Digitization and Robotic Reconstruction Broad-Edge Calligraphy Using Twisting Bezier Splines

Muhammad Umar Hassan, Dr. Sabieh Anwar, Dr. Ali Raza

Abstract

Bezier curves can trace the outline of digital fonts and calligraphy scripts written with and without a broad-edge tool with acceptable accuracy. However, this curve data is only good enough for rendering the scripts on computer screens and photo printers; the data needs complex algorithms that can convert it into a format which is usually required by a robotic manipulator equipped with a conventional broad edge tool. This work introduces a new kind of bezier spline that can not only be manipulated and rendered on the screen as intuitively as other splines but bridges the gap between digital vector data and machining data by altogether removing the step where outline vectors or images are artificially analyzed to estimate the broad-edge tool movement on a pitch line. These rotating bezier splines are first characterised using two scripts of Islamic calligraphy to compute a performance metrics. The created scripts are then tested using visual simulations of a robotic manipulator that writes the scripts virtually on a piece of paper. It is believed that with further contribution and adoption, this research can be considered a major milestone in mechanising broad-edge calligraphy.

Contents

I. INTRODUCTION

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, printing is not always a choice; in many cases, it is desired that a script is written using a conventional tool by a robotic manipulator [cite] in a similar fashion a human artist would have used in order to preserve the essence of artistic norms [cite]. Separate techniques [cite] are needed to convert output of the outline or pitch curve functions to a format that a robot can directly use to for the required tool movement. Even though many of these techniques [cite] can promise accurate tracing, none of them can promise an exact tool movement that a human would like to use. Also, once the machining data is produced the idea to manually fine tune is an intuitive one. Twisting bezier splines bridge this gap by introducing a small but important innovation in the conventional bezier spline curves. Instead of working as outline curves, they directly record the pitch line of a twisting broad-edge tool stroke along-with the twist information of the tool. Their usage is just as intuitive as the conventional bezier spline curve and the information they poses 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 the 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. A user can physically relocate these curvature handles 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. Although these functions are continuous, there output can easily be discretized as

closed paths made up of closely located two dimensional points with controllable resolution. 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, the rotating bezier splines 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 way to interpret them is to consider them as a pitch line for a thick 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 to fill in the outline of the glyph using a 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, for a particular glyph created with a broad edge, the twisting bezier spline curves 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 we call the "Rotation/Twist" handle. Just like the curvature handles represent the curve function inputs responsible to define the curvature, the rotation handles represent the inputs that control the twist of the simulated broad-edge tool. Just like the conventional bezier splines, the functions of the twisting bezier splines can also be discretized and converted into a list of two dimensional points, a closed path to emulate the ink-mark of the broad edge tool, needed by the computer display drivers. This is how an artist can intuitively use the twisting curves not just to trace but also to create calligraphy that is not bound by any culture or language.

The interesting part is that, logically, the twisting beizer splines are more near to the machine than they are to computer graphics driver. 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 an actual tool that can be mounted on a robotic manipulator. The spline directly controls the position and twist of the tip of the tool which can directly be translated into machine movement codes. The rest of the process is inverse kinematics

and is already handled by the robot controller.

The article is divided in 5 sections. After the introduction, Chapter II presents the working principle and mathematical model of the twisting/rotating bezier splines. In section III we discuss some performance metric and discuss the tests performed to gauge the performance of the twisting bezier splines. Section **??** discusses the simulation of a robotic manipulator to verify the results. We finally conclude in Section IV. Also see Appendix **??** and **??** which shows a user manual of the software tools written and used in this research, Appendix C which is a digital copy of the source codes, this thesis and some videos on how to reuse the tools and the code written.

II. ROTATING/TWISTING BEZIER SPLINES

A. Introduction

As discussed in section **??**, the first part of the main problem is extracting digital machine data from the existing calligraphy specimens. Conventionally, image processing is being used 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 maintain that there is no alternative to the minor details only a real artist can observe and recreate. So a solution is needed that fulfils the technical needs as well as the artistic demands.

B. Digital Script Data

An important question while the modeling a calligraphy scripts is the choice of a digital data format that will hold the extracted script data. One potential answer to this question 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. Another question however arises: Let alone a Twisting Bezier spline, what is a (conventional) Bezier spline curve in the first place? The following sections now answer this question in detail.

C. Mathematical Model of a Rotating Bezier Spline

1. The Conventional Bezier Spline

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 are 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 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 handles, $\overline{AH_1}$ 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 1 (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 point between sections 3, 4 and 5. Take section 4 for example here. The polygon mathematically defines the complete shape of this curve part. If P_{spline} is a point on the section 4, with coordinates

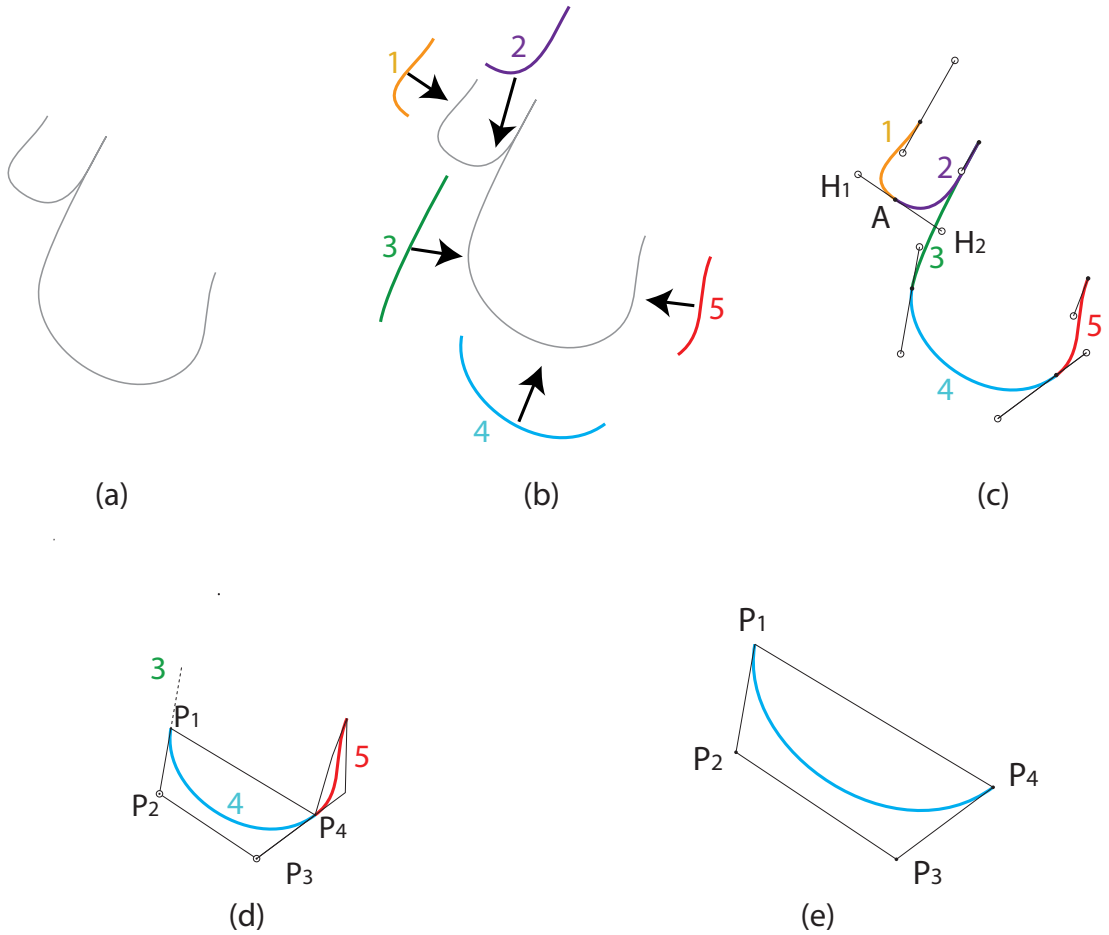


FIG. 1: An illustration showing the construction of Bezier Spline Curve. (a) A sample of a Bezier spline path (b) an exploded view of inner curves of the Bezier spline path (c) Handles that control the shape of the two adjacent sub curves (d) and (e) Construction polygon of the sub curve.

x and y in a cartesian plane with 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

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

where,

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

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

and

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

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

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/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 segment centered on P_{spline} 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 (c) shows the final form of the twisting bezier spline after the twist handles have been iteratively moved to position that give the spline the desired look.

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

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

where f is the same factor that was used to define P_{spline} and θ_A and θ_B are the angles between the first and the second anchor and their rotation handles respectively. It may be

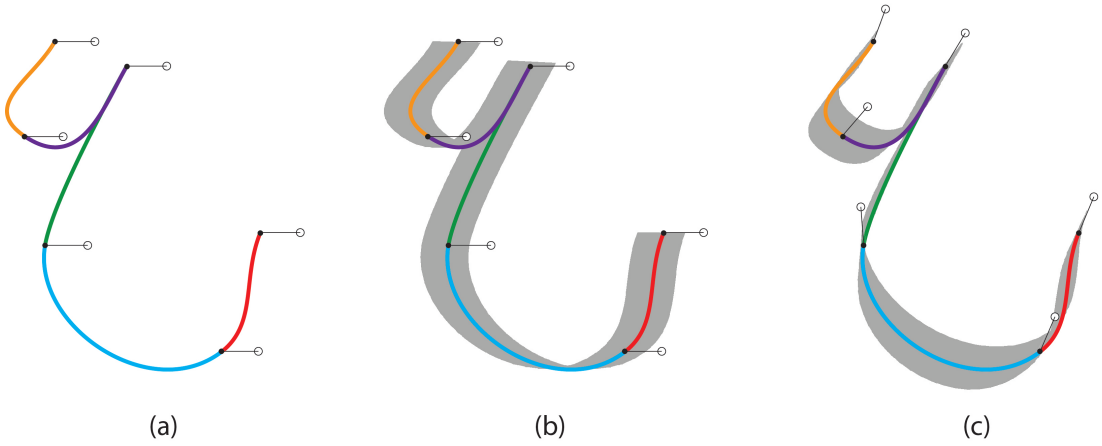


FIG. 2: A normal bezier spline is given the rotation handle. (a) rotation handles are shown on the anchors. (b) The rotation handles are given a visible thickness value. (c) The rotation handles are moved to give the spline the desired shape.

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 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. 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 Appendix D which compiles the rectangular coordinates of all the anchors of the rotating bezier spline shown formatted in a manner that Gregor can interpret. In chapter A, we discuss in detail "Gregor", the tool that uses this data to save the created splines.

D. 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 which is way out of the scope of this thesis.

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. The fundamental problems it must solve and the features it must have are:

- easy to use interface
- converting the existing photos to digital form and,
- generating machine data that encapsulates the pen rotation information along with other positional and speed information.

Keeping in view these requirements we created "Gregor", the first tool to edit, modify and create rotating Bezier splines. See chapter Also see Appendix **??** and Appendix **??** which discusses in detail about this tool.

E. 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. 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. Minus the information about a three dimensional reference system, this is exactly the information a rotating Bezier spline contains once an artist has drawn them on the computer screen. In other words, to call the rotation Bezier splines the 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 and is set by the machine.
- To produce thinner strokes, another spline will be used. This means that the machine would have to use multiple tools for such splines.
- The axial pressure the pen inserts on the drawing board while drawing is also fixed and is set by the machine as well.

It is now obvious that 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 the artist 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]. These expansions alone are in themselves worthy of research projects that rivals the scope of the current one.

III. PERFORMANCE AND BECHMARKING

A. 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 give us some idea how effective the rotating Bezier splines can be.

B. Supported Scripts

By "supported scripts" one may imply deriving a mathematical model for a particular font or a scipt family. The mainline scripts used in calligraphy are not necessarily as mathematical

as the model of the rotating Bezier splines. Especially, when the artists start to utilize their writing tool in unique ways to extract some unique value from the scripts they create, forming a mathematical model becomes practically impossible. However, since in the first place, it would be an artist who will be creating and tracing scripts on the screen of a computer, it is safe to claim that given the similarity of the emulation, rotating Bezier splines can be used to produce any script that is written with broad edge tools. However, these are some limitations inherited by this statement:

- If the tool changes thickness during a stroke (like a flexible brush), 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.
- Although the rotating splines have a defined tool width, we still assume the tool to be infinitely thin on the other side, more like a narrow line. This makes negligible but still some difference when the virtual tool is replaced with an actual tool. One way to overcome this issue would to come up with another rendering algorithm that also asks for this missing information. This has been discussed in later chapters when we suggest some other improvements in the overall project.

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 produced by tracing the processed images. One-to-one pixel comparison was made using computer scripts to measure the fraction of ink that maps exactly on the original script, lies outside it or is completely missing. Table I presents some values that give some idea of efficacy of the proposed bezier splines.

This metrics is comprehensive but still not complete. Some additional metrices are still needed to give a verdict about how good the proposed solution is. Table II presents a couple of those metrices that may also be desired by the researchers.

Please note that the third metrics in Table II was planned to be used but is no longer valid given the nature of fabricated splines. There are also some other metrices that were

Metrics	**Results**
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
Total number of splines measured	> 100
Total pixels compared	9.9 million
Tested scripts	Nastaleeq, Thuluth

TABLE I: Benchmark of the mathematical accuracy of the twisting bezier spline curves

Metrics	**How can it 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

TABLE II: Advanced metrices to gauge the effectiveness of twisting bezier splines.

not measured because of lack of resources and because they required testing the tool with a large group of actual artists.

2. Sample Results

As 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 and other in Thuluth. Figure 3 shows a detailed comparison of a part of a famous script produced by Gohar with the version traced with twisting bezier splines. The comparison unveils how accurately the bezier splines appear to be tracing the

original script. The original appearing in Figure 3(a) image is first processed to be used with the spline editor and then converted to a binary image which is shown in Figure 3(b). Figure appearing in Figure 3(c), (d) and (e) are then produced by rasterizing the traced spline and a one-to-one pixel comparison of both images. See Table II which shows a quantitative measure of the trace accuracy. With these values, it is fair to assert that the twisting splines are very accurately mimicking the original sample.

However no matter how accurate they look, it must also be noted that since the original image is not available in a sufficiently high digital resolution, strictly speaking, Figure 3(d) and (e) do not do not show the true picture of the error that the curves produce but only a reasonably good estimate. The final verdict about the accuracy can only be given in form of the appeal this technique gets when actually presented in the community.

Figure 4 and 5 show sample outputs produced by tracing two of famous Gohar's articles in "Nastaleeq" and "Thuluth" scripts. The twisting splines do contain vector data and have no resolution but for the sake of presentation, the images have been rasterized by "Gregor".

3. Machine Data Generation and Simulation

To check how accurately the data generated by these splines can be traced by an actual robotic manipulator, computer simulation was used. In section **??** we discuss in detail "Drogon", the simulator used to simulate and analyze the splines with a 6 DoF manipulator.

C. Simulation

Using the same model of the spline model, the simulator first rasterizes the whole spline and then transforms the two dimensional points onto a plannar surface simulated on a user desired position in the workspace. Then it creates a pseudo machine code to mimic the tool changes required before each stroke and the transformed rasterized points are directly considered as G-Codes for the manipulator. Once computed it starts implementing the program.

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

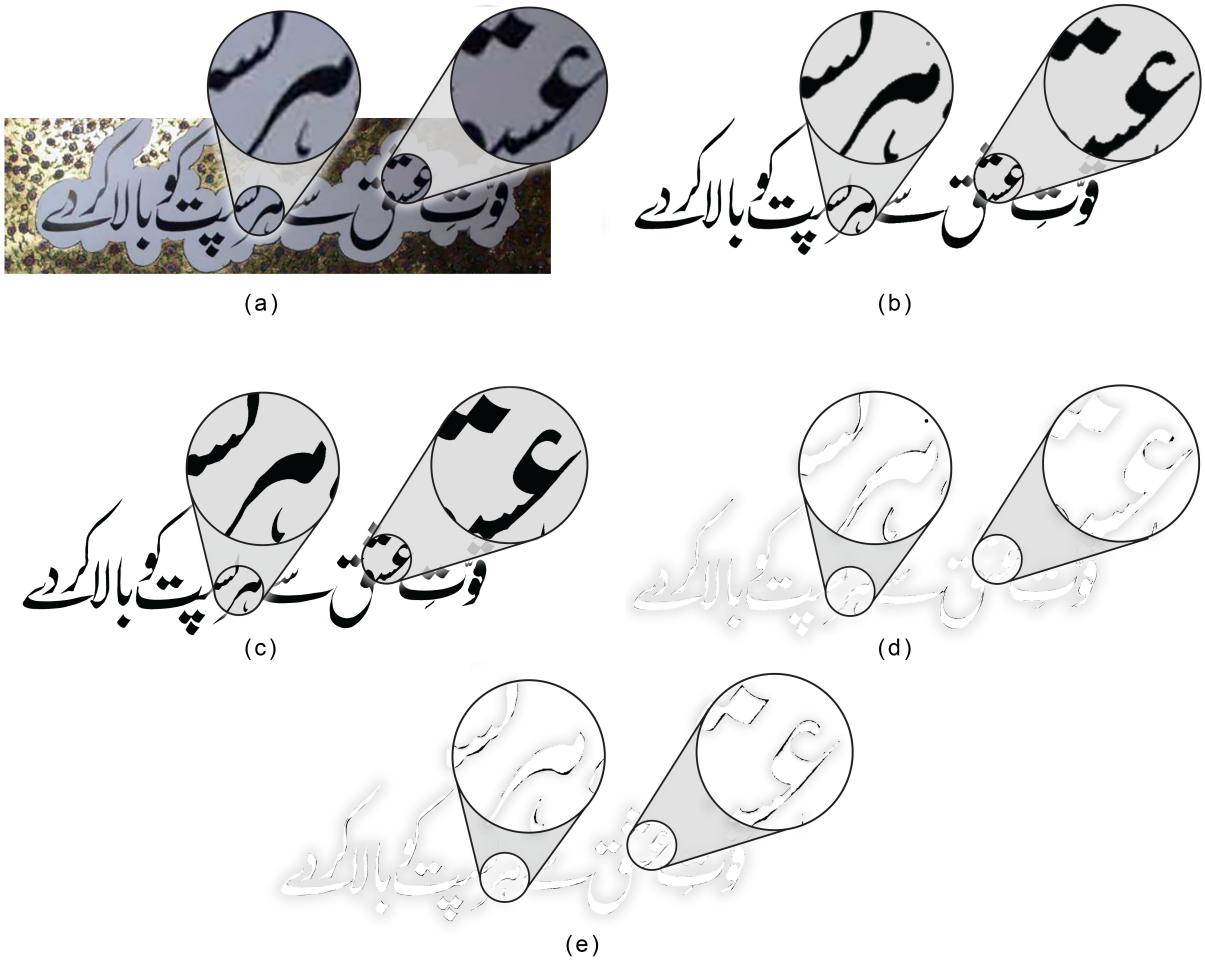
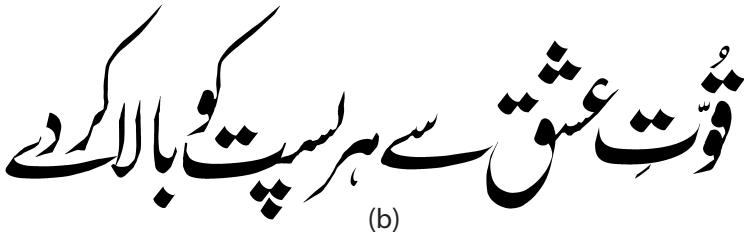


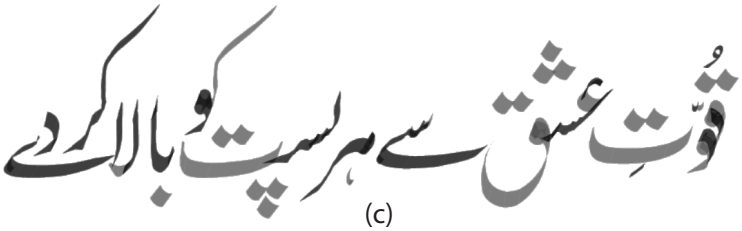
FIG. 3: A test comparison of the original specimen of Gohar's script and the one traced with twisting bezier splines. Each image includes magnified version of two typical areas of the script for better visibility. Also, in (d) and (e) a light shadow is added for better understanding and is not a part of the scripts. (a) Photo of the original script. (b) Inkmarks extracted using photoshop from the original image. (c) Rasterized inkmarks generated by twisting bezier spline curve. (d) Area of the original ink missing in the twisting splines. (e) Area of the ink marked outside the original ink.

analysis. The results of such a comparison can be seen in Table III. The value indicate that the robot very closely reproduces the given spline thus verifying the thesis of the research.

(a)



(b)



(c)

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.

	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%

TABLE III: Benchmark of the mathematical accuracy of the twisting bezier spline curves with a simulated manipulator

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.

FIG. 6: A preview of the calligraphy produced using the rotating bezier splines.

IV. CONCLUSION

The twisting Bezier spline curves very closely mimic the ink mark of broad edge tools thus creating very accurate calligraphy scripts as seen in Figure 6. With most of broad edge scripts, on an average, they give more than 95% coverage of the reference image an less than 5% overdraw. This has now been verified by comparing them with original calligraphy specimens as well as after simulating the output of a robotic manipulator.

With tools created as a by-product of this research, an artist can not only trace existing calligraphy scripts, but also create and modify new ones with a very lean learning curve. The ease of use of the tools created for the tast assures that the focus of the artist is more on the art itself than the caveats of the software solution.

Where the accuracy of the splines has been characterised and ease of use has been demonstrated, there still are a lot of unexplored areas that require more research. As discussed in section II E, one can pack the tool inclination and normal pressure information into the rotating splines. One can also choose to implement other kinds of manipulators and actuators to test the performance of the splines in the simulator. Last but not the least, since the simulator can emulate a pseudo robot, it can also be modified to be used as a live controller for a real robotic manipulator.

While this thesis puts together the results of some crucial tests to establish the usefulness of rotating splines, the final verdict will still be given by the community that takes the work forward in more scenarios and conditions.

[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] 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.

[19] George R. R. Martin, *"A Game of Thrones"*, Chapter 30.

[20] Project code repository on GitHub, https://github.com/umartechboy/Thesis_2017-MS-MC-17—, accessed on August 8, 2021.

[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.

APPENDIX A: GREGOR – THE TWISTING BEZIER SPLINE EDITOR

1. Introduction

"Gregor" is an open source software developed specifically as a minimalist editor for the twisting bezier splines. A complete user manual be viewed online [cite] but the main features are discussed here. The most fundamental user requirements are very simple.

- The most fundamental requirement was that the tool be able to let the user graphically draw a rotating bezier spline. It was not only convenient but also logical to make the editing sequence similar to other vector editing software. This will make the transfer easier for people who already have some experience in other applications.
- The application must be able to save and load the edited work using a data file.
- The user should be able to drag and zoom the view port using the mouse cursor and keyboard shortcuts.
- There should be a provision to load images into the workspace so that they can be traced.

2. Overview of the implementation

Gregor provides the minimal functionality to work with twisting bezier splines. Additionally, many of the secondary features were implemented to fulfil the needs of actual artists

who found the basic interface too wanting. Microsoft .Net framework was chosen to construct the GUI application mainly because it supports writing code in Microsoft Visual C Sharp (C#) and because all the intended features like mouse and keyboard interactions and graphics are easy to implement in it. Gregor can create, modify, import and save twisting splines. it mainly consists of a single *Windows Form* that hosts the basic editing and viewing controls and an interactive workspace. The workspace is a virtual page that can be zoomed into and panned around using the cursor and keyboard shortcuts. One can change the viewing modes of the workspace to better suite the editing needs. One can choose to display or hide certain anchors, ink-marks, curvature spline, and change the appearance of a twisting spline. To assist the user, Gregor also allows to import and scale images that can be used to trace twisting bezier splines.

See Appendix **??** that describes the interface, enlists the features and describes the usage of most important parts of the application.

3. Interface

The view of the software is the spline editor with a detailed main menu as shown in Fig. 7, document summary and a list of most commonly used toggle buttons. The application also provides some keyboard shortcuts for the most frequently used toggle options like changing the visibility of different elements and toggling the editing modes.

The information a rotating spline contains is too much to be viewed simultaneously. The center point of the anchors, the curvature handles, twist handles, the curve, the inkmark and the background image, when displayed simultaneously is just chaotic. On top of that, when all of these elements are interactive, using a single mouse cursor to interface becomes a headache. This is why the application presents viewing and editing modes.

a. The Viewing Modes

The viewing modes can be controlled using options (g1) through (g3) as shown in Fig. 7, the "View" menu (a2), or the keyboard shortcuts. There are essentially three modes of view.

- (g1) is ink and curve mode. In this mode, the splines curves and the selected handles

will be shown alongwith the ink marks.

- (g2) is ink-only mode. In this mode, the curvature handle of the splines alongwith the anchors and the handles are hidden.
- (g3) is curve-only mode. The inkmark will be hidden in this mode and only the curve alongwith the selected handles will be visible.

it is obvious that only one mode can be activated at a moment and it can be selected either from the options (g1) through (g3) in Fig. 7 or through the View menu. Instead of clicking on the toggle buttons, using the keyboard shortcuts can sometimes be even more convenient.

b. The Editing Modes

The editing models enable or disable the anchors, curvature handles and the twist handles. As shown in Fig. 7 (h1) through (h3), the editing mode toggle buttons can be used to enable or disable any of these handles. Just like the editing modes, these modes can also be controlled using the keyboard shortcuts. Unlike the viewing modes, however, these modes can be enabled all at a time. It must be noted that while in ink-only mode, none of the editing modes will have any effect of the usability of the editing handles.

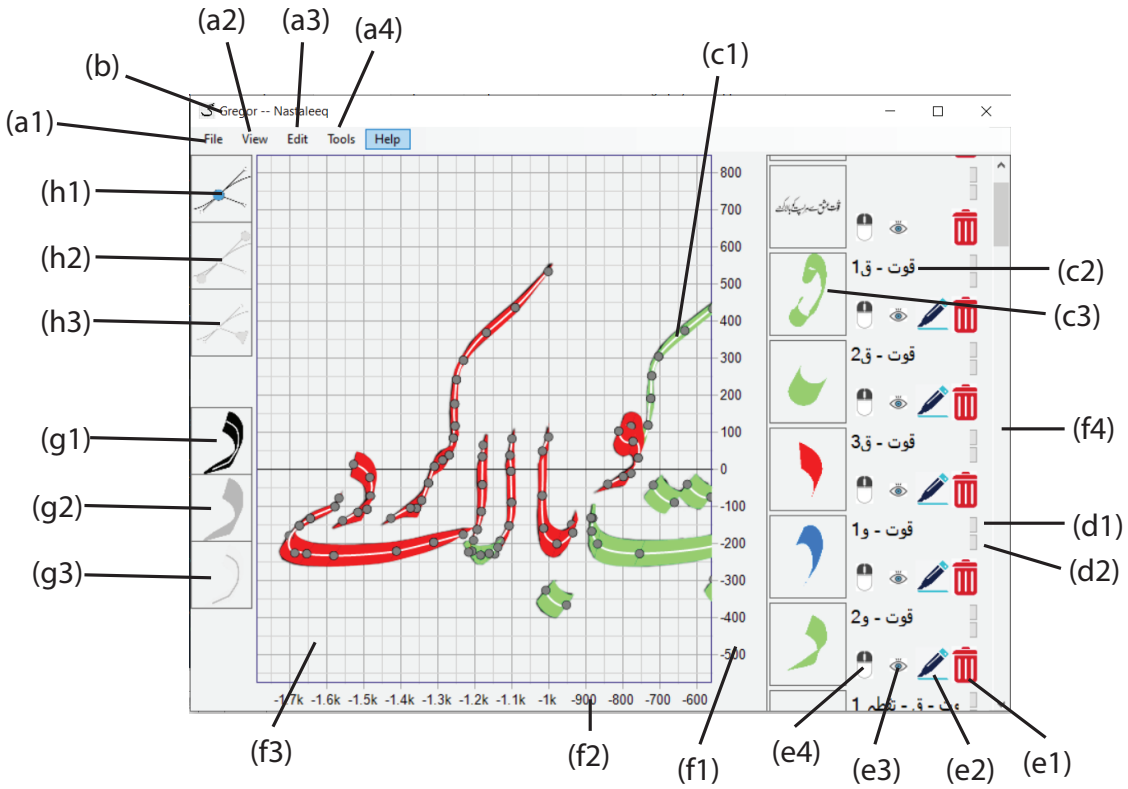


FIG. 7: (a1-a4) The main menu options: (a1) Contains the options to open, save, import, export and clear splines and background images, (a2) enlists some viewing options; the ink viewing modes, visibility different elements of the workspace, visibility of background images, opacity of the ink-marks and rendering algorithm (a3) contains options related to editing on the workspace. It has the option to change the spline editing modes, the behaviour of left click on the workspace, and whether the splines can be dragged or not. (a4) has some analysis tools. (b) is the name of the currently open document. (c) is the main view of each spline in the document, (c2) and (c3) are the name of each spline element and its thumbnail respectively. (d1) and (d2) change the vertical order (z order) of the elements. While hovering the cursor over overlapping splines, the spline with higher position in the list will receive mouse event earlier. (e1), (e2), (e3) and (e4) are used to toggle editability, change visibility, modify color and thickness and delete the respective spline curve respectively. (f1) and (f2) are x and y axis of the workspace. (f3) is the main viewport. (f4) is the list of all the splines and images in the document. (g1) switches the view mode to both spline and inkmark, (g2) changes the viewing mode to ink only. (g3) changes the viewing mode to spline only. (h1), (h2) and (h3) toggle the visibility of center of anchor, curvature handle and the twist handle of the splines.

@@ -1,465 +0,0 @@

APPENDIX B: TABLES

a. Gregor – Keyboard Shorcuts

Key	Action
The File Menu	
Ctrl + O	Open an existing document
Ctrl + S	Save the document to the last selected file or a new file if no previous file is associated.
Ctrl + Shift + S	Save the data in a new file
Ctrl + I	Import spline data from file
Ctrl + E	Open up the export menu
Shift + Del	Clear the workspace
Alt + F4	Close the application
The View Menu	
Ctrl + 4	Combined Mode; Shows both ink and curves
Ctrl + 5	Ink Only Mode; Shows Only the ink marks and hides the curve alogwith the handles
Ctrl + 6	Splines Mode. Hides the ink marks
Ctrl + G	Toggle the visibility of the grid
Ctrl + Shift B	Toggle the visibility of the background images
The Edit Menu	
Ctrl + 1	Toggles the splines anchor centers
Ctrl + 2	Toggles the splines curvature handles
Ctrl + 3	Toggles the rotation handles
Ctrl + V	Toggles the action of mouse left click between adding the anchor and dragging the workspace.
Help	
F1	Shows quick help
F2	Opens up the project Git.

APPENDIX C: SOURCE CODE

APPENDIX D: CODE SNIPPETS

```
//Sample code of a rotating Bezier spline that will render the Urdu letter
<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>
```

APPENDIX E: IMAGES

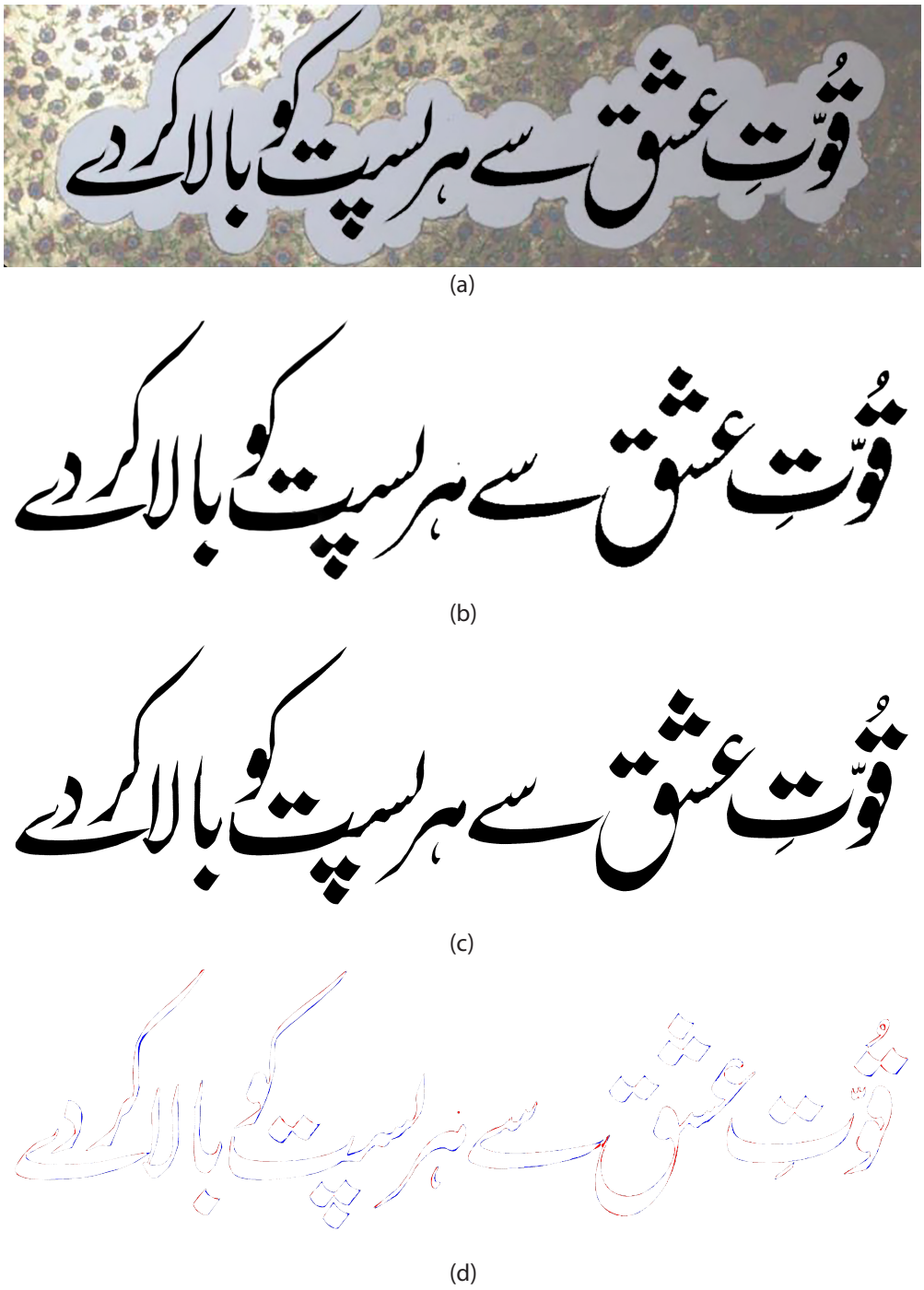


FIG. 8: Nastaleeq sample by Gohar Qalam. (a) Original calligraphy photo. (b) Original photo processed for analysis. (c) Traced rotating bezier spline ink. (d) Difference between (b) and (c). The red pixel indicate the portions that are missing in (c) but are present in (b) and the blue ones show the missing pixels in (b) but are present in (c).

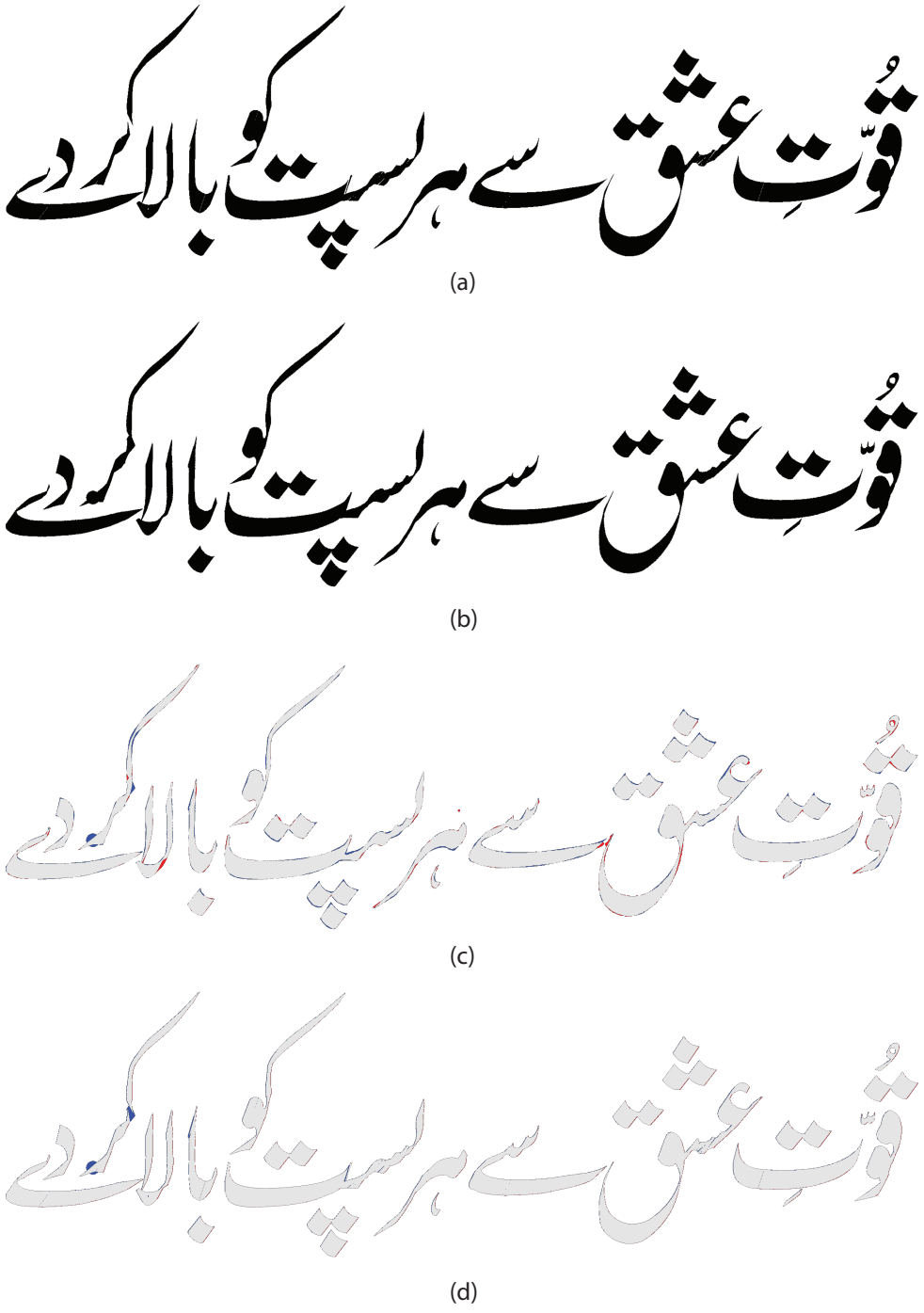


FIG. 9: Machined Nastaleeq sample by Gohar Qalam. (a) Rasterized rotating bezier spline for machining (b) Ink marks machined by a simulated robotic manipulator. (c) and (d) are differences between simulated ink mark and the rasterized photo and the processed original photo respectively. The red pixel indicate the portions that are missing in (c) but are present in the reference image and the blue ones show the missing pixels in reference but are present in the ink mark.

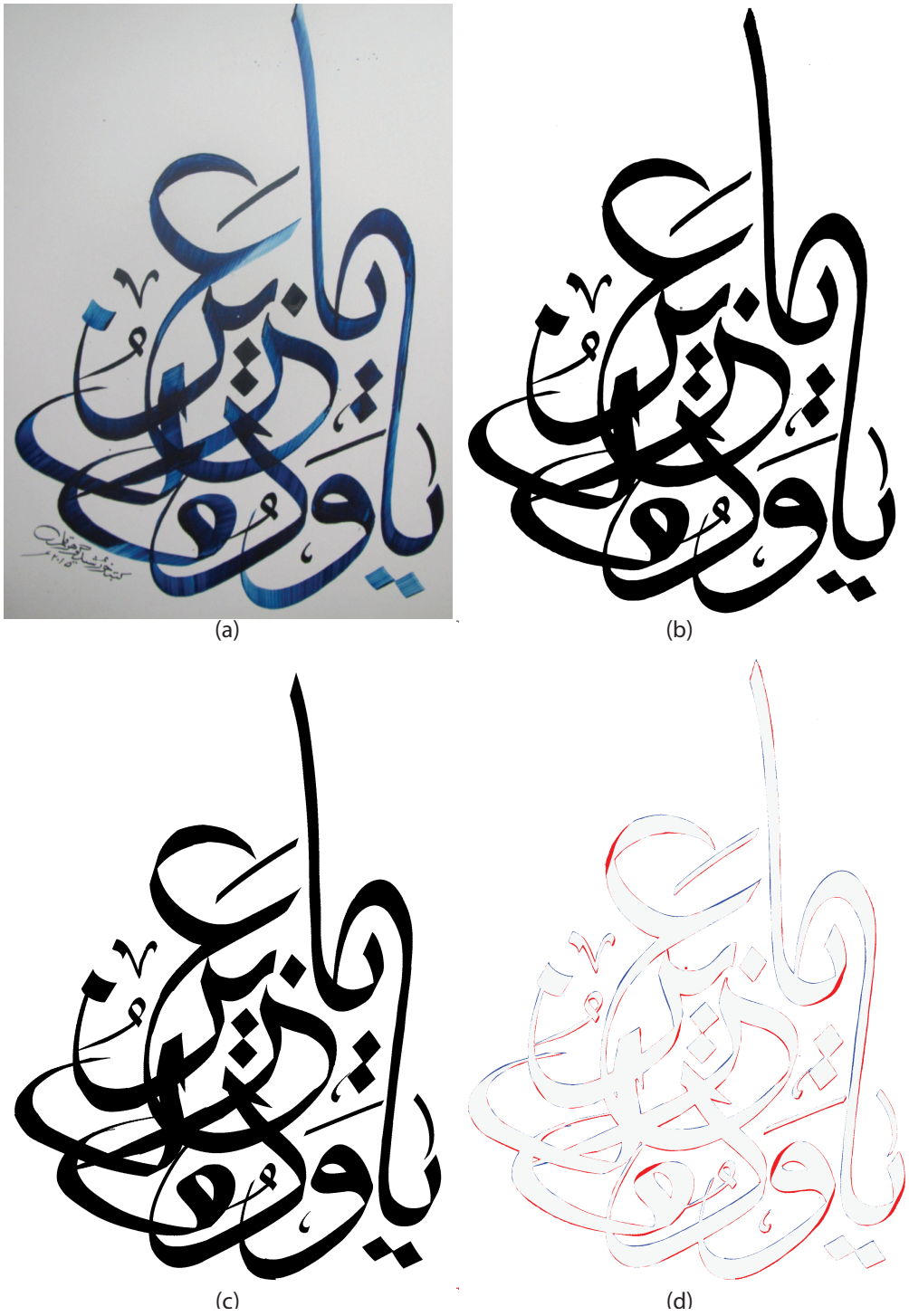


FIG. 10: Thuluth sample by Gohar Qalam. (a) Original calligraphy photo. (b) Original photo processed for analysis. (c) Traced rotating bezier spline ink. (d) Difference between (b) and (c). The red pixel indicate the portions that are missing in (c) but are present in (b) and the blue ones show the missing pixels in (b) but are present in (c).

FIG. 11: Machined Thuluth sample by Gohar Qalam. (a) Rasterized rotating bezier spline for machining (b) Ink marks machined by a simulated robotic manipulator. (c) and (d) are differences between simulated ink mark and the rasterized photo and the processed original photo respectively. The red pixel indicate the portions that are missing in (c) but are present in the reference image and the blue ones show the missing pixels in reference but are present in the ink mark.