

Abstract

[We write it at the end]

Contents

I. Introduction	4
II. Rotating/Twisting Bezier Splines	6
A. Introduction	6
B. The Necessity	6
C. Mathematical Model of a Rotating Bezier Spline	7
1. The Conventional Bezier Spline	7
2. Twist/Rotation Handle	9
D. Conversion of Existing Calligraphy Artwork	10
E. Machine Data Generation	11
F. Characterization	11
G. Supported Scripts	12
1. Coverage	13
2. Sample Results	13
3. Performance with Different Scripts	13
III. Robotic Manipulator	13
A. Functional Requirements of the Manipulator	13
B. Robot Representation	14
C. Selecting a Configuration	18
D. Mathematical Modelling	18
E. Simulation and Analysis Solutions	19
F. Inverse and Forward Kinematics	19
G. The Calligraphy Job	19
H. Forward Kinematics	21
IV. Gregor – The Twisting Bezier Spline Editor	21
A. Introduction	21
B. Requirements and Features	21
C. Usage	21
D. Interface	21

E. Drawing and Modifying a Spline	21
F. Save, Load, Export	21
G. Short-keys	21
H. Other useful features	21
I. Analysis Tools	21
J. Development	21
1. Code Organization	21
2. The BezierBoard Class	21
3. RBSPoint	21
4. Spline Elements	21
5. Miscellaneous Helper classes and form	21
6. Areas needing Improvement	21
V. Drogon The Robot Simulator	21
A. Introduction	21
B. Simulation	22
C. About the Tools	23
D. Workspace Optimization	23
E. Script Path Planner	24
F. Script Motion Simulator	24
G. Requirements and Features	26
H. 3D Visualizer	26
I. Workspace Optimizer	26
J. PID Tuner	26
K. Rotating Spline Importer	26
L. Usage	26
M. Interface	26
1. Manual Control	26
2. Analysis Tools	26
N. Development	26
1. Code Organization	26
2. The SphericalRobot Class	26

O. Simulation	26
1. Assumptions and Limitations	26
2. Accuracy of simulated Quantities	26
3. Forward Kinematics	26
4. Rotating Bezier Spline Curves	26
P. Sample Results	26
1. Accuracy	26
Q. Areas that Need Improvement	26
VI. The Project	26
A. Set Milestones	26
B. Expected Outcomes	26
C. Missed milestones	26
VII. Conclusion	26
References	26
VIII. Code Snippets	28
IX. Tables	29

I. 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 languages⁴ and sharing uncommon biographies⁵⁻⁸. 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 tasks⁹⁻¹² and they can surely uplift this art as well. At the very least, they can be employed in restoration and replication of existing calligraphy work¹³. 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 movement³. 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.

II. ROTATING/TWISTING BEZIER SPLINES

A. 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 artist. 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.

B. 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.

C. Mathematical Model of a Rotating Bezier Spline

1. The Conventional Bezier Spline

In order to describe how twisting splines work, let's 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 angle 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 $\overline{AH_2}$ 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 polygon. If P_{spline} is point on the section 4, with coordinates x and y in a cartesian plane with some origin, it is defined by the following equation.

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

where,

$$P_a = P_2 3f + P_1 2(1 - f) \quad (2)$$

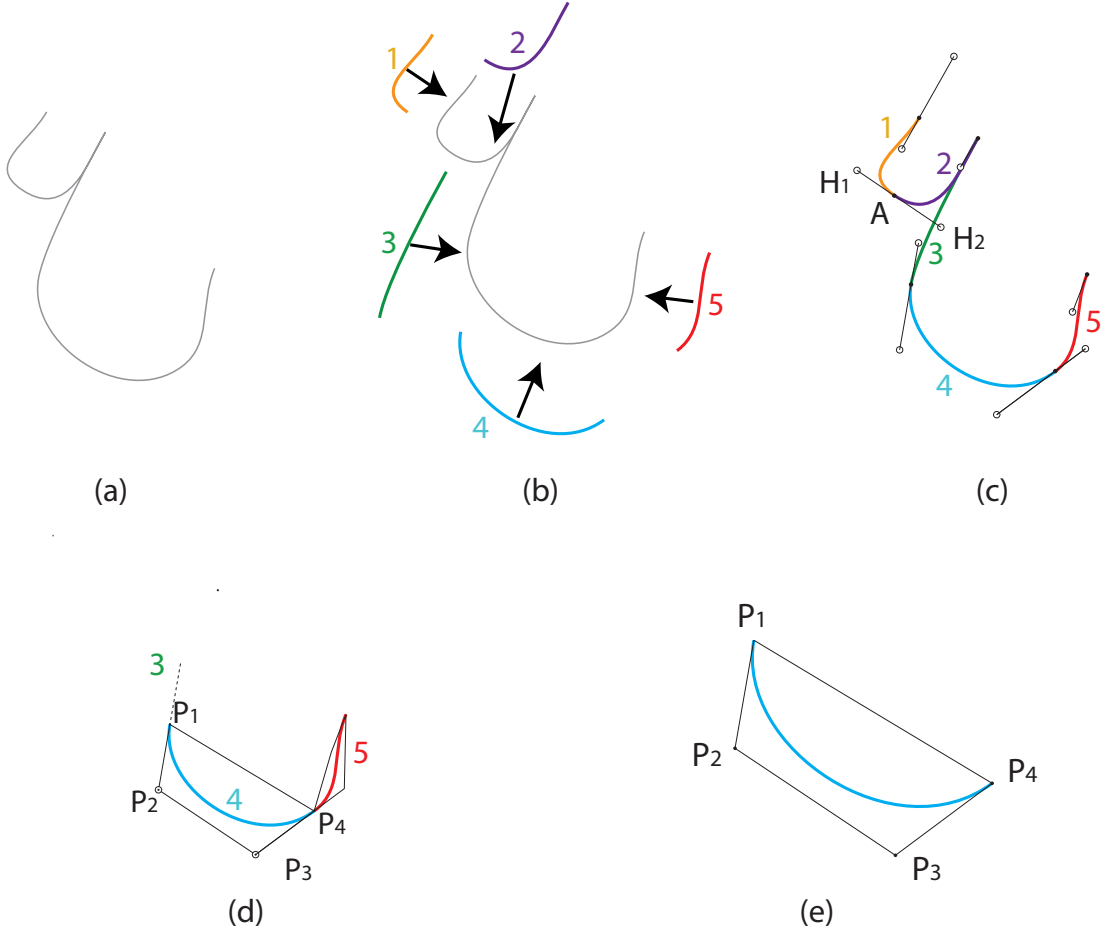


FIG. 1: An illustration showing the construction of Bezier Spline Curve. (a) A sample of a Bezier spline path (b) an exploded view of inner curves of the Bezier spline path (c) Handles that control the shape of the two adjacent sub curves (d) and (e) Construction polygon of the sub curve.

and

$$P_b = P_3 4f + P_2 3(1 - f) \quad (3)$$

where,

$$P_1 2 = P_2 f + P_1(1 - f) \quad P_2 3 = P_3 f + P_2(1 - f) \quad (4)$$

and

$$P_3 4 = P_4 f + P_3(1 - f) \quad (5)$$

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

It can also be proved that the side segments of the polygon $\overline{P_1 P_2}$ and $\overline{P_3 P_4}$ 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 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.

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

$$\begin{aligned} &\theta_{twist} \\ = & \theta_A(1 - f) + \theta_B \end{aligned} \tag{6}$$

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 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 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.

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 popular 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

- easy to use interface
- converting the existing photos to digital form and,
- generating machine data that 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. It will be discussed in detail in later chapters

E. 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, to call the rotation Bezier splines the machine data, the following assumptions must be made:

- The flat tip of the pen 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 the is set by the machine.

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

F. 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.

G. Supported Scripts

The mainline scripts are not necessarily as mathematical as the model of the rotating Bezier splines. Especially, if 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 it would be an artist who will be creating and tracing scripts on the screen of a computer, it was 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 with 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, the still assume the tool to 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 the missing information. This has been discussed in later chapters when we suggest some other improvements in the overall project.

With these assumptions in mind, the “Gregor” was coded.

1. Coverage

Metrics	
Percentage of area outside the original bounds	
Percentage of area covered	
Maximum lateral deviation of the Bezier path from the pitch line	N.A. (This list was planned in th
Total number of compatible scripts	

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 metrics 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 can it be
Easy of usage	A survey based o
Time efficiency of tracing an existing specimen.	Comparison of the time taken by the same artists tra
The artistic quality of the specimens produces.	A survey based on Likert scale and

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 of the National College of Arts (NCA) cite ??? were borrowed; one in Nastaleeq and other in Thuluth.

3. Performance with Different Scripts

III. ROBOTIC MANIPULATOR

A. Functional Requirements of the Manipulator

The robot is required to perform precise maneuvers with complete 6 degrees of freedom of the end effector in both planar and cured planes. It is also required to provide at least one extra actuator to reduce singularities. In addition to position, the robot is also required to perform precise velocity and force controlled maneuvers.

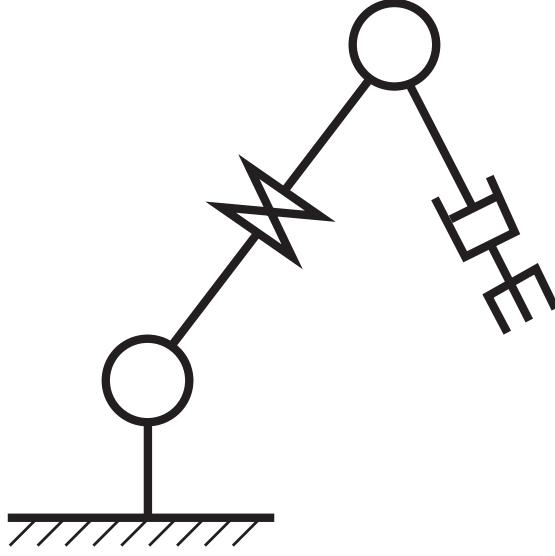


FIG. 2: Conventional way of representing a manipulator. The circles show a rotation between the interconnected links on the plane of the paper. The cross symbol shows rotation on the link line. The square symbol is for prismatic joint. It represents change in length along the link line.

B. Robot Representation

Before going on to the selection of a robot, one unexpected issue arose when I wanted to represent my robot. Conventionally, the robot can be represented on a piece of paper using simple symbols and links as shown in Fig. 2. This kind of representation is clean and simple but doesn't give a complete view of the robot. Specially, with virtually zero link lengths, the visual representation can confuse someone new to the realm of robotic.

Based on repeated experiments and to answer the weakness of conventional representation, let us define a new representation that encapsulates the basic idea of representation with some features of $D-H$ table. See Fig. 3

Interestingly, this representation can't only represent physical manipulators, it can also represent abstract transformations. See Fig. 4 which represents a target P_t being represented as $z-x-z$ euler angles.

Now, if one represents both target and the manipulator representation in a solved state, a closed loop representation can be formed which becomes extremely convenient in inverse kinematics. See Fig. 5 which exemplifies a closed loop **transformation frames** repre-

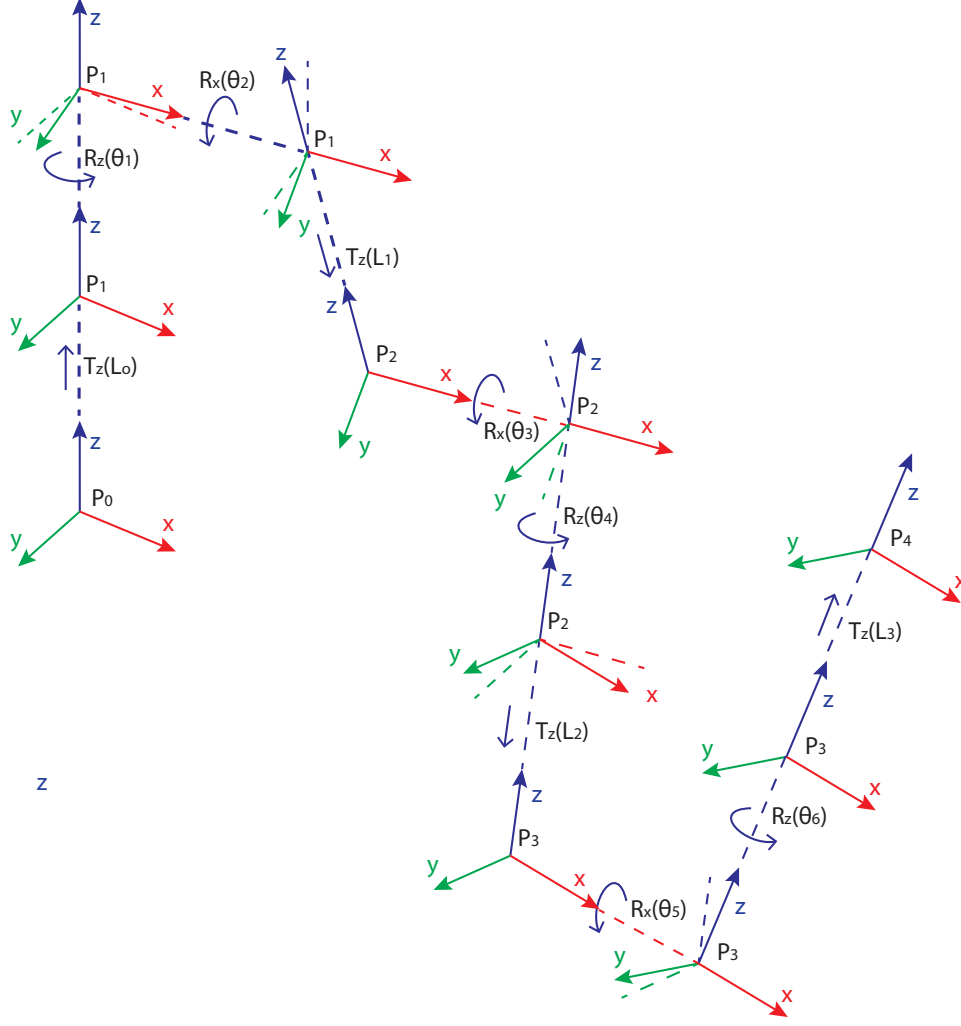


FIG. 3: An example of **transformation frames** representation. This representation shows how a reference frame is linked to the previous using two basic transformation functions, T_n and R_n where n is the axis along or about which the transformation is performed. The schematic is drawn in the direction of F_0 to F_N (N being the last frame) but can also be interpreted in reverse direction. One can easily see which transformations are required to convert one frame of reference to the other. One only needs to take one thing in account. When going in forward direction, all the transformations are applied as they are but while going back, each successive transformation is taken as inverse. For example, frame F_5 can be achieved from frame F_3 by applying two successive transformations: translation along x of magnitude L_1 and rotation about x of magnitude θ_3 . However, to go to frame F_3 from frame F_5 , one needs to apply two successive transformations: rotation about x of magnitude $-\theta_3$ and translation along x of magnitude $-L_1$. It should be noted that if a transformation changes the origin of the frame, the next frame has a different point shown at the center. This point is defined in the base frame.

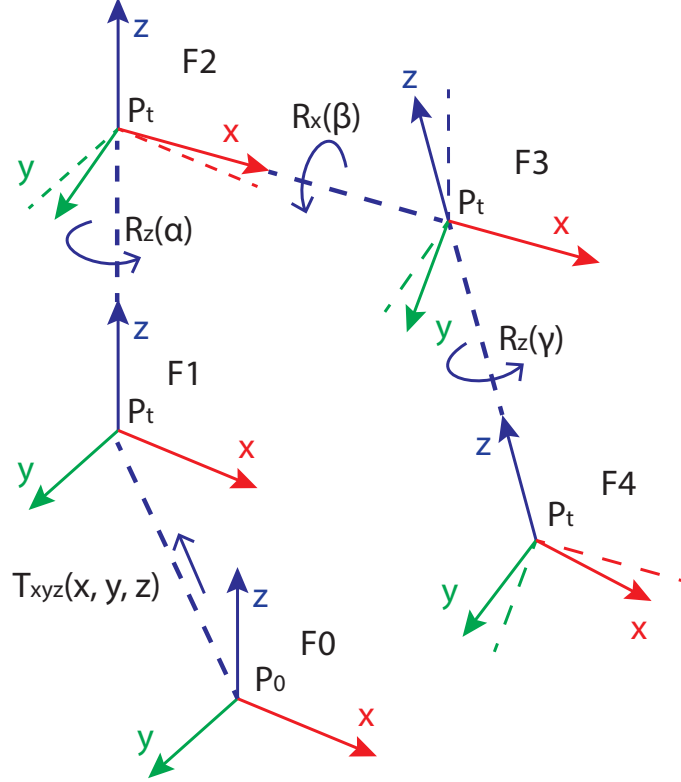


FIG. 4: Three successive rotation $z - x - z$ and similar orders are call **Euler Angles Transformations**. A base Frame $F0$ is transformed though three translations and three euler rotations. This way, one can easily represent a target point to the robot which would have three extra components than x, y and z ; α, β and γ .

sensation. It should be quite clear that once a closed loop is constructed, one can make transformations in any direction.

It is interesting to see that while doing a complete cycle transformation, one virtually transforms a frame through nothing! Also, if a $N - 1$ frame transformation is applied in a N frame closed loop, one can identify one last transformation quite easily by comparing the initial and the final frames. This method is called **Known Error Propagation** and will be used extensively in the inverse kinematics.

Last, but not the least, of the usages of a closed loop representation is that some transformations can be skipped/swapped safely in order to simplify the loop. This can lead to finding further more unknown parameters using **Known Error Propagation**.

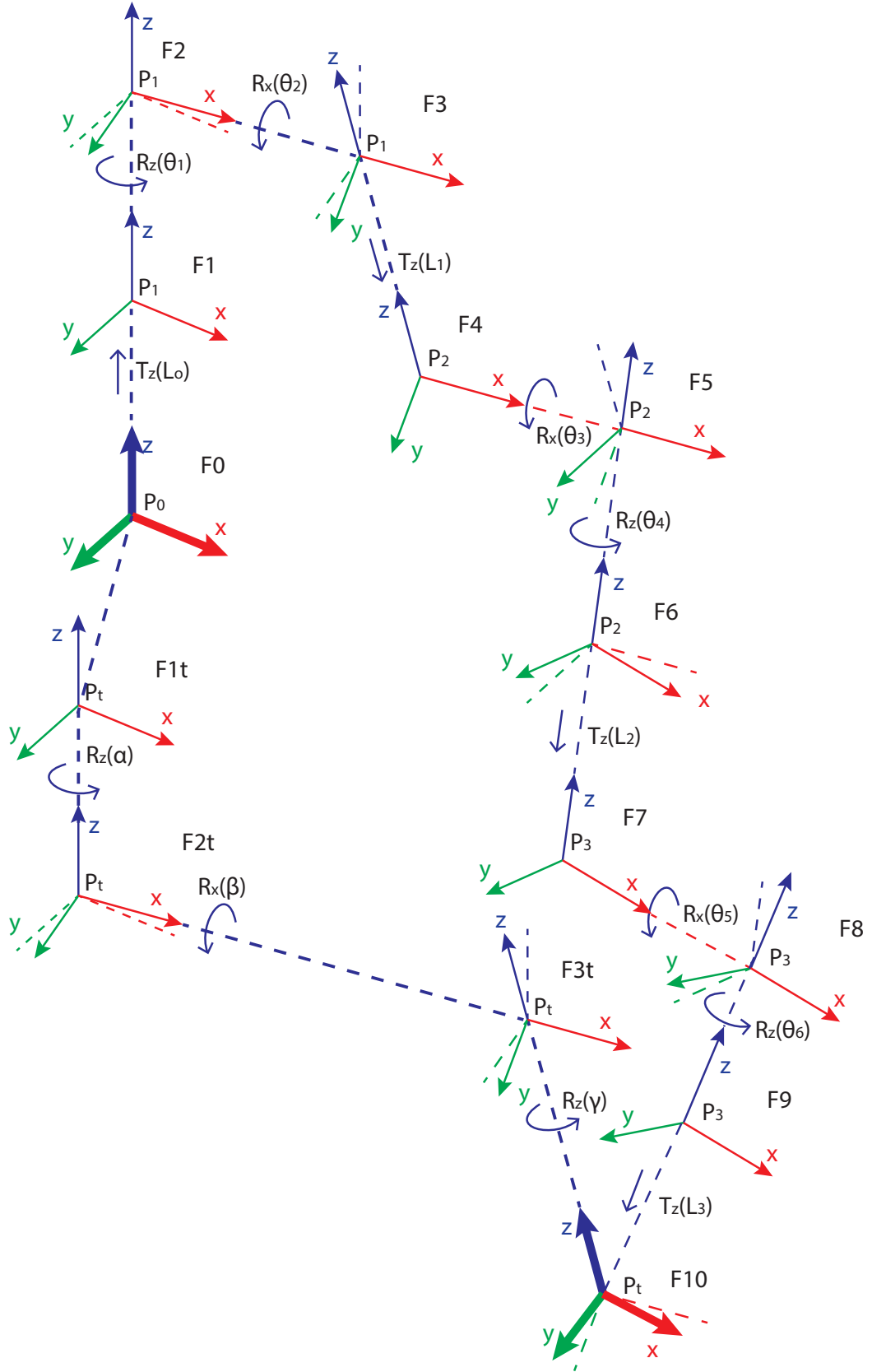


FIG. 5: A manipulator along with a target represented in **transformation frames** representation. F_0 corresponds to the base frame, F_0, F_1, F_2, \dots represent the forward direction of the manipulator and $F_{t0}, F_{t1}, F_{t2}, \dots$ represent the forward direction of the target. Both chains of transformations lead to a common reference frame eventually, the target frame of reference, or the end-effector frame of reference, F_t .

C. Selecting a Configuration

The configuration selected for this project is widely used by other engineers but the main inspiration behind choosing this configuration was a B.Sc. final year project of the *Mechanical Engineering Department of U.E.T* of the batch 12' with title, 'Design and Implementation of a 6DoF Spot Welding Robot'. The project report presents a solution and describes all the forward and inverse kinematics. And all this, without using a regular transformation matrix! The effort in the current project takes some inspiration from the previous one and also elaborates not only the faults and issues with the previous but also the effectiveness of using transformation matrices.

See Fig. 5 which represents this manipulator.

In addition to these six degrees of freedom, we have attached a linear force control actuator along the end effector, since this actuator is responsible to keep the normal force under control, effectively, the robot solution remains the same as for six degrees. This scheme will be described later.

D. Mathematical Modelling

Using the new representation, properties of homogenous transformation matrices and basic highschool trigonometry, I was able to calculate the unique and redundant solutions for the spherical manipulator. The process will be described later, but, summarily, the robot has two primary solutions: different joint positions yield the same end effector position and orientation. Connected to both solutions, are 4 more secondary solutions which effect only the motor angles not the link locations.

Finding out the mathematical equations is one job, verification is another. While dealing with three dimensional realm on a two dimensional paper, one can the solution but making small human errors on the way. It is quite easy to confuse the directions of rotations in a reference frames, resulting in typical errors which can be mitigated by some hit-and-trial of a few operations; adding or subtracting integral multiples of $\frac{\pi}{2}$ from the angles, inverting the length signs, inverting the angle directions. This can be done easily only if a representation tool is build alongside the robot modeling.

E. Simulation and Analysis Solutions

Even with a complete mathematical model, there are tasks which require the power of simulation either for design and planning or for the verification of the existing model. The tool used for this step is discussed in detail in the coming sections.

F. Inverse and Forward Kinematics

Some of the motor angles are quite obvious. See the series of hand-sketched figures labelled with mask Fig. RDn in the subsequent pages which demonstrate how an inverse kinematic solution was discovered for this robot.

G. The Calligraphy Job

Calligraphy is kind of art that finds its links in the, so called, technical areas. The write and discover new fonts on the way but one thing remains the same. Whatever the idea an artist has in his mind, has finally to be represented as Ink marks which, usually, are made flat head pens and rectangular paint brushes. The tool doesn't know what actually drives it, it only knows how it should react to the specific surface and specific kind of input.

This is where we innovate: replace the hand of the artist with the robot. The mind that imagins/creates/discovers the art, however, cannot be replaced by the robot. Or even if it can be, it would be out of scope of this project. So, now we need to create two more interfaces: between the human and the robot and second, between the robot and the writing tool.

For the first interface, we discovered a modified form of bezier curve, "the rotation supported bezier". Bezier curves were first reported in use by a French physicist, Paul de Casteljau using de Casteljau's algorithm. He used bezier splines to model the bodies of Renault cars in 1959. We, after all those years, have discovered how asymmetric pen rotation can be incorporated in the bezier curves to define language scripts that require an asymmetric pen tip.

See Fig. 6 which shows a screenshot of a bezier rotation curve constructed in Drogon v2.

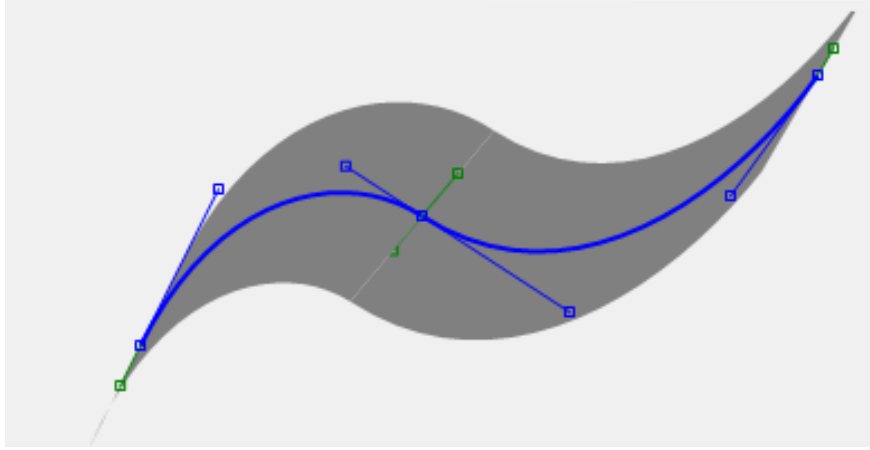


FIG. 6: A usual bezier spline has only two interlinked handles on a each anchor point which can control the shape of the curve before and after the anchor (shown blue). We have added a second (green) handle to each anchor which controls the rotation of the pen. Since the rotation handles act similar to the shapre handles, one can, in-effect, control change in rotation very accurately and smoothly.

H. Forward Kinematics

IV. GREGOR – THE TWISTING BEZIER SPLINE EDITOR

A. Introduction

B. Requirements and Features

C. Usage

D. Interface

E. Drawing and Modifying a Spline

F. Save, Load, Export

G. Short-keys

H. Other useful features

I. Analysis Tools

J. Development

1. *Code Organization*

2. *The BezierBoard Class*

3. *RBSPoint*

4. *Spline Elements*

5. *Miscellaneous Helper classes and form*

6. *Areas needing Improvement*

V. DROGON THE ROBOT SIMULATOR

A. Introduction

21

The name **Drogon** was inspired from an invincible and all rounder character from a famous novel by George. R. R. Martin cite???

With the availability of a highly customizable modeling solution such as **Drogons**, I was tempted to modify it to be used with a robotic arm. This might seem like an over-estimation of the project requirements but, an often undervalued yet important step of project management is the optimum utilization of the available skill-set.

Comprehensiveness is the core attribute of this simulator. It provides, under one screen, the flexibility to introduce an elementary change in the design, like, the maximum speed of an actuator and directly observe the consequences on all of the analysis tools. One can choose to programmatically or manually signal an actuator to move in a particular direction and position, and visualize the outcome in a 3D preview window.

Besides comprehensiveness, our design tool offers a set of powerful analysis tools which enabled us to solve complex robot maneuvers and optimize the solution. Performing mechanical simulation with simplified real world constraints, 3D live preview of the moving robot, work-space optimizer, 2D art drawing, investigating the actuator velocity in continuous and discrete domain are some of the tools we used for optimization in our design.

The tool can easily integrate with Microcontrollers, Proteus and SolidEdge to give more design flexibility.

B. Simulation

The simulator section of **Drogon v2** considers the robot fixed on a horizontal surface at a point $(0, 0, 0)$ and allows the user to signal each actuator either manually or programmatically. These movements are then simulated in tiny timer intervals as linear variables (just like the other simulators). It should be noted that the simulator simulates only the kinematics of the manipulator yet. However, for the sake of simplicity, all the other parameters are not considered. For example, simplest of the motors is the DC stepper motor which is simulated in the current version. Still, one could modify the simulation thread routine to implement a PID controller as well. An example of which can be seen in the simulation of the 7th force control actuator of the robot. Obviously, a price would have to be paid, namely; excessive computational load, software design complexity.

More details on the working principles of the simulator will be discussed in later chapters. Also, the installation and usage of the tool is described in detail in Appendix B.

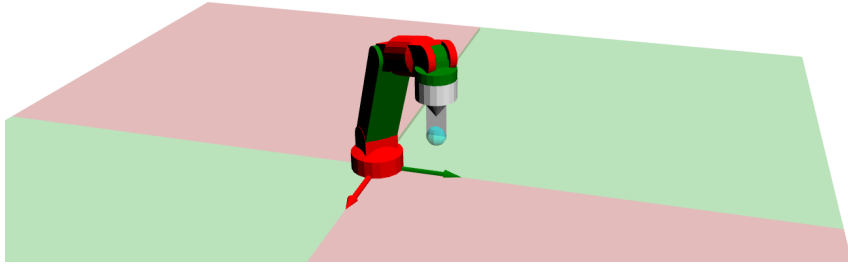


FIG. 7: A screenshot of the 3D representation tool. The blue sphere under the end effector shows the target given to the robot.

C. About the Tools

An shape independent model of the robot under simulation can be visualized in real time using “3D Animation” tool as seen in Fig. 7. The “Robot Feet Position” tool gives a superior insight on the stability of the robot. It can be seen in Fig. 1A(e) that the center of gravity plot can help optimize the motion where stability is a concern. Using the trajectory of the center of the robot in the global frame of reference can help determine the most efficient actuator movements which can be used to displace the robot from a specific position. Screen shots of the plot produced using the “Top Trajectory” tool can be seen in Fig. 1A(d).

All of the tools can output in real time. The robot configuration and other parameters change with time, producing animations which make it easier to decipher the underlying information.

D. Workspace Optimization

The workspace optimizer allows to simulate the robot solution recursively through discrete sections of the defined workspace and evaluates the degrees of freedom the robot has specific parts of the workspace. It then colors the segments to give a more clear idea of the robot workspace. The user can then modify the robot and see, in-result, the change in the workspace. A typical output of the workspace is show in Fig.

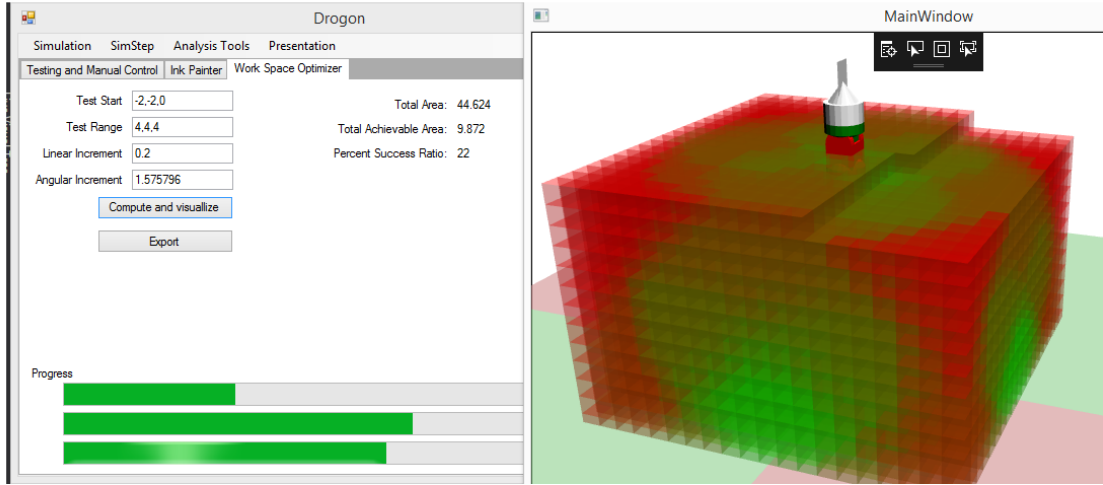


FIG. 8: The workspace is coded in colors. Purely green blocks represent a point where the robot has all of the degrees of freedom and the slightly red blocks indicate the parts where robot starts to loose one or more degrees of freedom.

E. Script Path Planner

As already mentioned, I've included a tool to construct bezier rotation splines in the tool which can serialize the data in computer files and also load from existing files. A screenshot can be seen in Fig.

F. Script Motion Simulator

Once the script is constructed using the script maker, it can be transported to the workspace in any orientation. The robot then plans how to make the required maneuvers and simulates the behaviour. A screenshot of a robot writing a script can be seen in figure 10

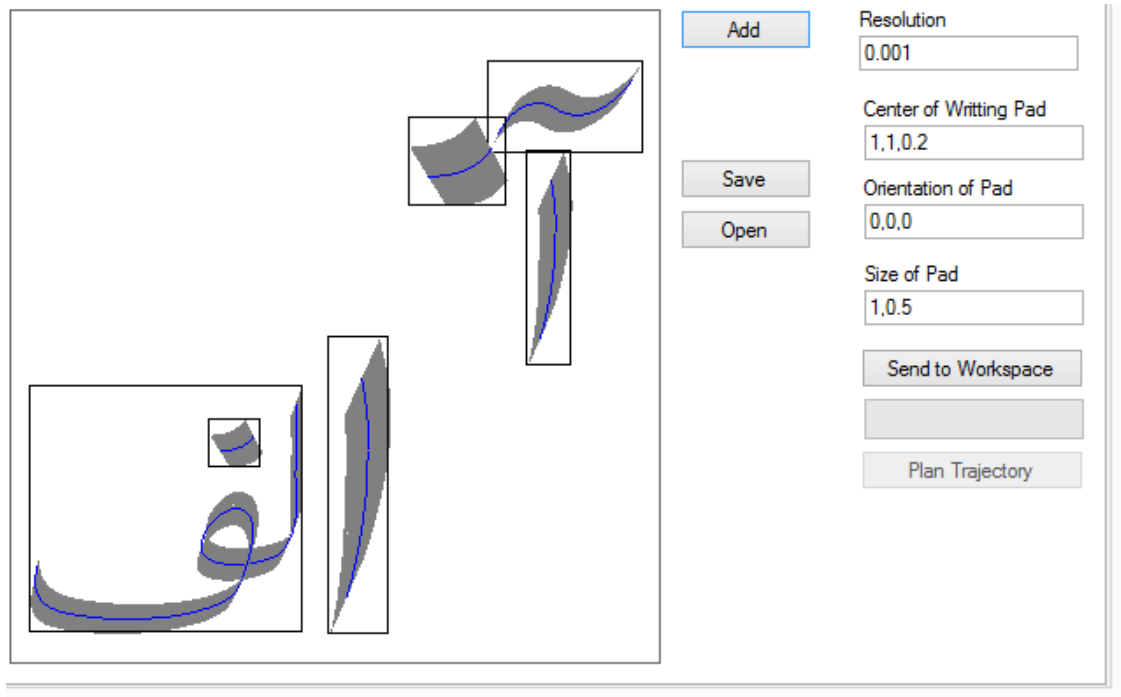


FIG. 9: The script maker can manage multiple splines and render robot trajectory according to the user resolution requirement.

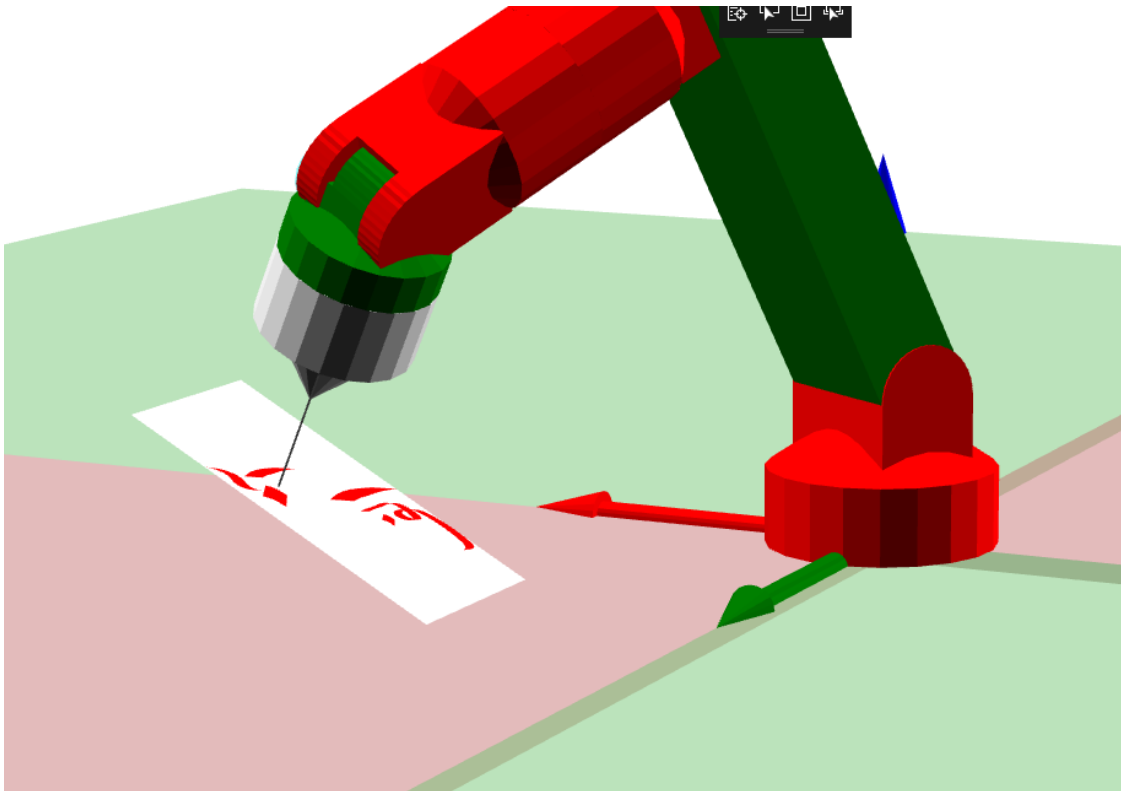


FIG. 10: Robot writing on a slanted plane.

G. Requirements and Features

H. 3D Visualizer

I. Workspace Optimizer

J. PID Tuner

K. Rotating Spline Importer

L. Usage

M. Interface

1. Manual Control

2. Analysis Tools

N. Development

1. Code Organization

2. The SphericalRobot Class

O. Simulation

1. Assumptions and Limitations

2. Accuracy of simulated Quantities

3. Forward Kinematics

4. Rotating Bezier Spline Curves

P. Sample Results

1. Accuracy

Q. Areas that Need Improvement 26

VI. THE PROJECT

A. Set Milestones

- 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.

APPENDIX A: CODE SNIPPETES

```
//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 B: TABLES