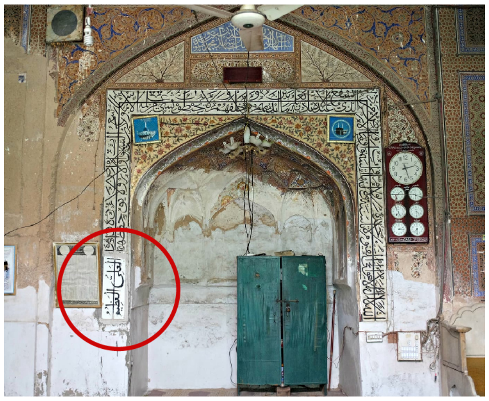

Fig. 1. A photo of a wall inside Mariyam Zamani Mosque (also known as Begum Shahi Masjid) Lahore. Some of the calligraphy has been reconstructed (highlighted in red) using conventional techniques and can easily be recognised because it looks different than the rest.

Robotic Reconstruction of Islamic Calligraphy Using Twisting Bezier Splines

Muhammad Umar Hassan, Muhammad Sabieh Anwar and Ali Raza

Abstract

Automated reconstruction of Islamic architectural calligraphy can save thousands of nobel and historic scripts all over the world from going extinct. One of the fundamental problems in such reconstruction tasks is the absence of a tool that can bridge the gap between artists and the machines (such as a robotic arms) by taking input as conventional calligraphy format and transform into a robot-program. Not just the reconstruction, even in this digital age, no such method exists that lets the artists generate new broad-edge scripts that can be reproduced mechanically. In this context, the research presented herein introduces a novel innovation in the conventional Bezier spline curves to use them to effectively answer the artistic requirements of copying or creating broad-edge calligraphy scripts and can directly produce data needed by the industrial robots. To demonstrate the effectiveness of our approach, two famous scripts are reproduced using the new splines and a comparison is made with the originals. Additionally, to establish how these splines can be used with machines, a robotic simulator is also discussed and the output analyzed both qualitatively and quantitatively. In addition to mechanized reconstruction Islamic calligraphy, the presented approach can also be used for generating calligraphy for virtual 3D models, e.g. in Metaverse, 3D documentation of non-reachable and ruined buildings and even OCR of islamic calligraphy scripts.

Index Terms

Art and Entertainment Robotics, Motion Control, Human-Centered Automation, Software Tools for Robot Programming

I. Introduction

THE art of Islamic calligraphy has a history that dates back to the seventh century [1], [2]. It has witnessed many evolutionary stages [2], [3] and has been used by artists speaking several different languages [4] and sharing uncommon biographies [5], [6], [7], [8]. Unfortunately though, the industrial age and the advent of technology has not spared this beautiful art from its subsuming tide, with traditional calligraphy struggling to keep up with digitization and dwindling community support. The very existence of Islamic calligraphy now faces a serious threat. Public buildings and infrastructure that once used to be a showcase for the most laudable artists of the time have turned into museums; awaiting to be wiped away slowly with each round of the monsoon and every splash of the ocean's waves. One such site is shown in Figure 1.

Potentially, we can use robotic dexterity to help us in this domain. Industrial robots have already been used in the field to perform unorthodox tasks such as [9], [10], [11], [12] and they can surely uplift this art as well. At the very least, they can be employed in restoration and replication of existing calligraphic work [14]. In other words, they can be used as printers, or rather one may say, "painters" that give an extra hand to the calligraphers to open up new dimensions of art work that can not only revamp existing calligraphy sites but also create new ones.

Mechanized robotic drawing of the Islamic calligraphy scripts requires not just the ink-mark information but also the information about the tool movement [3]. Compared to industrial color printers that mostly print on flat surfaces, or rolled surfaces at best, using a flexible broad edge brush to draw on possibly uneven curved surfaces situated in narrow spaces, makes the job relatively more complex. This complexity arises because in this realm, in addition to normal tool movement information, a robot needs to take special care about the orientation of a flexible tool and its surface facing force as well.

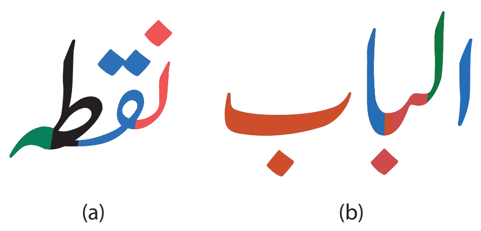

Fig. 2. Words written in Nastaleeq script. (a) Individual letters in a word are colored differently to show that a single stroke can contain multiple letters. (b) The parts painted in the same color look different but actually represent the same letters.

Since an industrial robotic arm can accurately maneuver paths in three dimensional space with controlled speed and downward force, the residual problem can be mainly divided in two parts. First, discover a method that can not only digitize existing Islamic calligraphy specimens but can also allow modifying and creating new ones. Second, discovering a method to extract machine data from graphical data.

This research proposes one solution to answer both of these questions: unifying graphical data with machine data. Conventionally, the kind of data used by the computer to render and print graphics on the screen or on paper fundamentally differs from the data needed by a machine [cite] that uses a mechanical end-effector to produce output. We propose a rather simple innovation in the conventional bezier splines which enables them to mimic the ink-mark of broad-edge tools and can directly generate machine movement data. We name them “Rotating or Twisting Bezier Spline Curves”, or simply, “Twisting Bezier Splines”.

There are still some challenges which need consideration even with a twisting bezier spline. The inkmark in most islamic calligraphy scripts is usually a result of multiple closely located tool strokes [cite] that form up words using overlapping letters. Furthermore, as a result of spin of a broad-edge tool, the width of the strokes continuously varies. No computer algorithm currently exists that can accurately determine the pitch lines of these islamic calligraphy strokes because of the aforementioned complexities. Twisting bezier splines can intrinsically extract this pitch line during the tracing process.

The article is divided in 5 sections. After the introduction, we briefly highlight the related complexities involved in writing islamic scripts in Chapter II. Chapter III then presents the working principle and mathematical model of the twisting bezier splines. In section IV we discuss some performance metric and discuss the tests performed to gauge the performance of the twisting bezier splines. Section V discusses the simulation of machine data produced using twisting bezier splines of a robotic manipulator in order to validate the results. We finally conclude in Section VI. The software tools used in this research, the software user manuals, video tutorials and the source codes can be viewed on the project GitHub repo[13].

II. Challenges Involved in Digitizing Islamic Calligraphy

Islamic calligraphy is mostly performed in scripts that are used to write Arabic, Urdu, Persian and similar languages. These scripts share some common traits that make their digitization a challenge.

- Most of the times, the individual letters have to overlap with each other to form a complete word. See Figure 2(a). This makes it difficult to break down words into individual letters.
- Most of the letters can appear in multiple forms depending upon its position in the word as well as the artist’s discretion. Figure 2(b) shows a word that uses letters in multiple shapes.
- Most of the times, a single stroke of pen is used to write major portions of words that include multiple overlapping letters.
- Many a times, parts of words and letters that seem to be drawn using a single stroke are actually drawn using multiple strokes and are only made to look like a continuous stroke. See Figure 4(c) and 5(c) which shows overlapping regions of multiple strokes that overall have the look of a single stroke.

III. Rotating/Twisting Bezier Splines

A. The need of twisting bezier splines

Conventional Bezier curves are commonly used to define outline fonts [cite] and digital calligraphy [cite] due to their ability to accurately trace [cite] the outline of scripts written with and without a broad edge tool [cite]. These curves can be easily manipulated [cite] and rendered [cite] on a computer screen as well as they can be printed on paper. However, in light of the underlaying problem, and in order to preserve the essence of artistic norms [cite], it is desired that a script is written using a conventional broad-edge tool by a robotic manipulator [cite] in a similar fashion to a human artist. Separate techniques [cite] are needed to convert output of the outline or pitch curve functions to a format that a robot can directly use for the desired tool movement. Even though many of these techniques [cite] can promise accurate tracing, none of them can promise that the robot will use the exact tool movement of a human agent.

In most literal terms, beizer spline curves are sets of decimal values that define the graphical shape of certain mathematical functions [cite]. The input of these functions contains two dimensional points located in the frame of reference of the screen on which they are created and work as handles to control the shape of the curve using a screen pointer. A user can physically relocate these curvature handles on the computer screen and the shape of the resulting spline will follow. The output of these mathematical functions is absolutely repeatable and can be linearly scaled to any units. Usually, a curve is made of up a lot of such lines each with its own set of input coefficients. Although these individual functions are continuous, there output can easily be discretized as closed paths made up of closely located two dimensional points with controllable resolution [cite]. This is exactly the kind of information required by most of computer graphics and printing drivers to render an output.

Now, as effective as the bezier curves are for screen and paper printing, they still cannot tell how a physical tool should move on a piece of paper to create the desired output. The splines are just organic shaped paths with no thickness. One of the ways to convert them into ink-mark, is to interpret them as the pitch line for a thick round tool tip. This technique is used by plotters [cite] and some hand writing replicators [cite] to produce a written script. However, only a few fonts [cite] can be replicated by this technique. The other technique is to fill the glyph by moving the tool continuously on a path computed by algorithms such as those used by CAD tools [cite] to fill in (or cut) the area contained by the outline of the glyph using a thick round tip tool. The later can produce outputs that looks similar to broad-edge calligraphy but will still not be the same due to the visible tool paths that are not expected when using an actual broad edge tool.

On the other hand, twisting bezier splines, instead of working as outline curves, directly record the pitch line of a twisting broad-edge tool stroke along-with the twist information and recreate the live result to the artist who can than morph the spline in a convenient way as normal splines. Their usage is just as intuitive as the conventional bezier splines and will be demonstrated in the next sections, and the information they contain can not only be used to render the script back on the screen or a photo printer, but also be directly considered as machine data for robotic manipulators.

In contrast to normal bezier splines, twisting bezier splines not only trace the pitch lines of all the strokes but also the twist of the tool independent of the curvature. This is done by introducing another input to the curve function that we call the "rotation" or "twist" handle. Just like the curvature handles represent the function inputs responsible to define the curvature of the pitch line which is in fact a normal spline, the rotation handles represent the inputs that control the twist of the simulated broad-edge tool. Just like conventional bezier splines, functions of the twisting bezier splines can also be discretized and converted into a list of two dimensional list of points; a closed path to emulate the ink-mark of the broad edge tool. As was the case with normal splines, this is exactly the kind of data needed by the computer display drivers.

Interestingly, although the twisting splines originate on a computer screen, they are still nearer to the machine than they are to computer graphics data. To compute the list of the points needed to create a filled path that represents the ink-mark, a broad edge tool is emulated to be moving on the rotating spline with the twist also controlled by the spline. In actuality, the emulated tool is replaced by a real tool that can be mounted on a robotic manipulator. The spline directly controls the position and twist of the tip of a tool which can directly be translated into machine movement codes. The rest of the process for painting involves inverse kinematics [cite] and is already handled by the robot controller.

B. Mathematical Model of a Rotating Bezier Spline

1) The Conventional Bezier Spline: We first describe conventional bezier splines. Figure 3 shows an illustration of a spline path composed of several sub curves. Each curve section is only partly independent of the other. Figure 3 (a) shows the final shape of the curve without any construction elements. In Figure 3 (b), we explode different sections of the curve into smaller elements and show how they fit together to form the complete spline. There are five sections in this curve labeled 1 through 5. Figure 3 (c) shows an assembled form of these five sections. It also shows, what are called, anchors and construction handles. The anchor is the point that sits at the terminals of two adjacent curve sections. For instance, anchor point A is connecting the sections 1 and 2. Like all the other anchors, this anchor also has two handles, H_1 and H_2, connected through a straight line passing through the anchor. The length of each handle, $\overline{AH_1}$ and $\overline{AH_2}$ on both sides of the anchor define the shape of the curve section on their respective side where as the orientation of line connecting both handles contributes to the shape on both the curve sections. This is how both sections become partly independent. For instance, handle $\overline{AH_1}$ contributes to the shape of curve segment 1 and $\overline{AH_1}$ to section 2.

Now, it may look like the shapes defined in this way are pretty organic but in fact, the whole shape is defined by simple mathematical equations. Figure 3 (d) focuses on section 4 and 5 of the curve and also shows a polygon defined by the points P_1, P_2, P_3 and P_4. It must be noted that the points P_1 and P_4 of this polygon are also the anchor points between sections 3, 4 and 5. Take section 4 for example. The polygon mathematically defines the complete shape of this curved part. If P_{spline} is a point on section 4, with coordinates x and y in a Cartesian plane referenced to some origin, it is defined as

$$P_{spline} = P_b f + P_a(1 - f). \tag{1}$$

where,

$$P_a = P_{23} f + P_{12}(1 - f) \tag{2}$$

and

$$P_b = P_{34} f + P_{23}(1 - f) \tag{3}$$

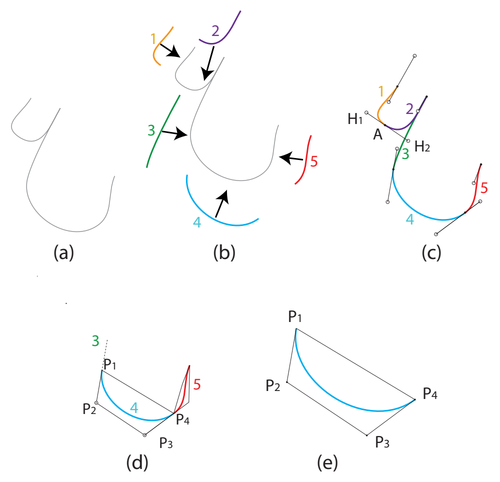

Fig. 3. An illustration showing the construction of Bezier Spline Curve. (a) A sample of a Bezier spline path (b) an exploded view of inner curves of the Bezier spline path (c) Handles that control the shape of the two adjacent sub curves (d) and (e) Construction polygon of the sub curve.

where,

$$P_{12} = P_2 f + P_1(1-f), \tag{4}$$
$$P_{23} = P_3 f + P_2(1-f) \tag{5}$$

and

$$P_{34} = P_4 f + P_3(1-f) \tag{6}$$

and, $f \in [0,1]$.

It can also be proved that the side segments of the polygon $\overline{P_1P_2}$ and $\overline{P_3P_4}$ are tangent to the curve at the point they meet it at P_1 and P_4 respectively.

2) Twist Handle: On top of the conventional bezier splines that work around anchor points possessing curvature handles, we add a "rotation/twist" handle to each anchor and one scalar thickness parameter to the whole curve. The thickness parameter defines the length of a flat line segment centered on P_{spline}, oriented arbitrarily at a constant angle with reference to the curve origin and, sweeping on the curve to form a two dimensional region representing the inkmark. We then define the orientation of this line segment as it sweeps along as a linear function of three variables. First, is the fractional position of the center of the segment which is the same as f. The second and third variables are the two angles that are subtended by the rotation handles about their respective anchors with respect to the horizontal axis. This means that the orientation of the sweeping line when located exactly on a particular anchor is the same as the angle between the twist handle and the center of the respective anchor. Figure **??** (a) shows rotation handles added in the example under discussion. It must be noted that the curvature of the spline remains the same after adding twist handles that are lying horizontally yet. The swept inkmark region is shown in Figure **??** (b). Figure **??** (c) shows the final form of the twisting bezier spline after the twist handles have been iteratively moved to positions that impart the spline with the desired look.

In simpler words, it is similar to sweeping a broad-edge pen centered on the actual spline while twisting it uniformly and continuously about its own axis at an angle

$$\theta_{twist} = \theta_A(1-f) + \theta_B \tag{7}$$

with respect to the horizontal axis. Here f is the same factor that was used to define P_{spline} and θ_A and θ_B are the absolute angles subtended by the first and the second anchor and their respective rotation handles measured from the horizontal axis. It may be noted that since each anchor is connecting two adjacent sub curves, the ending angle of the sweeping line at the end of the first curve is always the same at the beginning of the latter. This obscures the visual transition of the twisting curve from one sub curve to the other.

It must also be noted that the angle of rotation handle cannot be constrained in a 2π domain. Instead, it is completely unbounded, and the sweeping pen may actually take multiple turns both clockwise and anticlockwise while moving on a single curve section as well as the whole curve.

C. Digitization of Calligraphy Artwork

Using rotating bezier splines we can include the artists in the process of digitization of existing calligraphy work. An open source tool called "Gregor" [link to source] is available that can create, modify, save, and reload rotating Bezier splines. With this tool, one can also manually trace existing calligraphy work available in the form of computer images. One can also extend

TABLE I
BENCHMARK OF THE MATHEMATICAL ACCURACY OF THE TWISTING BEZIER SPLINE CURVES

Metrics	**Results**
Area over flow	$< 2\%$
Coverage	$> 94\%$
Lateral deviation from the pitch line	N.A.
Compatibility	All broad edge scripts
Total splines measured	> 100
Total pixels compared	9.9 million
Tested scripts	Nastaleeq, Thuluth

the tool to automate the tracing process by creating a curve fitting technique that tries to fit the output image of the rotating bezier splines on the existing ink-marks in the photos by iterating the coefficients of the spline to minimize the count of pixels in the difference of both images.

D. Machine Data Generation

The rotating spline curves are themselves emulated ink-marks of a broad edge marking tool. This is the reason extracting machine data and even G-codes from them becomes natural. The minimum information required by a robot to draw a broad-edge stroke trickles down to the definition of the path on which the pen must move and the twist of the pen in the world coordinates. Except for the information about a three dimensional reference system, this is exactly the information that a typical rotating bezier spline contains. In other words, to call the rotation bezier splines "machine data", the following assumptions must be made:

- The flat tip of a broad edge tool is always completely touching the drawing area.
- The inclination of the pen with respect to the drawing area or with respect to the direction of the drawing is either normal or always fixed at an angle which is set by the machine.
- For each tool with different tip width, a separate spline will be used.
- The axial pressure a pen inserts on the drawing board while drawing is also fixed and is set by the machine as well.

It is worth mentioning that just like the rotation handle gives the axial rotation information, it is intuitive to add more handles to govern more parameters like pen inclination, tool thickness during a stroke, and normal pressure. The effect of varying thickness will appear right on the screen but to visualize the effect of varying pressure or inclination of the pen will have to be displayed using some 3 D tool visualization, color coding or a similar data presentation technique.

IV. PERFORMANCE AND BECHMARKING

A. Characterization

An important aspect of fabricating a new technique is measuring how well it performs in different usage scenarios. Developing metrics for judging the artistic quality of a calligraphy specimen produced by the Bezier or rotating Bezier splines is altogether a separate discussion and out of scope of this research. However, there are some aspects that we have tried to measure the effectiveness of the rotating splines.

B. Supported Script

All the scripts we analyzed are written with rigid broad-edge tools, such as broad-edge pens and markers, where the tool keeps complete contact with the surface throughout the stoke. However, if the script requires changing the extent of tool contact with the surface, the best alternate to achieve a similar appearance of the script would be to use multiple splines with multiple thicknesses that overlap each other in a gradual manner. The rotating bezier splines cannot control the tool inclination and applied pressure, therefore, they are not suitable for scripts written with flexible tools like broad-edge brushes.

1) Coverage: To benchmark the performance and accuracy of rotating bezier splines, some test results are presented here. These values are produced by comparing high resolution binary images of actual scripts, extracted with Adobe Photoshop and rasterized images produced by the twisting bezier splines originally formed by tracing the processed images. The pixel density is set such that a pixel-pixel comparison can be performed between the original image and the rasterized splines. The accuracy is analysed by computing the overflow and underflow of the ink by matching pixels of both images. Table I shows a summary of these benchmark results.

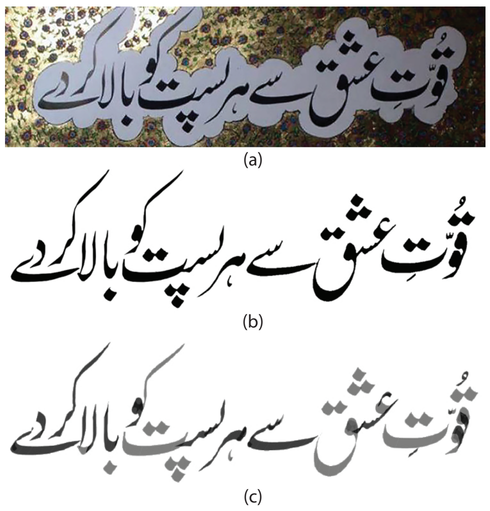

Fig. 4. A specimen produced in "Nastaleeq" script. (a) Original photograph of the specimen (b) Rasterized binary image of the twisting spline curves. (c) Rasteriezd shaded image of the twisting spline curves highlighting individual curve parts.

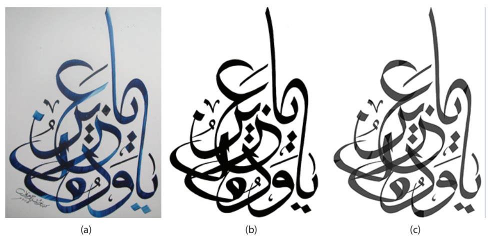

Fig. 5. A specimen produced in "Thuluth" script. (a) Original photograph of the specimen (b) Rasterized binary image of the twisting spline curves. (c) Rasteriezd shaded image of the twisting spline curves highlighting individual curve parts.

2) Sample Results: As both a test and a tribute, two scripts by the famous teacher, artist and author of 18 calligraphy books, late Khursheed Gohar Qalam [23] of the National College of Arts (NCA) were borrowed; one in Nastaleeq style and other in Thuluth. Figure 4 and Figure 5 show the bezier spline sets created by tracing both scripts respectively. Figure 4(a) and Figure 5(a) show the images of both script samples. Figure 4(b) and 5(b) are the rotating splines created by manually tracing the originals. Figure 4(c) and 5(c) highlight individual overlapping spline strokes. [Question: Who traced these scripts?]. Table **??** shows a quantitative measure of accuracy for these script traces.

It must also be noted that since the original script image is not available in a sufficiently high digital quality, the digitization and preprocessing error also contributes to the final error in under and over flow of the rotating bezier splines.

V. Machine Data Generation and Simulation

To check how accurately the data generated by these splines can be traced by an actual robotic manipulator, a computer simulation was used. We developed an open-source simulator and tailored it for the task. The simulator assumes an open loop [type of robot] robot powered by stepper motors with controllable step size. To generate machine data, the simulator first rasterizes the whole spline with a variable resolution such that the distance between two adjacent points is always smaller than and required linear resolution. To fulfil this requirement for a point $N+1$ on the discrete spline, the program iterates the fraction f in the spline model [how do we refer to multiple equations?] to minimize the error between the required resolution and the linear distance between N and $N+1$. For each computed f, the program then calculates the twist component. The set of these the two coordinates of each computed point and the twist component against f defines the position and twist about the normal axis for the tool. The program then transforms this rasterized curve onto an emulated flat writing surface placed in a three dimensional space with some user-set position and orientation. To switch from one spline to the other, the program also adds some additional target points in the array of these points that will lift the tool normally upwards from the writing surface. Finally, the simulator tries to travel on this array of points at a constant speed.

[Description 1] To emulate the writing pen, whenever the simulator detects that the distance of the tip of the tool lies within a configurable thin range, the program records the position and the twist component of the tool on every simulation step. The array of these points is connected using a 3 D polygon that can be visualised on the screen. The 2 D projection of this polygon on to the writing pad is extracted to be processed for the pixel-to-pixel comparison with the original script and the ideal rotating spline. The summary of these tests is shown in Table II.

TABLE II
BENCHMARK OF THE MATHEMATICAL ACCURACY OF THE TWISTING BEZIER SPLINE CURVES WITH A SIMULATED MANIPULATOR

	Reference	**Coverage**	**Extra**
Nastaleeq			
Rotating Bazier Spline	Original Image	95.8%	5.4%
Machined Output	Original Image	96.7%	7.0%
Machined Output	Rasterized Image	96.7%	4.3%
Thuluth			
Rotating Bazier Spline	Original Image	93.4%	2.8%
Machined Output	Original Image	95.0%	3.4%
Machined Output	Rasterized Image	97.8%	4.4%

[Alternate shorter description] For the sake of analysis, in parallel, the end effector orientation is captured continuously to construct an ink mark with each sweep of the tool. The reproduced image is then used for analysis. The results of such a comparison can be seen in Table II.

VI. CONCLUSION

The accuracy analysis presented in this article demonstrates how rotating bezier splines closely mimic the ink mark of broad edge writing tools thus creating accurate calligraphy scripts. They can be traced on photos of existing calligraphy scripts while also involving the artists to better preserve the artistic spirit. Their resemblance with normal bezier splines makes the learning curve lean for the people who are not accustomed with engineering software and tools. Once traced, they can directly generate machine data for robotic reconstruction and be reproduce the original script on flat and uneven surfaces. Tracing accuracy, ease of use and the ability to generate machine data makes them a strong candidate, not just for Islamic calligraphy but also other broad-edge scripts, to bridge the gap between machines and artists to reproduce and innovate in this field.

Furthermore, rotating bezier splines can be involved in optical character recognition of Islamic scripts. Since they can be regarded as mathematical curves of the actual ink-marks, a modified curve-fitting technique can be developed to find the perfect curve to match a particular calligraphy script. At the least, the open-source proof-of-the-concept editor presented in this research can be mixed with such a curve fitting tool to help fine tune the manual tracing process.

REFERENCES

[1] Web page of Baytulhabeeb, an artist, https://bit.ly/islamic_cal_history, accessed on Sep 4, 2020.
[2] Arabetics™, a private foundry and consulting firm, specializing in Arabic type and lettering designs, “Roots of Modern Arabic Script: From Musnad to Jazm”, https://bit.ly/islamic_cal_history_2, accessed on Sep 4, 2020.
[3] “Islamic Calligraphy”, https://bit.ly/Islamic_cal_wiki, accessed on Sep 4, 2020.
[4] By Julia Kaestle , “Arabic calligraphy as a typographic exercise”, https://bit.ly/cal_styles, accessed on Sep 4, 2020.
[5] Haji Noor Den, Portfolio, http://www.hajinoordeen.com/about.html, accessed on Sep 4, 2020.
[6] Mohammad Zakrya, Portfolio, https://mohamedzakariya.com, accessed on Sep 4, 2020.
[7] Kamel Al Baba, Mokhtar Al Baba, Portfolio, http://www.arabiccalligraphy.com, accessed on Sep 4, 2020.
[8] Abed Yaman, “A look at the history of Arabic calligraphy” https://bit.ly/islamic_cal_stages, accessed on Sep 4, 2020.
[9] M. A.-de Lemos, and E. V. Liberado, “Industrial robotics applied to education”, Proceedings of 2011 International Conference on Computer Science and Network Technology.
[10] Robotics in Agriculture: Types and Applications, https://bit.ly/ind_robots_in_agri, accessed on Sep 4, 2020.
[11] A robot playing table tennis, https://www.kuka.com/timo, accessed on Sep 4, 2020.
[12] A printing robot, https://bit.ly/ind_robot_cal, accessed on Sep 4, 2020.
[13] Project code repository on GitHub, https://github.com/umartechboy/Thesis_2017-MS-MC-17—, accessed on August 8, 2021.
[14] M. Bilal, A. Raza, M. Rizwan, M. Ahsan, H. F. Maqbool, S. Abbas Zilqurnain Naqvi, “Towards Rehabilitation of Mughal Era Historical Places using 7 DOF Robotic Manipulator”, in Proceedings, IEEE International Conference on Robotics and Automation in Industry, pp. 1-6, 2019.
[15] GIMP, GNU Image Manipulation Program Developers Resources, https://bit.ly/gimp_developers, accessed on July 7, 2021.
[16] Inkscape Extensions, https://bit.ly/inkscape_developers, accessed on July 7, 2021.
[17] Wacom Intuos, https://bit.ly/wacom_intuos_official, accessed on July 7, 2021.
[18] Lenovo Thinkpad x380, https://bit.ly/lenovo_thinkpad_x380, accessed on July 7, 2021.
[19] Microsoft Surface Pro 7, https://bit.ly/ms_surface_pro_7—, accessed on July 7, 2021.
[20] George R. R. Martin, *“A Game of Thrones”*, Chapter 30.
[21] *Solving a 6 DoF Robot*, https://bit.ly/SolvingA6DoFRobot, accessed on November 30, 2021.
[22] Microsoft .Net Framework, https://bit.ly/ms_dotnet_framework, accessed on Jan 09, 2022.
[23] Khursheed Gohar Qalam, https://bit.ly/gohar_qalam, accessed on Jan 09, 2022.