AUTHENTICATING SIGNATURES BASED ON THE B-SPLINE TECHNIQUE

RINELA KAPÇIU[1], BRIKENA PRENI[2], JULIAN MATA[3]

[1,3]Department of Computer Science, Faculty of Information Technology, Aleksander Moisiu University of Durres, Albania

[2]Department of Mathematics, Faculty of Engineering Mathematics and Engineering Physics, Polytechnic University of Tirana, Albania

e-mail: rinelakapciu@uamd.edu.al

Abstract

This research article presents a novel approach for genuity checks of digital signatures based on B-spline curves, a mathematical representation widely used in computer-aided design (CAD) and graphics. The method transforms the signature into a mathematical representation using knots, control points, and curve interpolation, which is then used to develop a signature verification algorithm. Techniques in signature image processing and optimization methods, such as BGFC, are employed to address these challenges. OpenCV is used for signature image processing and extraction, with morphological mathematical methods for skeletisation. An optimization algorithm like BGFC is used to fit the B-spline with the traced signature curve, aligning the traced curve's initial guess with the signature's shape. The research uses a comprehensive dataset of signatures to validate the effectiveness of the proposed B-spline-based genuity check. Experimental results show the algorithm can accurately distinguish between genuine and forged signatures, even in the presence of distortions and noise. The computational efficiency of the method positions it as a practical solution for real-time signature authentication applications. This work contributes to digital signature verification by enhancing the accuracy and reliability of genuity checks, with potential applications in secure document management, financial transactions, and other digital interactions.

Key words: *B–Spline, AI, BGFC, skeletisation, curve fitting, automation, sign.*

Përmbledhje

Në këtë artikull paraqitet një metodë shumë e saktë për kontrrollinin e nënshkrimit dixhital duke u bazuar në sheshimin me anë të kurvave B-spline, një paraqitje matematikore gjerësist e përdorur në dizenjimet kompjyterike (CAD) dhe grafike. Metoda transformon nënshkrimin në një paraqitje matematikore duke përdorur nyje, pika kontrolli dhe interpolim me anë të kurvave të lakuara, të cilat më pas përdoren për të zhvilluar një algoritëm për verifikimin e nënshkrimit. Teknikat për procesimin e imazhit të nënshkrimit dhe metodat optimizuese, si për shembull BGFC, janë përdorur për adresuar këto sfida. OpenCV është përdorur për procesimin e nënshkrimit si imazh dhe për ekstraktimin e saj, me ndihmën e metodave të matematikës morfologjike për skeletizimin e problemit. Një algoritëm optimizues si BGFC përdoret për të përshtatur kurbën B-spline me kurbën e nënshkrimit të gjurmuar, duke përafruar interpolimin fillestar të kurbës me nënshkrimin origjinal. për këtë punim është përdorur një databazë publike me nënshkrime për të kryer validimin e efektshmërisë së metodës së propozuar, e cila bazohet në interpolimin me anë të B-spline-ve. Rezultatet eksperimentale tregojnë se algoritmi dallon me saktësi nënshkrimet origjinale nga ato të fallcifikara, edhe në prezencën e shtrembërimeve ose zhurmave. Efektshmëra llogaritës e metodës e pozicionon atë ndër zgjidhjet më praktike për verifikimin e nënshkrimit në kohë reale. Ky punim kontribuon në verikifikimin e nënshkrimit dixhital duke rritur saktësinë dhe besueshmërinë në kontrollet e origjinalitetit, me aplikime të mundshme në menaxhimin e sigurt të dokumentave, transaksionet financiare dhe ndërveprime të tjera dixhitale.

Fjalë kyçe: *B–Spline, AI, BGFC, skeletizim, sheshimi me vijë të lakuar, automatizim, nënshkrim.*

Introduction

The B-spline curve is a powerful and versatile approach to curve design, addressing issues with the Bezier curve. It addresses problems with Bezier curves and is widely used in 3D animations to determine the path of motion and interpret critical segments. B-splines are also used to model animation characters, animated movies, and True Type fonts, providing a comprehensive solution to curve design challenges.

B-spline models were first mentioned by (Schoenberg. 1969). Initially, the practical application of B-spline curves and surfaces was limited by the instability of the algorithms for their calculation. This setback was resolved by (de Boor 1972), who proposed an algorithm for numerical computation based on a recursive template. Then, Reisenfeld algorithm reveals that the B-spline is a powerful tool for representing free-form shapes and applied the B-spline base for curve definitions.

B-spline curves are a powerful technique used in modelling complex shapes and surfaces, often used in computer modelling and graphic design environments like AutoCAD and Blender. These curves are divided into small, controllable segments, making them suitable for creating and manipulating various graphical objects and creating moving animations. NURBS, or Non-Uniform Rational B-Spline, is a more common interface in 3D modelling, including B-spline curves and weight settings.

This study aims to use image manipulation techniques, such as signatures, to examine and present signatures as binary images using Computer Vision technology. A dataset of signature images is prepared using a moderate number of signature images from firm authors, and a dataset of forged signatures is created to evaluate the accuracy of the model used. A computer model is trained for signature recognition in the signature photo dataset, and the model is tested using forged signature images to determine its ability to distinguish them from actual signatures.

The main goal of this study is to develop an efficient and accurate tool for verifying the authenticity of signatures using computer technologies and related sciences, focusing on signature authentication and detecting forged signatures in security and fraud prevention contexts.

The order of the remaining paper is as follows: Relevant research in this field is compiled in the following sections. The suggested algorithm is described and the methods utilized are defined.

B-splines in pattern recognition

B-Spline theory has been widely developed and is explained in detail in (de Boor, 1972). This section outlines the necessary elements from the theory to comprehend the application of B-spline in this research.

Using standard computer-aided modeling techniques like curves and B-spline surfaces, geometric modeling creates and portrays free-form curves and surfaces.

The B-splines can describe various forms of curves (Rogers, 2001). Bezier curves generalization are splines with an orthonormal basis of recursive functions, consisting of curves with polynomials at knot points. The degree of these polynomials is identical to that of a B-spline, with cubic segments being an essential part of these curves.

A spline curve is a continuous curve formed by joining a sequence of segments. The B-spline curve addresses the problems with the Bezier curve. Bezier curves are a powerful and versatile approach to curve design, used in 3D animation to define the path of an object's movement. They are also used to model animation characters and fonts of various shapes, with their two-dimensional curves used for rendering key segments and their three-dimensional path defined by B-splines. B-spline models were first mentioned by Schoenberg (Schoenberg, 1969). The initial practical application of B-spline curves and surfaces was restricted due to the instability of their calculation algorithms. This obstacle was solved by Cox and de [de Boor 1972], who proposed an algorithm for numerical calculation based on a recursive model. Then, using the Reisenfeld algorithm is clearly found that the B-spline is a powerful tool for representing free-form shapes and applied the B-spline basis for curve definitions.

The B-spline's flexibility can be enhanced by increasing its order and inserting nodes.

The B-spline curve's flexibility is being enhanced.

If $S_k=[B_0^k(t)\ B_1^k(t)\ \ldots\ B_n^k(t)]^T$ is a B-spline basis defined in vector form, then

$$S_k(P;t)= S_k^T\, P = \sum_{i=0}^{n} B_i^k(t)P_i \quad , \quad t \in [t_{k-1}, t_{n+1})$$

Is the B-spline curve defined by $P=[P_0\ \ldots P_n]^T$.
It is clear that the choice of basis functions and the choice of knot vectors significantly affect the shape of the B-spline curve.

The B-spline curve maintains smoothness with increasing order, while subdivision reduces the variability of embedded nodes, which depends on the diversity of elements in the node vector. Subdivision reduces variability to order k-2 for node vectors with few components.

By increasing the scale of the B-spline curve, the new curve should be identical to the original curve, so

$$S_k(P;t)= \sum_{i=1}^{n+1} B_i^k(t)P_i = \sum_{i=1}^{n+1} B_i^{k+1}(t)P_i^*$$

where P_i^* are the polygon control points for the new curve.

The B-spline curve's scale or order can be increased to make it smoother compared to the control polygon. However, the presence of many internal elements in the node vector reduces continuity in the node's value, resulting in a plot close to a control polygon. This also increases the curve's flexibility by introducing additional elements. There are two basic methods of inserting nodes. The first method contains the so-called Oslo algorithm developed by (de Boor, 1972; Rogers. 2001) and his colleagues and the Prautzch algorithm, which consists of simultaneously inserting multiple elements into the node vector. The improved method consists of repeatedly inserting a single element into the vector node. Let it be an initial curve with a node vector like this:

$$S_k(P;t)= \sum_{i=1}^{n+1} B_i^k(t)P_i \ , \quad \tau \in [t_1, t_{2,} \dots t_{n+k+1}]$$

The new curve, which is obtained after inserting the nodes, is determined by

$$R_k(P;s)= \sum_{j=1}^{m+1} M_j^k(s)C_j \quad , \tau^* \in [y_1, y_{2,} \dots y_{m+k+1}]$$

m>n , is the new node vector and $S_k(P;t)= R_k(P;s)$.

The Oslo algorithm is used to determine the new points of the control polygon.

$$C_j=\sum_{i=1}^{n+1} \alpha_{i.j}^k P_i \qquad 1 \le i \le n, \ 1 \le j \le m,$$

Where $\alpha_{i.j}^k$ are given by recursive relation.

$$\alpha_{i.j}^k = \begin{cases} 1 & t_i \le \ y_j \le \ t_{i1} \\ 0 & ndryshe \end{cases} \quad ,$$

$$\alpha_{i.j}^k = \frac{y_{j+k-1}-t_i}{t_{j+k-1}-t_i}\alpha_{i.j}^{k-1} + \frac{t_{j+k}-y_{i+k-1}}{t_{j+k}-t_i}\alpha_{i+1.j}^{k-1} \ ,$$

Where $\sum_{i=1}^{n+1} \alpha_{i.j}^k = 1$.

The original knot vector, if uniform, periodic, or open, may not consistently produce end vector knots. A constant knot vector can be obtained by inserting middle knots for each non-empty interval.

The implementation of b-splines in signature authetification

The signature serves as a tool for verifying document authorship and authenticity, a task that requires specialized techniques. Mathematical and Machine Learning techniques have been introduced to assist in this process, making it more accessible to humans. This study examined physical signatures written on documents and obtained from the database of Kaggle's handwritten images www.kaggle.com/datasets/divyanshrai/handwritten-signatures.

The lack of a specific method (Kalluci et. al., 2023) for validating signatures makes document verification unreliable, prompting studies to explore methods to verify signatures using designated author (Oden et. al., 2003).

This study examines the easiest and most accessible method for a simple user using a smartphone to extract signatures, focusing on image extraction. However, the convenience of this method has introduced challenges in removing the signature from documents, including cleaning the image from data, which can affect the accuracy of the model.

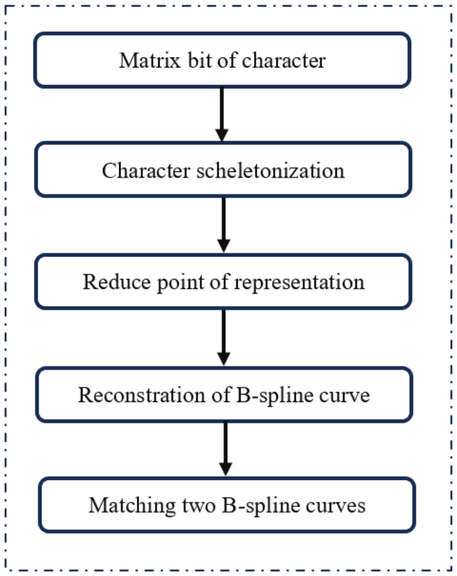
The study proposes methods to transform images, approximate signature shapes using B-Spline curves, save control points, and determine the Euclidean difference between these points and those extracted from verification signatures. Every time a signature is sent for verification, an approximation is made, and control points are extracted based on this, which are then compared with the control points stored in the training dataset.

The technique is restricted to verifying signatures used in the training dataset only.

Figure 1 provides a hierarchical model illustration.

Figure 1: Handwriting Recognition Steps

We have employed the following technologies and algorithms to put this approach into practice:


Matrix bit of character
Character scheletonization
Reduce point of representation
Reconstration of B-spline curve
Matching two B-spline curves

➢ **Python (Programming language):** Python is an ideal programming language for sensor integration and data processing due to its speed and simplicity. Its vast ecosystem of libraries, including OpenCV and RPLidar, provides a wide range of tools and algorithms for image processing and computer vision tasks. Python's cross-platform compatibility allows it to run smoothly on different operating systems, enhancing its suitability for data retrieval. It excels at real-time data processing when paired with specialized libraries like NumPy and SciPy. Its interoperability with APIs is particularly useful for cameras and LiDAR sensors.

➢ **OpenCV:** is an open-source library for computer vision tasks, offering cross-platform compatibility across various platforms. It supports various image and video file formats and is highly optimized in C++, enabling efficient real-time processing. OpenCV offers filtering, edge detection, object recognition, and more functions. It provides an intuitive API for interfacing with cameras and benefits from a large community. The library integrates with other machine-learning libraries and frameworks, combining computer vision with machine-learning techniques for advanced applications. Released under a BSD license, OpenCV is a popular choice in the computer vision community.

➢ **SciPy**: is an open-source scientific computing library for Python that offers a variety of modules and functions for various scientific and engineering applications. It builds on NumPy's capabilities and extends them to include tasks like optimization, signal processing, statistical analysis, linear algebra, integration, interpolation, linear algebra, statistics, special functions, spatial data structures, and data visualization. SciPy is a critical component of the scientific Python ecosystem and is often used with libraries like NumPy, Matplotlib, and Pandas for data analysis, visualization, and scientific computing. Its extensive feature set and user-friendly interface make it a valuable tool for scientists and data analysts working on complex mathematical problems in Python.

BFGS: is a local search optimization algorithm, a second-order method that uses the second-order derivative of an objective function. It is widely used for numerical optimization and is often used to adapt machine learning algorithms like logistic regression.

Mathematical morphology: is a branch of image processing and computer vision that analyzes binary images, grayscale images, and structured data. It focuses on shapes and structures in images and has applications in fields like image processing, computer vision, and pattern recognition. Key concepts include binary image operations, structure elements, and operations like dilation, erosion, opening, and closing. These operations help manipulate binary images, modify target images, and remove small objects and noise. Mathematical morphology is particularly useful in tasks requiring the geometry and structure of objects in images.

The step-by-step implementation process is as follows:

1) Dataset preparation:

- First, we focused on dataset preparation, which consists of capturing images of signatures by setting the images as part of the ID name of the author of the signature.
- We have generated signatures similar to the original for each author by introducing drastic changes to show that it is a forged signature.

2) Image processing:

This step consists of all the transformations to extract the binary form of the signature. This extract will be used later for approximation. Steps followed for image processing:

- Reading the image in Gray Scale: Reading the image with the OpenCV imread function, setting the reading mode in our case to GrayScale as a parameter, that is, to read the image in black and white.
- Convert the image to binary form: After reading the image in Gray Scale, we apply the conversion of the image to binary, get the values of the pixels, and compare them with 127; if this value is less than 127, it becomes zero; otherwise, it takes the value 255. This conversion is done so that we have algorithm confusion. The built-in library provides this method.
- Application of skeletonization: Skeletonization aims to remove pixels from the signature image without damaging its context. To build the B-Spline based on this extracted signature, we will only need the skeleton of the signature. The methodology used for signature skeletonisation is based on mathematical morphology.

✓ The first step that is part of the morphology is the erosion of the image content. This is done by removing the pixels next to the content without breaking the original shape. This makes it possible to remove excess without losing the context of the image. Erosion is based on a matrix with variable sizes depending on the case where we want to use it, and it is called kernel. In the case of the signature, the minimum size (3x3) was used, meaning we want to remove the "neighbours" of the pixels.

✓ This matrix iterates over the entire image with one step through all rows of the image matrix, checking how many pixels match the kernel. A threshold has been set which decides to write a pixel with a value of 255 when the size covered or reduced by the template is the total number of kernel pixels minus 2. This limit makes it possible whether it should be marked in the image or overwritten with the value 0.

✓ After erosion, dilation is applied to the image where the erosion was used. Iteratively, this process is repeated for each output of these processes until no more erosion can be done. The extension will return the context of the image but in a reduced way. This transition process reduces erosion without damaging the contents.

✓ This process of erosion and expansion continues until no more erosion can be done on the original image.

3) Implementation of the "Fit" function

Extracting features from the image: To approximate the B-Spline curve, we must determine its initial control points. The feature extraction method was examined to extract these points. The signature curve's edges, key points, sharp bends, and intersections between curves are selected as characteristics. In this study, the feature that extracts the contours of the signature was used. These removed references will be used as a guide for the assignment of checkpoints. These features determine the number of control points and, consequently, the order of the B-Spline curve.

4) Curve Fit

The control points have been extracted, and we are in the curve approximation step. To achieve this, the curve_fit function from the SciPy library was used. This function assigns the control points extracted from the feature extraction. Curve fitting is finding the best-fit curve (in this case, a Bézier curve) to a set of data points (control points extracted from the signature).

The curve-fitting steps are:

- The objective of the curve fitting step is to adjust the parameters of a Bézier curve so that it approximates the shape of the signature as closely as possible.
- Input data for curve fitting includes:

✓ `bezier_curve` function: This function represents the Bézier curve and is used to evaluate the curve for different parameter values.
✓ `t_values`: An array of `t` values ranging from 0 to 1, representing points along the Bézier curve.
✓ `control_points': Various control points extracted from the signature image.
✓ `guess_startup`: An initial estimate of the parameters of the Bézier curve.

- The `curve_fit` function is part of the SciPy library and is used for non-linear curve fitting. It performs a least-squares optimization to minimize the difference between the Bézier curve (defined by the `bezier_curve` function) and the actual control points.
- The optimization process iteratively adjusts the parameters of the Bézier curve to minimize the error (residuals) between the curve and the control points. The optimization algorithm seeks to find parameter values that result in the best possible fit to the data.
- Optimization parameters: In the `curve_fit' function, the optimization parameters are the parameters of the Bézier curve. These parameters include

the weights (multipliers) for the control points and the coordinates of the control points.

- Initialization: The initial guess (given as `$p_0$`) is crucial. It is an estimate of the parameters of the Bézier curve at the beginning of the optimization process. A reasonable initial guess helps the optimization algorithm converge to a better solution.
- Optimization Algorithm:

✓ The 'curve_fit' function uses a non-linear optimization algorithm (such as the Levenberg-Marquardt algorithm) to refine the parameter values iteratively.

✓ The algorithm adjusts the parameters to minimize the sum of the squared (least squares) differences between the Bézier curve and the control points.

- Convergence: The optimization process continues until a stopping criterion is met or a predetermined number of iterations is reached. The goal is to find parameter values that minimize the error.
- Fitted parameters:
- After the optimization, the function 'curve_fit' returns the provided parameters. These are the optimized parameter values that define the Bézier curve.
- Output: The fitted Bézier curve, represented by the optimized parameters, closely matches the shape of the signature. This curve can be used for various purposes, such as signature analysis, recognition, or smoothing.

The curve fitting step essentially adjusts the Bézier curve to approximate the shape of the signature as accurately as possible, based on the provided control points and an initial guess.

Experimental results

We need to construct a B-Spline curve using a subset of data points obtained from the signature images of two individuals. The B-spline curve specimen is saved in a database. To identify the person, we can calculate the differences between the test B-spline curves and all sample curves from the database using Euclidean distance.

By selecting the least value among all distances, we can determine the person's identity. Allow the software to acquire knowledge from three distinct sample signatures belonging to three individuals. The recognition process is illustrated in Figure 2.

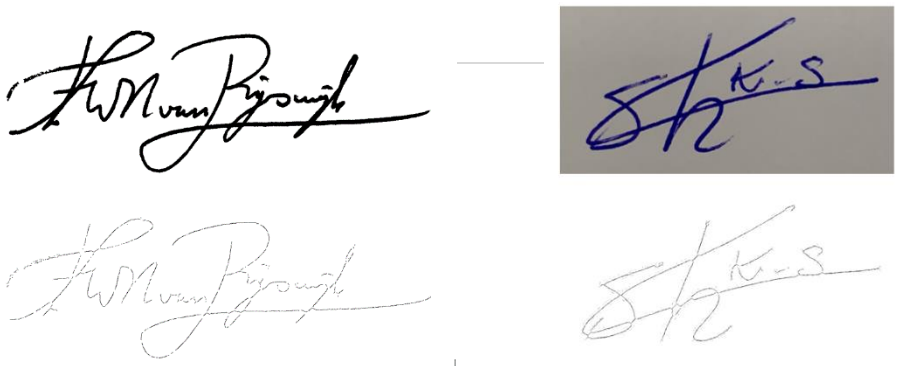


Figure 2 Process of recognition of the sign

The identification rate is 97%, and the verification error rate is 5%. This paper proposes non-uniform B-Spline curves for verifying the authenticity of signatures. B-spline curves can accurately record signature shapes, and the point projection method provides an accurate way to estimate the difference between two B-spline curves. The proposed technique achieves a high degree of identification and verification. Proposed future works include a more extensive test conducted on a database collected over some time.

Conclusions

This research focuses on confirming the legitimacy of individual signatures using the B-Spline algorithm. The process involves comparing the test signature to those trained using a sample, pre-processing input data, and identifying control points. An optimization tool like BGFC is used to fit the B-spline with the traced signature curve, aligning the traced curve's initial guess with the signature's shape.

The final stage involves calculating the Euclidean distance between the training samples and the test signature's control points, returning oriented samples with the sample signature's control points and the shortest Euclidean distance. The key feature of this method is that every reconstructed B-spline curve from a signature has the same degree and number of control points, reducing the inaccuracy of curve-matching.

The storage of sample data affects the recognition process's speed and accuracy. The algorithm has been tested on various signature styles and demonstrated good accuracy, demonstrating the accuracy and resilience of the suggested approach for matching B-spline curves.

References

Carl de Boor, (1972): On calculating with B-splines, Journal of Approximation Theory 6, 50-62

Farin G. (1988): Curves and surfaces for computer-aided geometric design: a practical guide, Academic Press INC, ISBN 0-12-24 9054-1.

Kalluci E., Noka E., Bani K., (2023): Optimization techniques in modeling hand calligraphy using B-splines, Bulletin of Natural Sciences, Issue 32.

Oden C., Ercil, A., Buke B., (2003): Combining implicit polynomials and geometric features for hand recognition, Pattern Recognition Letters, 24.

Rogers D. F. (2001): An Introduction to NURBS: With historical perspective, Academic Press.

Schoenberg I. J. (1969): Cardinal Interpolation and Spline Functions, J. Approximation Theory 2, 167-206.