), D is the sprocket diameter (100 millimeters) that was connected to the motor, motor steps per revolution is the number of motor pulses in one revolution (400 pulses), gearbox ratio is 5, and encoder pulses per revolution is the number of encoder output pulses for one revolution (7200 pulses). The final values of BLU and motor encoder ratio were calculated as 157 mm and 18 mm, respectively. In the next step (x_{actual}, y_{actual}) values were computed using the following equation:

$$x_{actual} = \frac{length^2 + L_{1actual}^2 - L_{2actual}^2}{2 \times length}$$
$$y_{actual} = width - \sqrt{L_{1acual}^2 - x_{actual}} \quad (42)$$

Substituting the values obtained from the encoder into Eq. (42), the actual coordinates of the travelled trajectory could be obtained. Five-point numerical differentiation was used to calculate the actual kinematic profiles. Finally, practical kinematic profiles were compared with the theoretical kinematic profiles obtained from the interpolation algorithm CSFA. Ultimately, by comparing the theoretical and practical kinematic profiles, error profiles could be calculated.

3.3. Motion generation

In order to generate the desired motion on the experimental setup, the trajectory was designed in a CAD software and NURBS curve's properties (control points, knot vector, degree of the curve, and weight of the control points) were extracted. In the next step, CSFA algorithm was utilized to interpolate the points in cartesian coordinate system followed by implementation of the inverse kinematics. Outputs (Δs commands) were uploaded on an SD card and imported to the STM32F746ZG microcontroller. The microcontroller sent the motion commands to the drivers to drive the stepper motors. Encoder's measurements were fed back to the microcontroller and transmitted to the PC for error profile calculations. Fig. 9 provides a flowchart of system configuration.

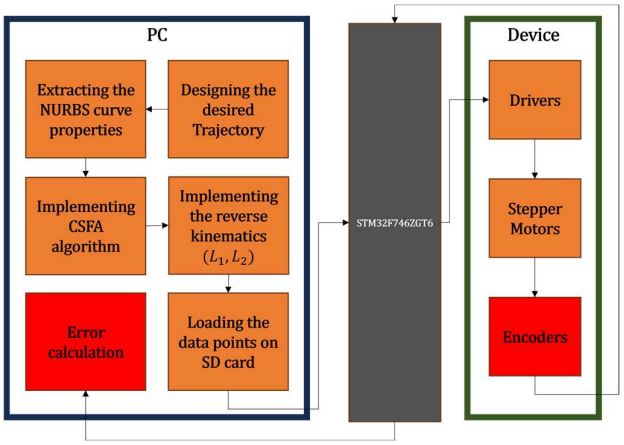

Fig. 9 Schematic flow chart of the experimental setup configuration

3.4. Verifying the accuracy of data measured by the encoder

In order to verify the accuracy of the acquired data from the encoders, a 4-axis CNC mill was employed. The rotary encoder was installed on the 4th axis (rotary axis) of the CNC mill. This axis was then commanded to rotate a specific degree and an encoder was tasked to measure the rotation. The commanded and the measured rotation were then compared to verify the accuracy of data measured by the encoder. In this experiment, the rotary axis of the CNC mill rotated 360 in 18 steps of 20 degrees. Encoder measured values are presented in Fig. 10.

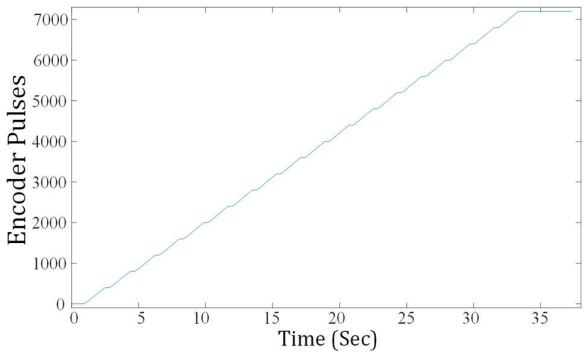

Fig. 10 Encoder measurements for one revolution of CNC's rotary axis

4. Results and Discussions

To assess the performance of the proposed interpolation algorithm, two freeform CAD models consisting of lines and curves with varying curvatures were selected and the required trajectory to plot them were generated, see Fig. 11 and Fig. 12.

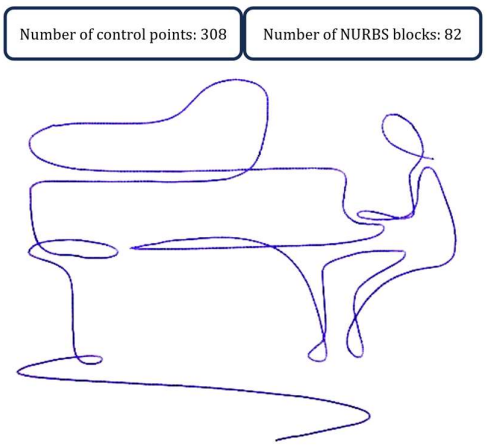

Fig. 11 "The pianist" created using the experimental setup.

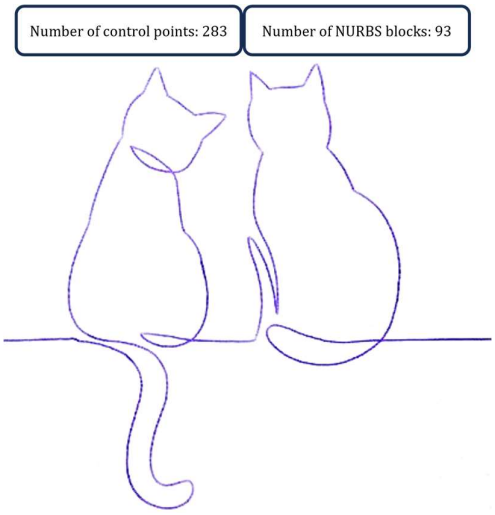

Fig. 12 "The cats" created using the experimental setup.

Parameters in Table 2 were kept consistent among the experiments. It should be mentioned that the values of the interpolation sampling time and data acquisition from rotary encoders has been limited by STM32F746ZGT6 card which has been used as the processing unit in the current study.

Table 2Constant values through the experiments

Property	Value
Feed rate	20 mm/sec
Normal and tangential acceleration	10 mm/sec^2
Normal and tangential jerk	30 mm/sec^3
Initial velocity	0 mm/sec
Final velocity	0 mm/sec
Sampling time	0.001 $Second$
Chord error	5 e -6 mm

It is worth noting that constant values in Table 2 for maximum acceleration, maximum deceleration, maximum jerk, and feed rate has been chosen based on the limitations of stepper motors and flexible axes.

4.1. Experimental tests

Fig. 13 compares the theoretical coordinates from original designed trajectory with actual values calculated using encoders.

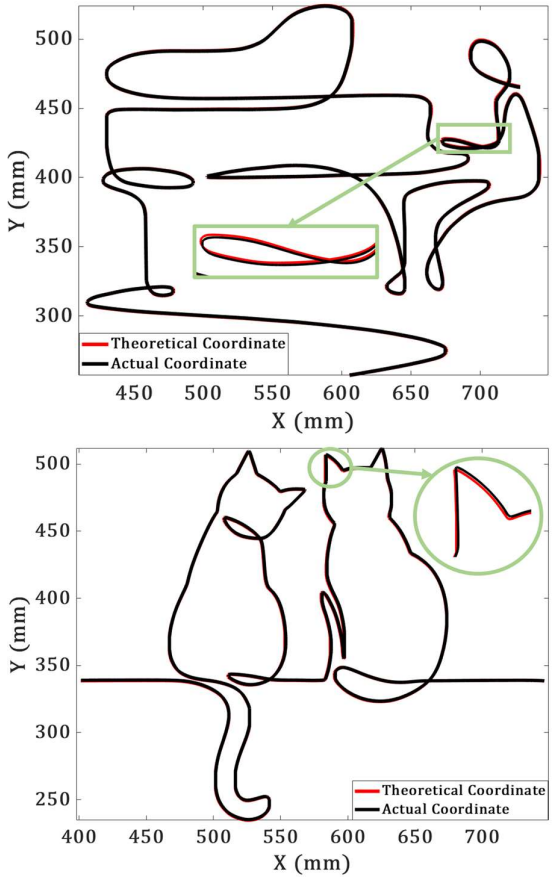

Fig. 13 Comparison of theoretical coordinate values with actual interpolated values for a) “The pianist” b) ”The cats”

A comparison between theoretical and actual velocity is presented in Fig. 14. In this figure multiple acceleration and deceleration phases could be seen which takes place in the transition from one NURBS block to subsequent one. Presented velocity profile ensures smooth transition between blocks. Fig. 15 and Fig. 16 presents the relative error profiles for length and velocity for “the pianist” and “the cats” respectively. As observed, the calculated errors in the velocity profiles are greater than the errors in the motor motion and length profiles. This phenomenon can be attributed to the nature of stepper motors, in which motion is not continuous.

In each step of motion, motor voltage varies logically between zero and one (in this experiment, the logical values correspond to zero volts and 55 volts) which leads to oscillations in each step of motion. Practical velocity profiles are calculated using numerical differentiation of length profiles which makes mentioned oscillations more prominent. The discrete behaviour of encoder measurements (digital pulses) is another reason for greater velocity errors. This discontinuity in pulses, similar to the impact of the stepper motor nature, have a more pronounced effect on the velocity profile.

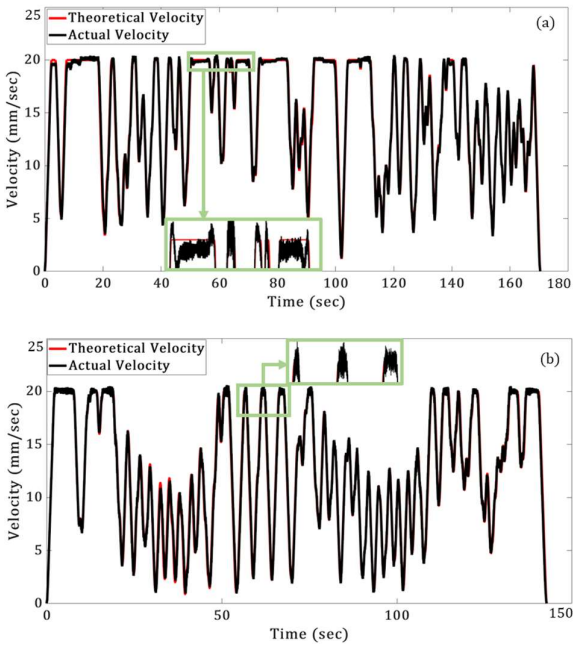

Fig. 14 Comparison of theoretical and actual velocity of a) "The pianist" b) "The cats"

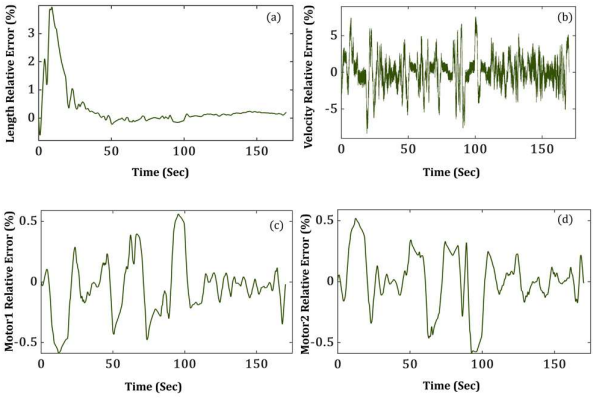

Fig. 15 Relative error profile for "The Pianist" (a) length (b) velocity (c) first motor rotation (d) second motor rotation

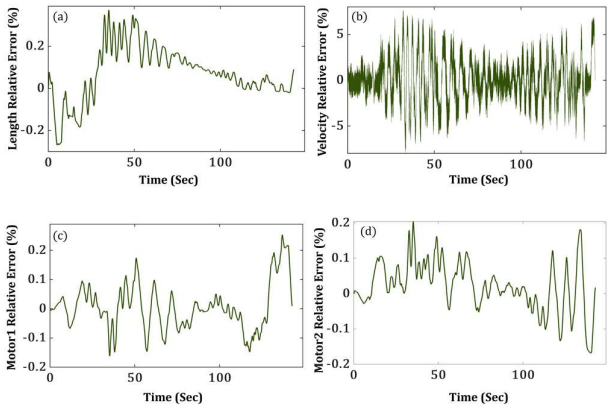

Fig. 16 Relative error profile for "The Cats" (a) length (b) velocity (c) first motor rotation (d) second motor rotation.

5. Conclusion

In this paper, a new algorithm for generating a jerk limited continuous tool path for non-cartesian coordinate systems was introduced. Considering the natural ability of NURBS curves in creating free form shapes, these curves were used to provide flexibility in the design process A complete S-shape feed-rate scheduling algorithm was used to guarantee the continuity of the kinematic profiles and avoid undesirable fluctuations during the movements. This method provided a continuous smooth movement along the desired path. In order to validate the proposed algorithm an experimental setup was designed and built. Unlike conventional manufacturing systems in which the work piece dimensions are limited by the worktable, designed experimental setup used flexible axes which could be installed on variable surfaces and would not limit the workspace. Stepper motor motion commands were transferred from PC using a STM32746ZG card. In order to verify the accuracy of the motion, two rotary encoders were installed at the end of motor shafts which acquired rotation values of the stepper motors.

References

[1] Tresset, P. and F.F. Leymarie, Portrait drawing by Paul the robot. Computers & Graphics, 2013. 37(5): p. 348-363.
[2] Bazaz, S.A. and B. Tondu. Online computing of a robotic manipulator joint trajectory with velocity and acceleration constraints. in Proceedings of the 1997 IEEE International Symposium on Assembly and Task Planning (ISATP'97)-Towards Flexible and Agile Assembly and Manufacturing-. 1997. IEEE.
[3] Erkorkmaz, K. and Y. Altintas, High speed CNC system design. Part I: jerk limited trajectory generation and quintic spline interpolation. International Journal of machine tools and manufacture, 2001. 41(9): p. 1323-1345.
[4] Sekar, M., V. Narayanan, and S.-H. Yang, Design of jerk bounded feedrate with ripple effect for adaptive nurbs interpolator. The International Journal of Advanced Manufacturing Technology, 2008. 37: p. 545-552.
[5] Zhao, H., L. Zhu, and H. Ding, A real-time look-ahead interpolation methodology with curvature-continuous B-spline transition scheme for CNC machining of short line segments. International Journal of Machine Tools and Manufacture, 2013. 65: p. 88-98.
[6] Lei, W. T., Sung, M. P., Lin, L. Y., & Huang, J. J. (2007). Fast real-time NURBS path interpolation for CNC machine tools. International Journal of Machine Tools and Manufacture, 47(10), 1530-1541.
[7] Du, X., J. Huang, and L.-M. Zhu, A complete S-shape feed rate scheduling approach for NURBS interpolator. Journal of Computational Design and Engineering, 2015. 2(4): p. 206-217.
[8] Heng, M. and K. Erkorkmaz, Design of a NURBS interpolator with minimal feed fluctuation and continuous feed modulation capability. International Journal of Machine Tools and Manufacture, 2010. 50(3): p. 281-293.
[9] Wang, T. Y., Zhang, Y. B., Dong, J. C., Ke, R. J., & Ding, Y. Y. (2020). NURBS interpolator with adaptive smooth feedrate scheduling and minimal feedrate fluctuation. International Journal of Precision Engineering and Manufacturing, 21, 273-290.
[10] Ji, S., Hu, T., Huang, Z., & Zhang, C. (2020). A NURBS curve interpolator with small feedrate fluctuation based on arc length prediction and correction. The International Journal of Advanced Manufacturing Technology, 111, 2095-2104.
[11] Nie, M., L. Zou, and T. Zhu, Jerk-Continuous Feedrate Optimization Method for NURBS Interpolation. IEEE Access, 2023. 11: p. 25664-25681.
[12] Piegl, L. and W. Tiller, The NURBS book.[Sl]: Springer Science & Business Media, 2012. Citado, 1995. 6: p. 34-39.