

Latex Equation Sequencer using Vector Mask Overlay and Localized Composition Analysis

Shalaka Deore¹, Ranu Mishra², Aishwarya Mapar³, Sakshi Shitole⁴, Pranav Mandani⁵ and Shriniwas Kulkarni⁶

¹⁻⁵Department of Computer Engineering, Modern Education Society's Wadia College of Engineering, Pune, Maharashtra, India

shalakadeore@mescoepune.org

⁶TextMover Labs LLC, Mesa, AZ, USA

skulkarni@fastspring.com

Abstract—LaTeX finds extensive application in report generation across diverse sectors. Predominantly, LaTeX code is incomprehensible without extensive training and compilation into vector or raster images. Complex LaTeX code blocks are somewhat difficult to interpret without code level comments at a glance. To address this, step-by-step animations of LaTeX components offer an interactive experience for readers, facilitating a more intuitive approach to learning different aspects of LaTeX syntax. By dissecting labyrinthine code segments, these animations resemble a debugging process in programming languages, allowing users to grasp each step more effectively. Thus, the proposed system overcomes the disadvantage of the existing system, which is that it can animate latex code.

Index Terms— LaTeX, Animation, Report Generation, Deep Learning.

I. INTRODUCTION

The COVID pandemic witnessed a sudden upward surge in the on-line education system throwing light on the pros as well as the cons of this system. Education became accessible to remote areas and online classes were the new norm. However, for students in high school, mathematics became a challenging subject. Most mathematics and calculus require extensive derivations of formula which are vital to understand further mathematical process. For teaching faculties, it became arduous and cumbersome, to derive formula for each class virtually. It was error prone and repetitive in nature. Henceforth we came up with the idea of converting complex LaTeX equations to a lucid and unambiguous form. The process involves three modules where we first convert the input LaTeX equation to SVG image format, then use Convolutional Neural Network to identify and classify handwritten characters, followed by animating the equations using animation techniques. Pronounced "lay-tech," LaTeX is a writing language that, like HTML/CSS or Markdown, uses markup elements to decorate plain text. Academics frequently utilize the LaTeX system to create papers similar to academic publications. Unlike word processors like Microsoft Word, LibreOffice Writer, or Apple Pages, which only stylize the content, LaTeX writers write code in plain language that needs to be compiled to produce a PDF page. When authoring a complicated document such as a book, LaTeX is a more advantageous tool than Word. If you need to swiftly produce several pages, Word is a superior option. Longer and more complex books take less work when using LaTeX. In this project we are using Docker to convert LaTeX equation to SVG and will be animating it using JavaScript libraries. An extension of this project is to identify handwritten characters so as to pave path for future

work where in, handwritten mathematical equations can also be identified and animated. Primary scope of this project is academic presentations and e-learning coursework material can use the output of this project tools. This system proves highly beneficial for students who regularly engage with LaTeX code in their daily work.

II. LITERATURE REVIEW

A. Handwritten Character Generation

In paper [19] Alex Graves presents a method for generating diverse handwriting styles using LSTM based RNNs. The author proposes a decoupled approach, where the style and content components of handwriting are learned separately. By conditioning the RNN on style variables, the model can generate characters in different styles while maintaining consistent content. The method is shown to generate visually appealing and natural-looking handwritten characters. Amit Choudhary et. al. [16] proposed using a binarization technique to convert the gray scale handwritten character images into binary images, where pixels are either black or white. Binarization is a common preprocessing step in character recognition tasks that simplifies the subsequent feature extraction and classification processes. T. Xu and X. Xiao [15] introduces the concept of generating sequences using Recurrent Neural Networks (RNNs). The author demonstrates the efficacy of RNNs, specifically Long Short-Term Memory (LSTM) networks, in generating realistic and coherent sequences, including handwritten characters. The paper highlights the ability of LSTM networks to capture long-term dependencies and generate sequences of variable lengths. The author of article [1] describes a technique for character template estimation that is unique to each document. Improving character recognition accuracy in the context of optical character recognition (OCR) systems is probably the paper's main objective. In order to address this issue, the article introduces the idea of "document-specific character templates." The authors recommend customizing character templates for each unique document, as opposed to utilizing universal templates for character recognition. The study conducted by Breuel et al. [3] centers on the transfer of data from paper documents to Personal Digital Assistants (PDAs) via the integration of mobile technologies and document image processing. The paper's primary goal is probably to suggest and illustrate an automated process for moving text and graphics from printed documents to portable electronic devices. T. Wang attempted something similar [17]. The problem of text recognition in photos—where the objective is to automatically translate text material present in photographs into machine-readable text—was probably addressed by the authors. The principal neural network architecture for text recognition that the authors suggested employing is CNNs. Because CNNs are deep learning models that can automatically extract hierarchical characteristics from input photos, they are frequently utilized for tasks involving images. Their goal was to create a text recognition system that could process images from input to output and recognize text without the need for further text detection or segmentation procedures. This is known as an end-to-end solution. The similar method was employed in paper [18] as well for the Convolutional Neural Network-based Urdu Handwritten Character Recognition. It is technique for clustering offline handwritten mathematical statements using Convolutional Neural Network (CNN) based spatial classification features is presented by C.T. Nguyen et al. [20]. The authors suggest improving the clustering of handwritten mathematical phrases by employing CNN-based features. The method seeks to increase the efficiency and accuracy of combining related mathematical statements. The study advances the field of pattern recognition and may find use in the analysis and digitization of handwritten mathematical text. In order to attain greater performance on computer vision problems, C. Szegedy and W. Liu [22] addressed the need for enhanced CNN architectures. Tasks like object recognition, image categorization, and image segmentation may fall under this category. A comparable technique for handwritten bangle characters using CNNs was proposed by S. Sen et al. [23], In order to address the online handwritten Bangla character identification task and suggested a solution for handwritten bangle character utilizing CNNs, which are deep learning models specifically suited for processing visual input. CNNs are good at identifying patterns and spatial characteristics in pictures. Vectorization in the context of graphics recognition is the main topic of K. Tombre et al. paper [9]. The primary aim of this study is probably to suggest methods and approaches for vectorizing graphical shapes that are taken from photographs in a consistent and reliable manner. The challenge of compressing text images—images with textual content—is introduced by P. G. Howard [13]. Traditional image compression methods are meant to work with generic images, however word images frequently have unique properties that could be used for more efficient compression.

B. SVG

The [2][4] article describes a broad procedure for automatically digitizing documents so they can be processed and stored on various digital platforms. The emphasis is on documents that offer regular structures in the form of text-like blocks, even though optical character recognition (OCR) systems are unable to read them. The use of

Scalable Vector Graphics (SVG) for graphical representations and Extensible Markup Language (XML)[6] for structural descriptions enables seamless integration and system-specific output.

Michael Plass and Maureen Stone [21] introduces a method for curve fitting using piecewise parametric cubics. Presented at SIGGRAPH '83, the authors propose a technique to approximate complex curves using cubic segments. This approach allows for smoother curve representations and efficient computational methods. For XML-based Processing, glyphs are vectorized and expressed in SVG [5,6]. The technique for converting text glyph bitmap images into a vector format that may be added to XML representations of digital documents is presented in this study. The primary focus is on the contour-based vectorization technique because it produces SVG glyph descriptions that are easy to translate. A specific implementation example is given, and the outcomes are scrutinised, paying particular attention to the visual element. The project entails developing a method for managing papers that are unreadable by existing OCR methods. A. Kotani et. al. [8] presents a novel approach to generating handwriting that involves breaking down the process into two main components: content and style. This likely refers to a method for separating the content of the text from the stylistic aspects of the handwriting, allowing for more flexible generation of handwriting with different styles. "Apache Batik Rasterizer"[14] provides a tool for converting SVG (Scalable Vector Graphics) images to raster formats. It's maintained by the Apache Software Foundation. The tool's purpose is to render SVG graphics into bitmap images like PNG or JPEG. Paper [11] focuses on a method for approximating complex curves with simpler polygonal representations, with an emphasis on achieving high-quality results using a relaxation-based approach. Here's a review and overview of the paper.

After studying and going through above literature review, the new features we will be adding are listed below:

Animation: Limitation of current existing system is animation. We can't animate latex code using current system which is available on google. So, we provide a platform where user can easily animate latex report.

Lucid learning: This system will also be a learning platform where user can provide documents from which he/she wants to learn. System will present that document in effective format for easy learning.

Enhanced user experience: Adding new features of animations enriches user experience and engages them more in academic activities.

III. METHODOLOGY

In this work we proposed machine learning to convert latex equation to SVG and will be animating it. This SVG is the intermediary input that is processed by the AI/ML algorithms. We will be using any of the Machine learning techniques (supervised mostly) - where we train the AI to take SVG element. We train the model so that they take the SVG components one by one – Identify what each component is, then identify the sequence it needs to be drawn in on the board. A well-known solution to curve fitting is the algorithm for automatically fitting digital curves that is described in the Graphics Gems chapter. Please note that this does not guarantee the endpoint tangent angles; it only accepts scattered points as input. It depends on improving the solution iteratively, as we'll go into much more depth later, and as a result, it can get stuck in one of the local minima. Potrace is another well-known tool that makes use of B-ezier curve fitting; it converts bitmap pictures into vector shapes. Still, Potrace opts for a simpler, faster approach that yields only a subset of all possible B-ezier curves rather than aiming for very exact fitting. Our goal is to convert latex code to SVG and will be animating output with the help of machine learning. In this project we are using machine learning to convert latex equation to SVG and will be animating it. This system is very useful for students who daily work with latex code.

A. Latex Input to SVG

Our system is divided into 3 modules as shown in Figure 1. The very first module is conversion of latex equation or code to SVG image. At this stage, user will give input to the system and it will be converted to SVG image format for further process. Our frontend system where user will provide input is built using HTML, CSS and JavaScript. Once you have the image, proceed with the following method:

1. Prepare the LaTeX Equation: Create a LaTeX file that includes the equation you wish to convert to an SVG image. You can name the file as desired and ensure it contains the correct LaTeX equation code.
2. Set up the Docker Container: Pull the Docker image that contains the LaTeX-to-SVG conversion library. This image should be specifically designed to convert LaTeX equations to SVG format. Make sure you have the necessary permissions to access the image.

3. Convert LaTeX to SVG: Place the LaTeX file (created in Step 1) in a directory that is accessible from within the Docker container. This can be achieved by mounting a volume or sharing the directory with the container.
4. Run the Docker Container: Execute the appropriate Docker command to start the container, specifying the mounted volume or shared directory where the LaTeX file is located. The command will instruct the container to convert the LaTeX equation to an SVG image.
5. Retrieve the SVG Image: Once the Docker container completes the conversion process, the resulting SVG image will be generated. You can find it in the same directory as the original LaTeX file or in a designated output directory.

By following this method, the LaTeX-to-SVG conversion library within the Docker container handles the conversion process seamlessly. This approach eliminates the need for manual installations or configuring additional software dependencies. It streamlines the conversion of LaTeX equations to SVG images, enabling easy integration into research workflow.

B. Handwritten Character Generation

As long as we have documented and noted down the geometry of that person's writing for recognized letters and symbols, we can use any latex input in this project to create an output in the handwriting of any other human. The most widely used curve format for both design and graphics is the cubic Bezier curve. Curve fitting, or finding the Bézier that is closest to a given source curve, is one of the basic issues when working with curves. Reliability in deriving this representation from a narrowly spaced group of points approximating the intended curve, e.g., input from a digitizing tablet or scanner, would be helpful for many applications. This work tackles the problem of automatically generating piecewise parametric cubic polynomial approximations to shapes that are effective from obtained data. With a set of sample points and optional endpoint and tangent vector constraints, our method repeatedly creates a single parametric cubic polynomial that is close to the data points as assessed by an error measure based on least-squares. When this procedure is used with dynamic programming methods to choose the knot location, good outcomes are obtained for a variety of forms and uses.

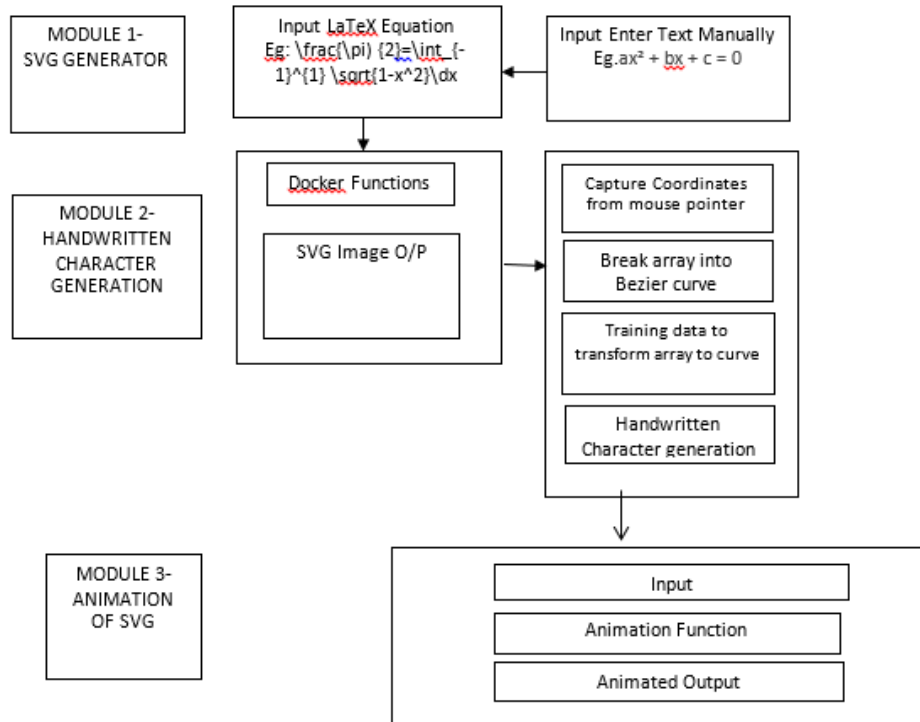


Figure 1: Proposed system architecture

Project uses a combination of open-source programs to produce hand drawn sequence of letter geometry using vector definition similar to Potrace and Graphics Gems chapter.

C. Animation

In this module, we implement a typing animation effect on the content represented by `<path>` elements within an SVG image. The objective is to demonstrate a practical approach to creating a dynamic and engaging typing animation using HTML, CSS, and JavaScript. The typing animation on the elements within the `<g>` group in an SVG image. implementation of a typing animation effect on the content represented by `<path>` elements within an SVG image. By dynamically loading the SVG image and manipulating the opacity attribute of the elements, a smooth and gradual reveal animation is achieved. This implementation example can serve as a starting point for incorporating typing animations into various SVG-based projects, enhancing their visual appeal and user engagement. The approach involves dynamically loading an SVG image using an `<object>` element and performing a typing animation on the elements within the SVG image. The code utilizes JavaScript to access the SVG document, retrieve the desired elements, and manipulate their attributes to create a gradual reveal effect. By using a combination of event handling, iteration, and timed function calls, the code achieves an engaging typing animation on the SVG elements.

IV. RESULTS

In this module, we implement a typing animation effect on the content represented by `<path>` elements within an SVG image. The objective is to demonstrate a practical approach to creating a dynamic and engaging typing animation.

A. Module 1 (Latex to SVG Conversion)

It is the first phase of project development. At this stage, user will give input to the system and it will be converted to SVG image format for further process. Our frontend system where user will provide input is built in HTML, CSS and JS After taking input we will be converting it to SVG using Docker in backend and this will be input for the further stage.

$$\frac{\pi}{2} = \int_{-1}^1 \sqrt{1-x^2} dx \quad \Rightarrow \quad \frac{\pi}{2} = \int_{-1}^1 \sqrt{1-x^2} dx$$

Figure 3: Latex code converted to SVG Output

For converting latex to svg we have used docker image, i.e., latex-docker which is a container that helps compiling latex sources as shown in Figure 3.

B. Module 2 (Handwritten Character Generation)

We perform Exploratory Data Analysis as the primary step to understand features and their dependencies in dataset. We perform data pre-processing to split training and testing data and reshape input into 3D array to be compatible with Convolutional Neural Network. The target labels are converted to one-hot encoded format. One-hot encoding is a technique used to represent categorical data in a format suitable for machine learning algorithms. Finally, the CNN model architecture is defined, compiled with appropriate parameters, and trained on the training data, with the validation data used for monitoring the model's performance during training. The Potrace(TM) tool can be used to trace a bitmap, converting it into a scalable, smooth image. One of several vector file types is used as the output, and bitmaps in the PBM, PGM, PPM, or BMP formats are used as the input.

C. Module 3 (Animation)

In this module, we implement a typing animation effect on the content represented by `<path>` elements within an SVG image. The typing animation on the elements within the `<g>` group in an SVG image. implementation of a typing animation effect on the content represented by `<path>` elements within an SVG image.

Figure 4: Animated Output

The animated output is shown in Figure 4. There is total four direction used like Animate Up, Animate Down, Animate left, Animate right. The complexity analysis for each animation direction is "Low." This is because animating objects or characters in a specific direction typically involves straightforward translation or

displacement of their position on the screen. The movement can be achieved by adjusting the object's coordinates within the animation software or by using simple transformations. Overall, these basic directional animations do not require complex calculations or intricate techniques, making them relatively easier to implement compared to more advanced animation effects.

V. CONCLUSION

The methods presented demonstrate promise for vectorizing glyphs and defining whole alphabets using SVG, which are valuable tools for preparing digitized documents. Despite the compression of shape descriptions, it has been observed that good visual quality can still be achieved. Thus far, we have exclusively employed methods that result in linear primitives for representing object contours. This approach offers advantages such as easy and fast computation, as well as a straightforward transformation into SVG paths. These simple descriptions have proven to be sufficient for the given task. However, exploring more complex primitives, such as Bezier curves, may enhance accuracy and reduce the number of vertices. Particularly in the context of glyph vectorization, it is still debatable whether the somewhat better outcomes warrant the higher computing effort. Potrace, a programme that transforms bitmap images into vector shapes, is another well-known Bézier curve fitting tool. However, Potrace chooses a quicker and easier method that simply produces a portion of all potential Bézier curves rather than attempting to fit the data with exceptional precision. This matter will be investigated further in future research. Primary scope of this project is academic presentations and e-learning coursework material can use the output of this project tools.

Our present work is concentrated on creating a framework that streamlines the retrospective digitization and publication of handwritten works in addition to enhancing the methods currently in use. The outcomes hold great promise for successfully incorporating the discussed vectorization techniques into the previously indicated document image analysis system.

REFERENCES

- [1] KOPEC, G. LOMELIN, M. Document-Specific Character Template Estimation. In Proc. SPIE. Edit. by Luc M. Vincent, Jonathan J. Hull. 1996, vol. 2660. Document Recognition III.
- [2] PLETTSCHACHER, S. OCR Alternatives for Electronic Publishing of Digitised Documents. In Proceedings 9th ICCV International Conference on Electronic Publishing. Leuven: Peters Publishing Leuven, 2005.
- [3] BREUEL, T. JANSSEN, W. POPAT, K. BAIRD, H. Paper to PDA. In ICPR '02: Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02). IEEE Computer Society, Washington, DC, 2002, vol. 1.
- [4] SVG - Scalable Vector Graphics, W3C. <http://www.w3.org/Graphics/SVG/>.
- [5] Stefan Pletschacher Vectorization of Glyphs and Their Representation in SVG for XML based Processing (2006).
- [6] XML - Extensible Markup Language, W3C. <http://www.w3.org/XML/>.
- [7] Apache Software Foundation, Squiggle - The SVG Browser: <http://xml.apache.org/batik/svgviewer.html>.
- [8] Kotani, Atsunobu & Tellex, Stefanie & Tompkin, James. (2020). Generating Handwriting via Decoupled Style Descriptors. 10.1007/978-3-030-58610-2_45.
- [9] TOMBRE, K. AH-SOON, C. DOSCH, P. MASINI, G. TABBONE, A. K. Chhabra, D. Dori, S. Stable and Robust Vectorisation: How to Make the Right Choices. In Graphics Recognition - Recent Advances.
- [10] PENG, JINGLIANG. An efficient algorithm of thinning scanned pencil drawings. Journal of Image and Graphics, 2000, vol. 5, no. 5, p. 434-439.
- [11] MIKHEF, Artem VINCENT, Luc FABER, Vance. High-Quality Polygonal Contour Approximation Based on Relaxation. In Proceedings 6th International Conference on Document Analysis and Recognition (ICDAR 2001). IEEE Computer Society, 2001.
- [12] Unicode Inc., Unicode Standard. <http://www.unicode.org/standard/standard.html>.
- [13] HOWARD, P. G. Text image compression using soft pattern matching. The Computer Journal, 1997, vol. 40, p. 146-156.
- [14] Apache Software Foundation, Batik Rasterizer. <http://xmlgraphics.apache.org/batik/svg rasterizer.html>.
- [15] Liang, T. Xu, X. Xiao, P. A new image classification method based on modified condensed nearest neighbor and convolutional neural networks. Pattern Recognit. Lett. 2017, 94, 105–111.
- [16] Amit Choudhary, Rahul Rishi and Savita Ahlawat, "Off-Line Handwritten Character Recognition using Features Extracted from Binarization Technique", 2212-6716 © 2013 American Applied Science Research Institute doi:10.1016/j.aasri.2013.10.045.
- [17] Wang, T. Wu, D.J. Coates, A. Ng, A.Y. End-to-end text recognition with convolutional neural networks. In Proceedings of the 21st-International Conference on Pattern Recognition, Tsukuba, Japan, 11–15 November 2012; pp. 3304–3308.
- [18] Husnain, M. Missen, M.M.S. Mumtaz, S. Jhanidr, M.Z. Coustaty, M. Luqman, M.M. Ogier, J.M. Choi, G.S. Recognition of urdu handwritten characters using convolutional neural network. Appl. Sci. 2019, 9, 2758.
- [19] Graves, Alex. (2013). Generating Sequences with Recurrent Neural Networks.

- [20] Nguyen, C.T. Khuong, V.T.M. Nguyen, H.T. Nakagawa, M. CNN based spatial classification features for clustering offline handwritten mathematical expressions. *Pattern Recognit. Lett.* 2020, 131, 113–120.
- [21] Michael Plass and Maureen Stone. 1983. Curve-fitting with piecewise parametric cubics. In *Proceedings of the 10th annual conference on Computer graphics and interactive techniques (SIGGRAPH '83)*. Association for Computing Machinery, New York, NY, USA, 229–239. <https://doi.org/10.1145/800059.801153>
- [22] Szegedy, C. Liu, W. Jia, Y. Sermanet, P. Reed, S. Anguelov, D. Erhan, D. Vanhoucke, V. Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
- [23] Sen, S. Shao, D. Paul, S. Sarkar, R. Roy, K. Online handwritten bangla character recognition using CNN: A deep learning approach. In *Advances in Intelligent Systems and Computing*, Springer: Singapore, 2018, Volume 695, pp. 413–420. ISBN 9789811075650.