**M.Sc Thesis**

**Robotic Reconstruction of Islamic Calligraphy**

Submitted by: Muhammad Umar Hasan (umar.techboy@gmail.com)

Registration No: 2017-MS-MC-17

Registration Date: 22-January-2020

Supervisor: Dr. Ali Raza

Email: umar.techboy@gmail.com



**DEPARTMENT OF MECHATRONICS AND CONTROL ENGINEERING**

University of Engineering and Technology Lahore.

May 2020.

Abstract

[We write the abstract at the very end]

Introduction

Islamic calligraphy is an art having 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 [5-8]. Unfortunately though, the industrial age and the advent of technology has not spared this beautiful art when it claims to provide better alternatives for almost everything related to human beings. Discovery of new facets of calligraphy aside, with the prevalence of modern technologies and resulting lack of expertise in this domain, 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 in-to museums; awaiting to be wiped away slowly with each round of the monsoon and every splash of the ocean’s waves.

Potentially, we can use robotic dexterity to help us in this domain. Industrial robots have already been used outside the industry to do unorthodox tasks [9- 12] and they can surely uplift this art as well. At the very least, they can be employed in restoration and replication of existing calligraphy work [13]. In other words, they can be used as printers, or rather one may say, “painters” that give an extra hand to the calligraphy artists to open up a new dimension of the art that can not only revamp the existing calligraphy sites but also create new art.

Mechanized/robotic drawing of the Islamic calligraphy scripts requires not just the ink-mark information but also the information about the tool movement [3]. Specially, using a flexible flat head brush instead of a solid round tip pen and all that to draw on un-even surfaces, makes the job extremely special indeed. A robot needs to take special care about the orientation and downwards force of the tool as well.

In a nutshell, the main problem can be divided in two major sections. First, transforming the printed scripts into machine data and second, recreating the scripts using a robotic end effector.

To solve the first part, instead of doing image processing, we propose a new way of transforming the existing scripts into machine data; the “Rotation/twisting Bezier Spline”. The idea is to bring real artists in the process. For the new scheme to be fully tested, we wrote a fully featured graphical spline editor and analyzer called “Gregor”. The tool was tested and tuned with the help of multiple real-world calligraphers.

Then we test the trace effectivity of the so-called rotating/twisting splines by taking specimens written in two Qat’s. We use image processing to quantitatively compare the source to the traced script.

In parallel, we developed a simulator of a six degrees of freedom robot, called “Drogon”. The simulator boasts a 3D view finder, lets us change the physical parameters of the robot, provides some useful tools to analyze, test and tune different aspects of the simulator itself, and last but not the least, tools to input scripts developed with twisting Bezier splines in to the simulated world and then quantitatively test how the robot performs given the target.

Last but not the least part of this project is an effort to efficiently document each aspect; the code, software architecture, the software tools, the physics behind the simulator and the areas which one can carry over for further research and improvement. Since not all of such information can be printed, we have set up a public Github repository where one can not only fetch information but also contribute easily to the project.

Table of Contents

[Chapter 1. Rotating/Twisting Bezier Splines 6](#_Toc76428854)

[1. 1. The Necessity 6](#_Toc76428855)

[1. 2. Mathematical Model of a Rotating Bezier Spline 6](#_Toc76428856)

[1. 3. Conversion of Existing Calligraphy Artwork 9](#_Toc76428857)

[1. 4. Machine Data Generation 9](#_Toc76428858)

[1. 5. Characterization 9](#_Toc76428859)

[Chapter 2. The Spherical Robotic Arm 10](#_Toc76428860)

[2. 1. Introduction 10](#_Toc76428861)

[2. 2. The Configuration 10](#_Toc76428862)

[2. 3. Inverse Kinematics 10](#_Toc76428863)

[2. 4. Forward Kinematics 10](#_Toc76428864)

[Chapter 3. Gregor -- The Twisting Bezier Spline Editor 11](#_Toc76428865)

[3. 1. Introduction 11](#_Toc76428866)

[3. 2. Requirements and Features 11](#_Toc76428867)

[3. 3. Usage 11](#_Toc76428868)

[3. 4. Development 11](#_Toc76428869)

[Chapter 4. Drogon – The Robot Simulator 12](#_Toc76428870)

[4. 1. Introduction 12](#_Toc76428871)

[4. 2. Requirements and Features 12](#_Toc76428872)

[4. 3. Usage 12](#_Toc76428873)

[4. 4. Development 13](#_Toc76428874)

[4. 5. Simulation 13](#_Toc76428875)

[4. 6. Rotating Bezier Spline Curves 13](#_Toc76428876)

[4. 7. Areas that Need Improvement 13](#_Toc76428877)

[Chapter 5. The Project 14](#_Toc76428878)

[5. 1. Set Milestones 14](#_Toc76428879)

[5. 2. Expected Outcomes 14](#_Toc76428880)

[Chapter 6. Conclusion 15](#_Toc76428881)

# Rotating/Twisting Bezier Splines

## Introduction

The first part of the main problem is extracting digital machine data from the existing calligraphy specimens. Conventionally, image processing is being used [cite?] to extract data that can be used to create machine data. We, however, propose a difference; no matter how strong and robust image processing gets, we propose that there is no alternative to the minor details only a real artist can observe and recreate. Image processing may even quite perfectly extract the exact tool movement the artisit We bring one little tweak in the conventional Bezier splines that enable them to not only quite accurately mimic a broad edge writing/drawing tool but also stores direct machine movement information. For the idea to be fully tested, we wrote a fully featured graphical spline editor and analyzer called “Gregor”. The tool was tested and tuned with the help of multiple real-world calligraphers.

## The Necessity

One potential way to solve the modeling issue of calligraphy involves using the existing digital calligraphy fonts. There are, however, two critical issues involved with this scheme; one is the need of an algorithm that will convert the font data to robot movement data and the other is the lack of a font variety. Additionally, working with fonts leaves a narrow space of modifying the scripts to look like artistic scriptures. This is the primary reason we must not use the existing digital fonts.

Keeping in mind the gaps left by the digital font, another solution to this problem is in the discovery of a new way to unify ink-mark information of digital Islamic script and tool movement performed by the artist. Making a mathematical model to learn the drawing tool information just from the printed text is quite a complex job. Instead, only if we could form a way an artist can give digital input, this problem can be overcome.

This is where the twisting Bezier splines come into play. We add a twist/rotation handle in the conventional Bezier spline curves and that is it. The, however, remains unanswered, what is a Bezier spline curve in the first place?

## Mathematical Model of a Rotating Bezier Spline

### The Conventional Bezier Spline

In order to describe how twisting splines work, lets first look into the working principle of a conventional Bezier spline. Figure 1 shows an illustration of a spline path made up of several sub curves. Each curve section is only partly independent of the other. Figure 1 (a) shows the final shape of the curve without any construction elements. In Figure 1 (b), we explode different sections of the curve into smaller elements and show how they fit together to form the complete spline. There five sections in this curve labeled 1 through 5. Figure 1 (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 is connecting the sections 1 and 2. Like all the other anchors, this angle also has two handles, and , connected through a straight line passing through the anchor. The length of each handles, and 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 section become partly independent. For instance, handle contributes to the shape of curve segment 1 and 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 1 (d) focuses on section 4 and 5 of the curve and also shows a polygon defined by the points , , and . It must be noted that the points and of this polygon are also the anchor point between sections 3, 4 and 5. Take section 4 for example here. The polygon mathematically defines the complete shape of this polygon. If is point on the section 4, with coordinates and in a cartesian plane with some origin, it is defined by the following equation.

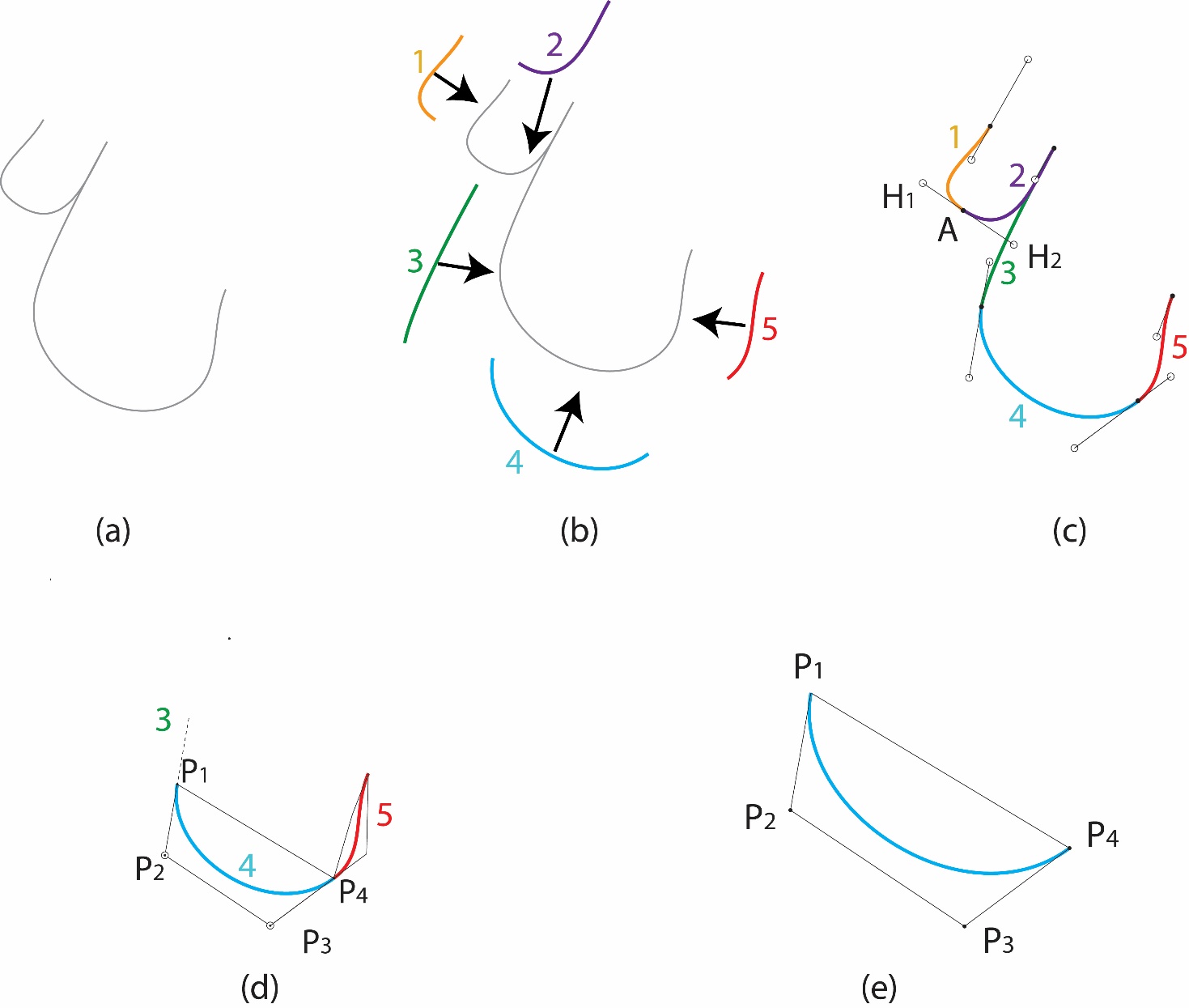


Figure 1. An illustration showing the construction of Bezier Spline Curve. (a) A sample of a Bezier spline path (b) an exploded view of inner curves of the Bezier spline path (c) Handles that control the shape of the two adjacent sub curves (d) and (e) Construction polygon of the sub curve.

where,

and

where,

and

for an f in the range [0, 1].

It can also be proved that the side segments of the polygon and are tangent to the curve at the point the meet it at and respectively.

### Twist/Rotation Handle

On top of the conventional Bezier splines, that work around anchor points that have curvature handles, we add a “Rotation”/”Twist” handle in the anchor and a thickness parameter to the whole curve. A rotation handle is like the curvature handle discussed earlier, except it does not have any effect on the shape of the curve. The thickness parameter defines the size of a flat line centered on and sweeping on it. The orientation of this sweeping line is the same as the angle between the twist handle and the respective anchor. See Figure 2 (a) that 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. We then add thickness to the curve in Figure 2 (b). The resulting curve may look a little out of order but it is normal. This is because the rotation handles are lying on their default position. The twist handles may be given some length but it is insignificant since the twist of the curve will only take the value of the angle the handle subtends about the anchor.

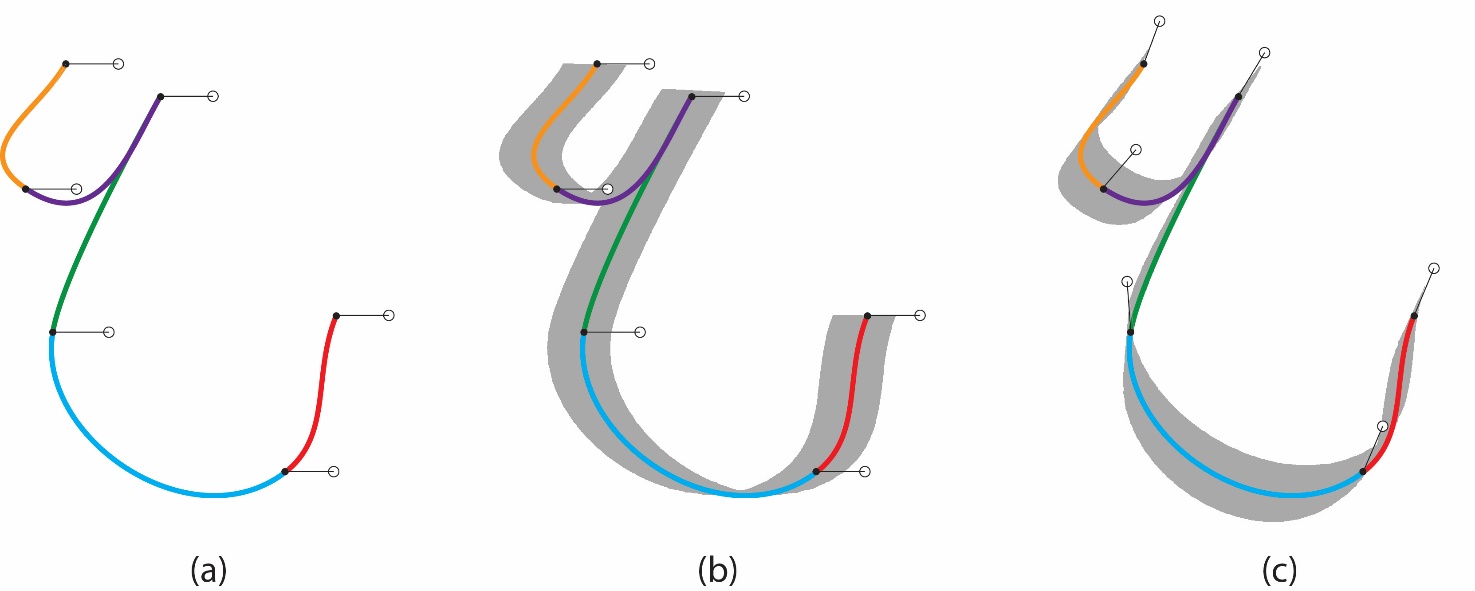


Figure 2. Adding a Twist/Rotation handle in the Bezier spline curves. (a) a Rotation handles added to the anchors without a curve thickness. (b) Thickness added without adjusting the twist yet (c) a roughly edited twisted Bezier spline curve.

In simpler words, its similar to sweeping a pen centered on the actual spline while twisting it uniformly and continuously about its own axis according to the equation

where f is the same factor that was used to define and and are the angles between the first and the second anchor and their rotation handles respectively. 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 later. This visually hides the 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 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. When the idea of the twisting splines was first conceived, it wasn’t envisaged that the angle had to be taken like in this scheme. Special care had to be taken in order to graphically read a continuous angle from the user.

See Table 1 in Appendix which compiles the rectangular coordinates of all the anchors of the rotating bezier spline shown in discussed in this example. Also, see code snippet 1 in appendix which shows the contents of a file that can be used to recreate the spline. In chapter 3, we will discuss “Gregor”, the tool that uses this data to save the created splines.

## Conversion of Existing Calligraphy Artwork

Instead of using image processing to try to extract data from existing scans and photographs of the artworks, using rotating Bezier splines we can now include the artists in the process. Just like any other computer-based graphics design application, either we can write a rotating Bezier splines curve editor plugin for an existing open-source application like GIMP [14] or Inkscape [15]. Unfortunately, the later is not a suitable option because the support for the plugins and extensions for both of these poplar software only lets the developer work with the image saving and processing, they don’t let us play with the behavior of the workspace which would be needed to convert the conventional spline tool into a rotating Bezier spline editor. It can still be done by modifying the source code and building the applications from the scratch.

With the second option not viable anymore, we are left with only one option. Writing our own tool to create, modify, save, and reload rotating Bezier splines. Like any other application, for it to be called a “Software”, we also develop some comprehensive documentation discussing the working and behavior of the tool. Although we will discuss in detail about the tool in chapter 3, the fundamental problems it must solve are

a) easy to use interface

b) converting the existing photos to digital form and,

c) generating machine data that that encapsulates the pen rotation information along with other positional and speed information.

## Machine Data Generation

The rotating spline curves are themselves an emulated ink-mark of a broad edge marking tool. This is the reason extracting machine data and even G-codes from them becomes natural. If the flat side of the tool is assumed to be entirely touching the writing surface, the minimum information required to draw a stroke trickles down to the line on which the pen must move and the twist of the pen in world coordinates. This is exactly the rotation Bezier splines can contain once an artist has drawn them on the computer screen. In other words, in order to call the rotation Bezier splines the machine data, the following assumptions must be made:

1. The flat tip of the pen is always completely touching the drawing area.
2. 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 and is set by the machine.
3. To produce thinner strokes, another spline will be used. This means that the machine would have to use multiple tools for such splines.
4. The axial pressure the pen inserts on the drawing board while drawing is also fixed and the is set by the machine.

It is now obvious that in order to remove the limitations of fixed angles and pressure values, one can add more handles similar to the rotation handle. A set of by directional inclination handles can be added right away with a three-dimensional pen position visualizer to assist the artist determine what angle they want to keep the pen at while drawing a specific stroke. The pressure angle, however, would not be recommended without interfacing some hardware that lets artist actually feel the pen pressure in real time before setting a handle value. This can be done using a pressure sensitive digital pen or writing tablets [16-18].

There is more detail on how the spline data is converted into machine data in the coming chapters.

## Characterization

An important aspect of fabricating a new technique is measuring how well it performs in different usage scenarios. The problem is, in terms of arts, not every mistake the technique makes can be regarded as an issue. Developing a 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 project. However, there are some aspects that we have tried to measure that gives us some idea how effective the rotating Bezier splines can be. These measurable metrices include:

|  |  |
| --- | --- |
| **Metrics** | **Result** |
| Percentage of area outside the original bounds | Less than 2% |
| Percentage of area covered | Better than 94% |
| Maximum lateral deviation of the Bezier path from the pitch line | N.A. (This list was planned in the synopsis but is no longer valid given the nature of fabricated splines.) |
| Total number of compatible scripts | Broad edge scripts of all languages |

Please note that the third metrics in this list was planned to be used but is no longer valid given the nature of fabricated splines.

There are also some other metrices that were not measured because of lack of resources and because they required testing the tool with a large group of actual artists.

|  |  |
| --- | --- |
| **Metrics** | **How it can be measured** |
| Easy of usage | A survey based on Likert scale |
| Time efficiency of tracing an existing specimen. | Comparison of the time taken by the same artists tracing with conventional and rotating Bezier splines |
| The artistic quality of the specimens produces. | A survey based on Likert scale and filled by a wide range of artists |

### Sample Results

### Performance with Different Fonts

# The Spherical Robotic Arm

## Introduction

## The Configuration

## Inverse Kinematics

## Forward Kinematics

# Gregor -- The Twisting Bezier Spline Editor

## Introduction

## Requirements and Features

## Usage

### Interface

### Drawing and Modifying a Spline

### Save, Load, Export

### Short-keys

### Other useful features

### Analysis Tools

## Development

### Code Organization

### The BezierBoard Class

### RBSPoint

### Spline Elements

### Miscellaneous Helper classes and form

### Areas needing Improvement

# Drogon – The Robot Simulator

## Introduction

## Requirements and Features

### 3D Visualizer

### Workspace Optimizer

### PID Tuner

### Rotating Spline Importer

## Usage

### Interface

### Manual Control

### Analysis Tools

## Development

### Code Organization

### The SphericalRobot Class

## Simulation

### Assumptions and Limitations

### Accuracy of simulated Quantities

### Forward Kinematics

## Rotating Bezier Spline Curves

### Sample Results

### Accuracy

## Areas that Need Improvement

# The Project

## Set Milestones

## Expected Outcomes

# Conclusion

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](https://mohamedzakariya.com/), accessed on Sep 4, 2020.

[7] Kamel Al Baba, Mokhtar Al Baba, Portfolio, [http://www.arabiccalligraphy.com](http://www.arabiccalligraphy.com/#!/mokhtar), 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] 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.*

[14] GIMP, GNU Image Manipulation Program Developers Resources, <https://bit.ly/gimp_developers>, accessed on July 7, 2021.

[15] Inkscape Extensions, <https://bit.ly/inkscape_developers>, accessed on July 7, 2021.

[16] Wacom Intuos, <https://bit.ly/wacom_intuos_official>, accessed on July 7, 2021.

[17] Lenovo Thinkpad x380, <https://bit.ly/lenovo_thinkpad_x380>, accessed on July 7, 2021.

[18] Microsoft Surface Pro 7, <https://bit.ly/ms_surface_pro_7>, accessed on July 7, 2021.

Appendix

Code Snippetes

Snippet 1: Sample code of a rotating Bezier spline that will render the Urdu letter “ع” in Nastaleeq.

<spline>

<FlatTipWidth>150</FlatTipWidth>

<Color>-5658199</Color>

<anchor>

<rotationoffset>0</rotationoffset>

<P>-198.3791, 452.6993</P>

<C1>-131.6351, 572.4461</C1>

<C2>-265.1234, 332.9534</C2>

<R1>-148.3791, 452.6993</R1>

</anchor>

<anchor>

<rotationoffset>0</rotationoffset>

<P>-296.5323, 156.2775</P>

<C1>-439.8357, 254.4304</C1>

<C2>-119.5302, 35.04326</C2>

<R1>-246.5322, 156.2775</R1>

</anchor>

<anchor>

<rotationoffset>0</rotationoffset>

<P>25.40986, 374.1774</P>

<C1>-47.22344, 262.2825</C1>

<C2>98.04301, 486.0714</C2>

<R1>75.40986, 374.1774</R1>

</anchor>

<anchor>

<rotationoffset>0</rotationoffset>

<P>-233.7143, -183.332</P>

<C1>-208.1945, -28.25013</C1>

<C2>-274.7982, -432.9961</C2>

<R1>-183.7143, -183.332</R1>

</anchor>

<anchor>

<rotationoffset>0</rotationoffset>

<P>315.9428, -517.0526</P>

<C1>95.77186, -679.5702</C1>

<C2>435.6645, -428.6809</C2>

<R1>365.9427, -517.0526</R1>

</anchor>

<anchor>

<rotationoffset>0</rotationoffset>

<P>441.5787, -144.0708</P>

<C1>388.576, -277.5591</C1>

<C2>494.5813, -10.58265</C2>

<R1>491.5787, -144.0708</R1>

</anchor>

</spline>

Tables