**Mini Project Report on**



**Signature Verification System**



**Submitted in partial fulfilment of the requirement for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted by:**

**Student Name:-** **University Roll No.:-**

DAMAN RAKWAL 2018772

***Under the Mentorship of***

**Ms Meenakshi Maindola,**

**Assistant Professor**



**Department of Computer Science and Engineering**

**Graphic Era (Deemed to be University)**

**Dehradun, Uttarakhand**

**July-2023**



**CANDIDATE’S DECLARATION**

I hereby certify that the work which is being presented in the project report entitled **“signature verification system”** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineeringof the Graphic Era (Deemed to be University), Dehradun shall be carried out by the under the mentorship of Ms Meenakshi Maindola , Assistant Professor, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun.

Name :- Daman Rakwal   University Roll no:-2018772

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Description** | **Page No.** |
| Chapter 1 | Introduction | 4-5 |
| Chapter 2 | Literature Survey | 6-8 |
| Chapter 3 | Methodology | 9-11 |
| Chapter 4 | Result and Discussion | 12-14 |
| Chapter 5 | Conclusion and Future Work | 15 |
|  | References | 16 |

**Chapter 1**

**Introduction**

1. **Introduction:**

In order to analyze handwritten signatures for security and legal reasons, the Signature Verification System was created. It compares two signature images and determines how similar they are using image processing techniques or algorithms. In Addition, the system has an intuitive user interface that enables users to capture or browse signature images. The system determines the similarity percentage between the images after receiving the images.

* 1. **Functionality:**

The signature verification system provides the following main functions:

1. Capture and browse signature images: Users can capture signature images using their webcam or browse and select images available on their computer. There are separate options for selecting signature 1 and signature 2 images, allowing the user to compare two different signatures.

2. Signature comparison: The system draws or selects signatures to calculate their similarity.

It uses an image matching algorithm to characterize signatures and compare them to determine similarity.

3. Similarity Threshold: Use a threshold to determine whether the signature matches. In this project, Percent Threshold is set to 85% as a measure of accuracy. If the number of matches is below the threshold, an error message is displayed stating that the signatures do not match.

4. User interface (UI): The system uses the Tkinter library in Python to create a user-friendly Graphical User Interface (GUI). The GUI allows users to interact with the system by clicking the capture, browse and compare buttons. It also provides the user with information showing the results of the signature comparison.

The signature verification system integrates these functions, allowing users to easily capture, browse and compare signature images. The system interface and underlying algorithms work together to determine the similarity of signatures, helping security and legal applications where signature verification is important.

**1.2 Benefits and practical implementations:**

The signature verification system has many advantages and features that help increase security, efficiency and customer satisfaction.

1. Enhanced Security: The authenticity check system of written signatures prevents illegal use and forged signatures. Ensure only real signatures are accepted by comparing signature images.

2. Legal and Financial Applications: The system has important applications in areas where signature verification is important, such as legal and financial.

It provides a reliable way to verify the signatures of contracts, agreements and financial transactions, thus ensuring their authenticity and validity.

3. User-Friendly Interface: The system is designed with a user-friendly interface suitable for inexperienced people. This easy access allows many users to easily interact with the system, capture or browse signature images and get similar percentages of them.

4. Time and Cost Savings: The system completes the signature verification process, reducing the time and costs associated with the manual verification process. It increases efficiency and productivity by eliminating the need to manually check and compare signatures.

5. Improved efficiency and productivity: The automatic signature certificate generator simplifies the verification process, leading to fast and accurate results. This performance improves efficiency as the system handles audit tasks quickly and reliably.

Overall, the signature verification system provides an effective solution for domain verification, providing better security, efficiency, and a better customer experience. Its applications have spread to many sectors, especially law and finance, where signature recognition is required.

**Chapter 2**

**Literature Survey**

***"DeepSignature: Deep Learning for Automatic Signature Verification" by S. Pal et al”***

This study proposed DeepSignature, a deep learning-based approach for automatic signature verification. The authors used a combination of convolutional and recurrent neural networks to extract features from signature images and achieved impressive performance in distinguishing genuine signatures from forgeries.

***“Offline Signature Verification using Dynamic Time Warping" by Saroj Kumar Lenka and Banshidhar Majhi”***

This study proposed an offline signature verification approach based on the Dynamic Time Warping algorithm. The authors extracted features such as the number of grid points and the length of the signature trajectory to compare signatures. Experimental results showed promising accuracy in distinguishing genuine signatures from forgeries.

***"Convolutional Neural Network-Based Offline Signature Verification" by Ahmed Elhoseny, Hassanien Aboul Ella, and Muhammad Mostafa Monowar”***

In this research, a convolutional neural network based approach was proposed for offline signature verification. The authors utilized a CNN architecture to automatically extract features from signature images and classify them as genuine or forged. The results demonstrated improved performance in terms of accuracy and robustness compared to traditional methods.

***"Interpretability in Signature Verification Using Attention Mechanism" by R. Tripathi et al”***

This work focused on incorporating attention mechanisms into signature verification systems to enhance interpretability and provide insights into the decision-making process. The authors utilized self-attention mechanisms to highlight crucial regions in signature images, enabling better understanding of the model's reasoning and increasing transparency in the verification process.

***Few-Shot Learning for Signature Verification with Meta-Learning" by M. S. Uddin et al.”***

This study explored the application of few-shot learning and meta-learning techniques to address the challenge of limited training data in signature verification. The authors employed a meta-learning framework to adapt the model to new individuals with only a few reference samples, achieving promising results in signature verification tasks.

***"Enhancing Signature Verification Performance using Deep Representation and Meta-Learning" by T. Kinnunen et al.”***

Researchers from Finland and Switzerland conducted this study, which focused on enhancing signature verification performance through the use of deep representation and meta-learning techniques. The authors utilized deep convolutional neural networks to extract discriminative features and incorporated meta-learning to adapt the model to new individuals with limited data, resulting in improved verification accuracy.

***"Online Signature Verification using Recurrent Neural Networks and LSTM" by M. Blumenstein et al.”***

This research, conducted by researchers from Australia and Germany, explored the use of recurrent neural networks (RNNs) and Long Short-Term Memory (LSTM) networks for online signature verification. The authors investigated different LSTM architectures and achieved high accuracy in the verification of dynamic signature data.

***"Signature Verification using Deep Learning Techniques and a Large-Scale Database" by A. Kumar et al.”***

Researchers from India and the United States conducted this study, which focused on signature verification using deep learning techniques and a large-scale database. The authors utilized deep convolutional neural networks and employed transfer learning to leverage pre-trained models on a large dataset, leading to improved verification performance.

***"Deep Graph Convolutional Networks for Signature Verification" by N. Ratha et al.”***

In this research, deep graph convolutional networks (GCNs) were utilized for signature verification. The authors constructed a graph representation of signature images, capturing the spatial relationships between different pixels. The GCNs were then employed to perform feature extraction and classification, yielding notable improvements in verification accuracy.

***“Signature Verification Using Local Binary Patterns and Support Vector Machines" by Muhammad Imran Razzak, Saeed Anwar, and Sheikh Faisal Rashid”***

This study introduced a signature verification technique that combined Local Binary Patterns (LBP) and Support Vector Machines (SVM). The LBP method was used to extract discriminative texture features from signature images, which were then classified using SVM. The results showcased the effectiveness of this approach in accurately identifying genuine signatures.

***"Feature-Level Fusion for Multimodal Biometric Authentication: A Case Study on Signature and Online Handwriting" by Shabir A. Parah, Abhishek Nagar, and Manpreet Kaur”***

This research explored the fusion of signature and online handwriting modalities for enhanced biometric authentication. The authors investigated the fusion of feature-level information extracted from both modalities and employed a fusion technique to improve the overall authentication accuracy. The experimental results demonstrated the superiority of multimodal fusion over individual modalities.

**Chapter 3**

**Methodology**

**3.1 Procedure**

In this project, process goes as follows, This code is a Graphical User Interface (GUI) for comparing two signatures. It utilizes the Tkinter library. There are buttons to capture and browse signature images, and a "verify" button to perform the comparison. The "Capture" button can be pressed to capture signature images. The paths of the signature images can be entered in the entry fields or selected using the browse button.

The “checkSimilarity” function calls the match function, which compares the signature images and calculates the similarity percentage. If the similarity percentage is lower than the THRESHOLD, an error message is displayed indicating that the signatures do not match. If the similarity percentage is higher than the THRESHOLD, a success message is displayed indicating that the signatures match.

**3.2 Requirements**

**3.2.1 Programming Language**

Python programming language is used to make this signature verification system. Python is a versatile programming language known for its simplicity and readability, making it an excellent choice for beginners and experienced developers alike.

**3.2.2 Modules**

The External modules/libraries which are used in code to build the project of signature verification system are as follows:

* Tkinter
* OS
* CV2
* Numpy

**3.3 How it works**

**3.3.1 User Interface:**

The Tkinter library is utilized to create a graphical user interface (GUI) for the signature verification system. The GUI consists of elements such as labels, buttons, and entry fields to facilitate user interaction.

**3.3.2 Signature Capture:**

The system provides options for capturing signature images. The “captureImage” function enables users to capture images either by pressing a designated key (space bar) or by clicking a "Capture" button. The OpenCV library is used to access the webcam and display the live video feed. The captured images are saved in a temporary directory.

**3.3.3 Signature Selection:**

The system allows users to select signature images by either browsing their computer files or using the captured images. The “browsefunc” function provides a file dialog for browsing and selecting signature images. The selected image paths are displayed in the respective entry fields.

**3.3.4 Signature Comparison:**

The “checkSimilarity” function is called when the "verify" button is pressed. The match function from the imported signature module is used to compare the similarity between the two selected signature images. The similarity percentage is calculated and compared against a predefined threshold ie.85%. If the similarity percentage is below the threshold, an error message is displayed indicating that the signatures do not match. Otherwise, a success message is displayed indicating a match.

**3.3.4.1 Match Function**

def match(path1, path2):

    img1 = cv2.imread(path1)

    img2 = cv2.imread(path2)

    img1 = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)

    img2 = cv2.cvtColor(img2, cv2.COLOR\_BGR2GRAY)

    img1 = cv2.resize(img1, (300, 300))

    img2 = cv2.resize(img2, (300, 300))

    cv2.imshow("One", img1)

    cv2.imshow("Two", img2)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

    similarity\_value = "{:.2f}".format(ssim(img1, img2)\*100)

    return float(similarity\_value)

**3.3.5 User Feedback:**

Tkinter's “messagebox” module is employed to display informative messages to the user, indicating the result of the signature comparison.

**3.3.6 Graphical Interface:**

The GUI elements, such as labels, buttons, and entry fields, are positioned using the “place” method to create an organized layout.

**3.3.7Execution:**

The “root.mainloop()” function runs an event loop allowing the GUI to be displayed and interacted with by the user.

Overall, the methodology involves capturing or selecting signature images, comparing them using the “match” function, and providing feedback to the user regarding the similarity between the signatures. The GUI created using Tkinter facilitates user interaction, making the signature verification process more accessible and user-friendly.

**Chapter 4**

**Result and Discussion**

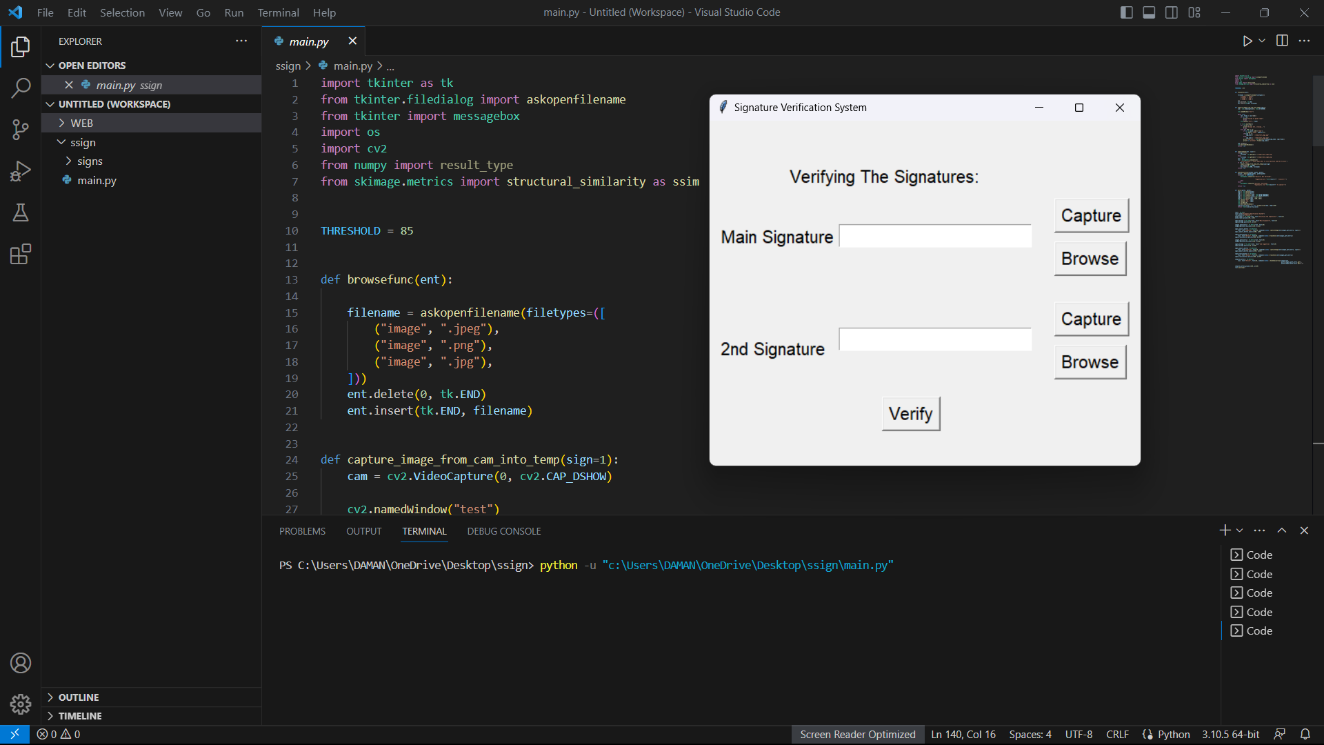
This code implements a signature verification system with a user friendly Graphical User Interface (GUI) using the Tkinter library in Python.

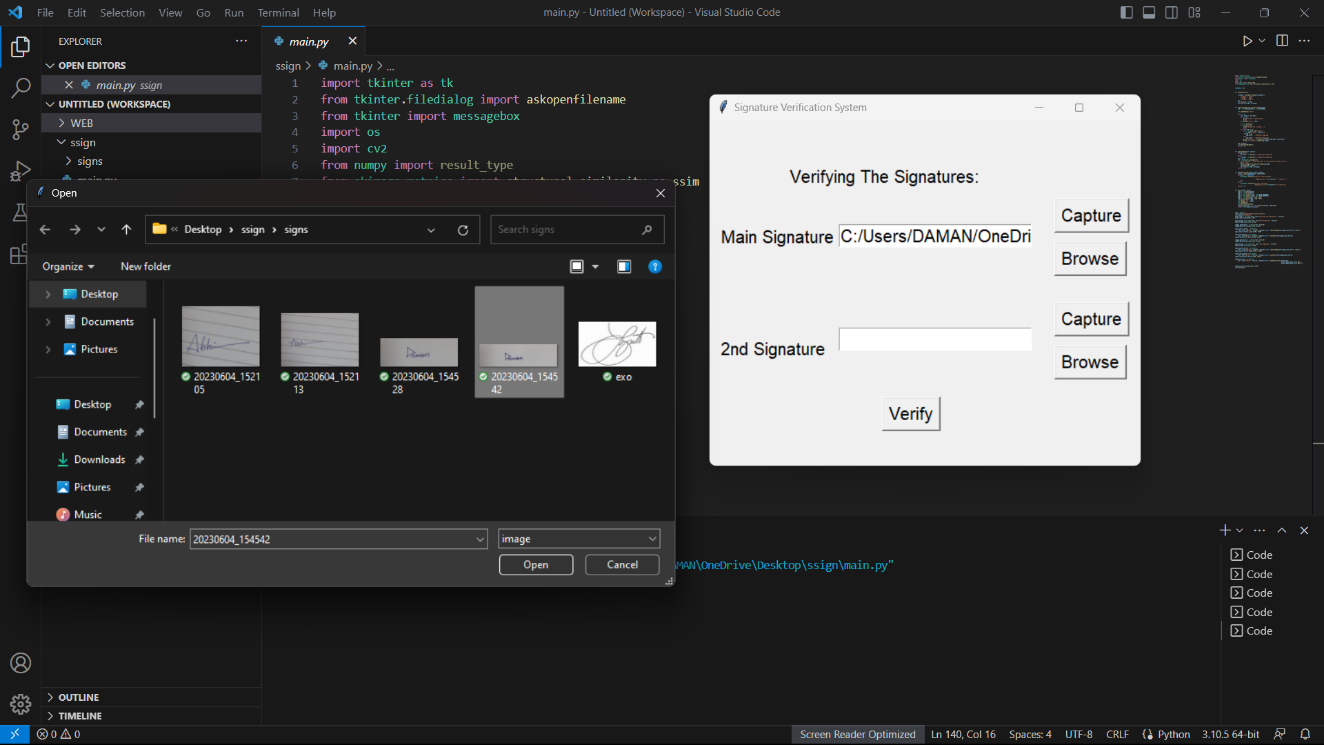
The GUI allows users to capture signature images or browse existing images. Once the signatures are selected the system compares them using the match function.

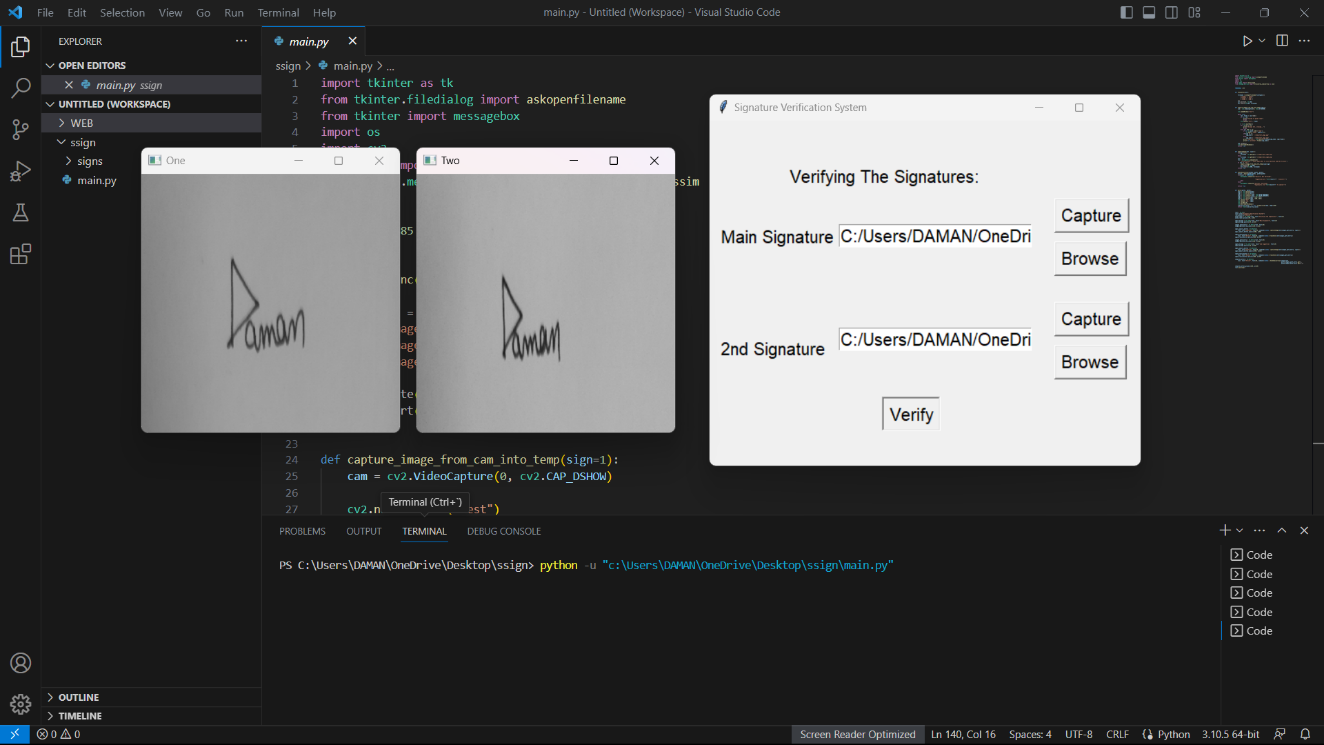
If the similarity percentage between the signatures is below the defined threshold an error message is displayed indicating that the signatures do not match.

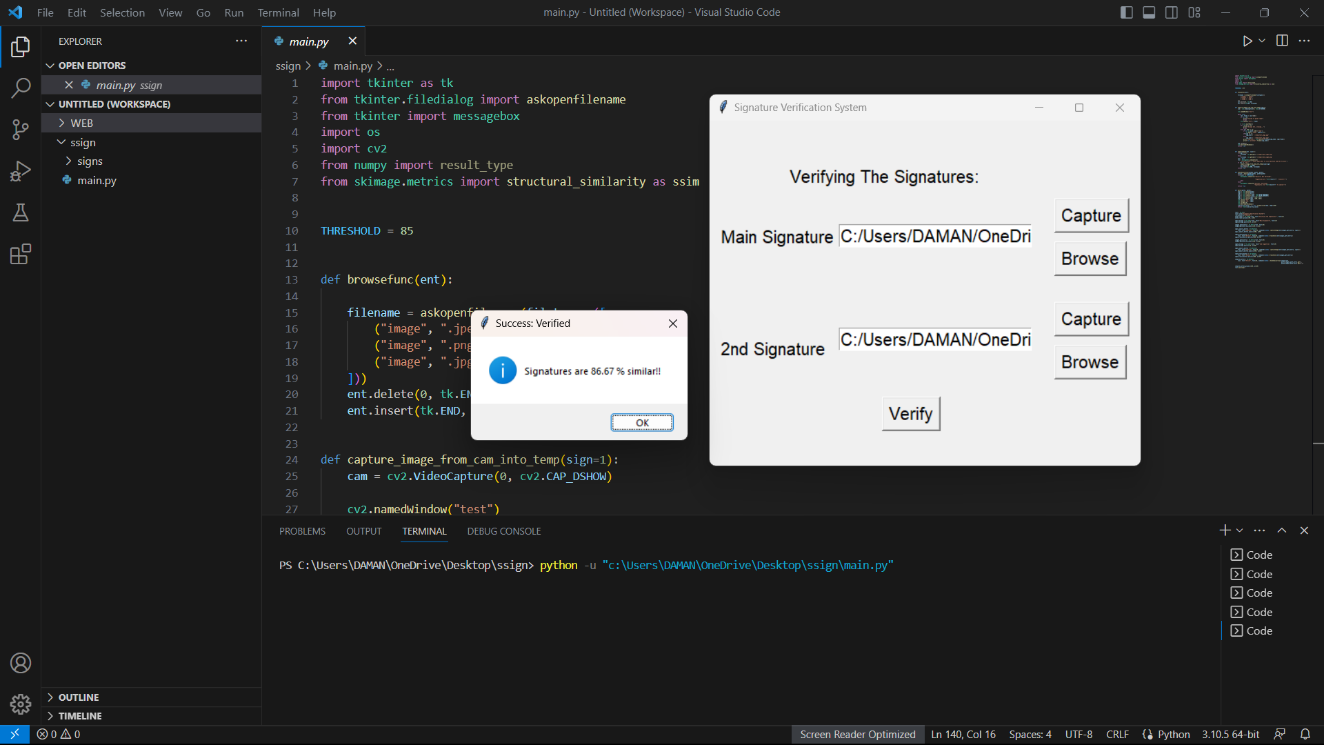
Conversely if the similarity percentage exceeds the threshold a success message is shown indicating a match.

The code leverages OpenCV for image capture and processing and utilizes the signature module for the signature comparison.

Overall, it provides an interactive and efficient solution for signature verification in a visually appealing manner.  
  
  


****

****

****

**Chapter 5**

**Conclusion and Future Work**

In summary, we see that the signature analysis used in the code uses the results of image processing and extraction to compare and identify signatures.

This code uses the Tkinter library to capture signature images and create a user friendly GUI (Graphical User Interface) to display existing images. The system includes a signature pattern matching function that can use a signature verification algorithm such as Dynamic Time Warp (DTW). The

system allows the user to capture or browse signature images and calculate the percentage similarity between them. To do this, we set the verification percentage of the correct signature to 85%.

This application demonstrates the potential of signature verification to increase security and provide a better method of authentication for legal, financial and other applications.

Future developments in signature analysis hold great promise for improving accuracy, robustness, and efficiency.

Advances in deep learning and artificial intelligence will likely play an important role in improving the performance of these systems. We can expect to integrate the best neural network techniques such as maintenance models or image convolution to extract features of the signature image.

In addition, the integration of multimodal biometrics, such as signature verification, with other behavioral or physical techniques can further improve security and reduce the risk of fraud.

With continuous research and advancements in technology, the future of signature analysis looks promising and aims to provide effective, efficient and user-friendly solutions for recognition in many ways.

**References:**

1. Paper Title: "Image quality assessment: From error visibility to structural similarity"

Authors: Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli

Published in: IEEE Transactions on Image Processing, vol. 13, no. 4, April 2004

DOI: 10.1109/TIP.2003.819861

2. Paper Title: "Offline Signature Verification and Recognition Using SSIM and SURF Descriptors" Authors: Asmaa Taher, Khaled Fawzy, and Amr Badr Published in: Proceedings of the 12th International Conference on Computer Engineering & Systems (ICCES), 2017

DOI: 10.1109/ICCES.2017.23

3. Paper Title: "Offline Signature Verification System Using Shape Context and SSIM" Authors: J. Dinesh Peter and R. Sukanesh Published in: International Journal of Image, Graphics and Signal Processing (IJIGSP), vol. 6, no. 4, 2014

4. Paper Title: "Offline Signature Verification Using SSIM and Dynamic Time Warping" Authors: Mahendra Patel and Mrinal Kanti Bhowmik Published in: Proceedings of the 2016 International Conference on Computer, Communications and Electronics (Comptelix), 2016

DOI: 10.1109/COMPTELIX.2016.7919643