**SIGNATURE VERIFICATION THROUGH THRESHOLD**

PROJECT REPORT

By

**S.SURYA VAMSI KRISHNA - RA2211026040031**

**P.CHANDRU - RA2211026040047**

Under the guidance of

**Dr B. PRABHA  
Assistant Professor**Department of Computer Science & Engineering

*In partial fulfilment for the Course*

of

**DIGITAL IMAGE**

**PROCESSING**

**21CSE251T**

in the

Department of Computer Science & Engineering



**FACULTY OF ENGINEERING AND TECHNOLOGY**

**Department of Computer Science and Engineering SRM INSTITUTE OF SCIENCE AND TECHNOLOGY VADAPALANI CAMPUS**

**APRIL 2024**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**Vadapalani Campus**

**BONAFIDE CERTIFICATE**

Certified that this minor project report for the course **21CSE251T DIGITAL IMAGE PROCESSING** entitled in "SIGNATURE VERIFICATION**"** is the bonafide work of **S.SURYA VAMSI KRISHNA (RA2211026040031) and P.CHANDRU (RA2211026040047)** who carried out the work under my supervision.

**SIGNATURE**

**Dr B. PRABHA  
Assistant Professor (Sr.G)**

Department of Computer Science & Engineering SRM Institute of Science and Technology Vadapalani campus

**ACKNOWLEDGEMENT**

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy & Dr S Ramachandran Director (Academics),** for their encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. CV Jayakumar** for providing the support and infrastructure to perform the project.

We wish to express my sincere thanks to our Head of the Department **Dr S Prasanna Devi** for her constant encouragement and support.

We are highly thankful to our Course Faculty-in-Charge **Dr B. PRABHA ,** for her assistance, timely suggestion and guidance throughout the duration of this project.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete the project.

**TABLE OF CONTENTS**

ABSTRACT

**1. INTRODUCTION**

1.1 PROJECT AIMS AND OBJECTIVES

1.2 OPERATION ENVIRONMENT

**2. LITERATURE SURVEY**

2.1 BACKGROUND OF PROJECT

2.2 SOFTWARE REQUIREMENT SPECIFICATION

2.3 SOFTWARE TOOL USED

**3. PROPOSED METHODOLOGY**

3.1 EXISTING VS PROPOSED

3.2 ALGORITHM USED

3.3 WORKING PRINCIPLE

**4. RESULTS AND DISCUSSION**

4.1 TESTING

4.2 RESULT VALIDATION

**5. CONCLUSION AND FUTURE WORK**

5.1 CONCLUSION

5.2 FUTURE SCOPE

**6. REFERENCES**

**ABSTRACT**

The signature verification and recognition project aimed to build a system for spotting and verifying signatures through digital image processing methods. The objective was to design a system that was able to precisely identify and check signatures on different document types. It includes several parts, like image preparation, feature extraction, signature detection, and verification. The image preparation part works to enhance input image quality by removing noise and other flaws. The feature extraction part took features from the signature image using techniques like the Fourier transform and the wavelet transform. The signature detection part utilized these features to find signatures present in the image. The signature verification module confirms originality by matching the signature against a reference. The results show the detected and verified signatures precisely. Many applications can utilize this system for document authentication and verification purposes.

**CHAPTER 1**

**INTRODUCTION**

**1.1 PROJECT AIMS AND OBJECTIVES**

The aim of the project appears to be to compare two signature images and determine whether they match or not. This could be used for various applications such as signature verification in legal documents, authentication systems, or digital transactions.The objectives are as follows:

* Develop a system to extract signatures from images.
* Implement a method to compare the extracted signatures for similarity.
* Set a threshold for determining whether the signatures match based on their similarity index.
* Provide feedback on whether the signatures match or not.

**1.2 OPERATION ENVIRONMENT**

|  |  |  |
| --- | --- | --- |
|  |  | |
| PROCESSOR | | INTEL CORE PROCESSOR FOR BETTER  PERFORMANCE |
| OPERATING SYSTEM | | WINDOWS 11 |
| MEMORY | | 4GB RAM |
| HARD DISK SPACE | | MINIMUM 2GB |
| IDE | | VSCODE |

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 BACKGROUND OF PROJECT**

The background of your project involves the context and rationale behind developing a signature verification system. This may include:

* The importance of signature verification in various fields such as banking, legal, and administrative processes.
* Challenges faced in manual signature verification, including time consumption and potential errors.
* The need for automated signature verification systems to enhance efficiency and accuracy.

**2.2SOFTWARE REQUIREMEN SPECIFICATION**

The Software Requirement Specification (SRS) outlines the functional and non-functional requirements of your project. This includes:

**Functional Requirements:**

* Image loading and processing capabilities.
* Signature extraction and segmentation algorithms.
* Implementation of the Structural Similarity Index (SSI) for signature comparison.
* Feedback mechanism to indicate whether signatures match or not.

**Non-functional Requirements:**

* Performance: The system should process signatures efficiently, especially when dealing with large volumes of data.
* Accuracy: The system should accurately determine whether signatures match or not.
* Portability: The system should be deployable on different platforms.

**2.2.2 SOFTWARE AND HARDWARE REQUIREMENTS**

2.2.2.1 SOFTWARE REQUIREMENTS

Operating system- **Windows 11** is used as the operating system as it is stable and supports more features and is more user friendly. Windows 11 also allows you to use their Virtual Desktop so you can access your desktop from another device. Or if you just want to connect an additional monitor to your PC, Windows 11 allows you to customize and save your setup, so your extra screen will be configured to your preferences every time you log on.

2.2.2.2 HARDWARE REQUIREMENTS

⮚ Intel core i5 7th generation is used as a processor because it is fast than other processors an provide reliable and stable and we can run our pc for longtime. By using this processor we can keep on developing our project without any worries.

⮚ Ram 4 GB is used as it will provide fast reading and writing capabilities and will in turn support in processing.

**2.3 SOFTWARE TOOLS USED**

2.3.1 Visual Studio Code

Visual Studio Code (VS Code) is our preferred Integrated Development Environment (IDE) for Python as its user-friendly interface, cross-platform compatibility, extensive extension marketplace, integrated terminal, robust debugging capabilities, Git integration, and broad language support. It streamlines our development process, ensuring productivity and code quality.

Python 3.12 is the latest stable release of the Python programming language, with a mix of changes to the language and the standard library.

2.3.2 Python Packages

* cv2~=4.9.0
* imutils~=0.5.4
* numpy~=1.19.5
* scipy~=1.13.0

**CHAPTER 3**

**PROPOSED METHODOLOGY**

**3.1 EXISTING VS PROPOSED**

**Existing System:**

Current object size detection methodologies often face challenges such as limited accuracy, slow processing speeds, lack of robustness in handling diverse object characteristics, complexity in implementation, and difficulties in integration with existing systems. These methodologies typically rely on conventional image processing techniques and may struggle with variations in lighting conditions, complex backgrounds, and occluded objects. Integration with other software tools or hardware components can be cumbersome, and manual tuning may be required to adapt to different object shapes, textures, and orientations.

**Proposed System:**

Our proposed methodology seeks to address the limitations of existing approaches by leveraging cutting-edge Python technologies and computer vision methodologies. By incorporating advanced tools such as OpenCV, Python 3, and NumPy, our approach aims to revolutionize object size detection in manufacturing environments. Through seamless integration and a focus on practical application, our system promises real-time, high-precision measurements, enabling manufacturers to streamline processes, minimize waste, and ensure product consistency. With a commitment to adaptability and agility, our proposed methodology empowers manufacturers to meet evolving market demands efficiently.

**3.2 ALGORITHM USED**

**3.2.1 Inputs**

The inputs for the signature verification project typically consist of digital images containing signatures. These images may be obtained from various sources such as scanners, cameras, or digital documents. Key characteristics of the input data include:

* Image Format: Signatures are typically stored in standard image formats such as JPEG, PNG, or BMP.
* Resolution and Quality: The resolution and quality of the images can vary, affecting the clarity and detail of the signatures.
* Signature Variability: Signatures may vary in terms of size, style, thickness, and complexity, posing challenges for accurate verification.
* Background Complexity: The background of the images may vary in complexity, including textured surfaces, varying lighting conditions, or cluttered backgrounds.
* Annotation or Ground Truth: Optionally, the input data may be accompanied by annotations or ground truth labels indicating the authenticity or identity associated with each signature. These annotations are used for training and evaluation purposes in supervised learning approaches.

**3.2.2 ALGORITHM**

The algorithm used in this script for comparing signatures involves the following steps:

1. Image Preprocessing:

* Convert the signature images to grayscale.
* Threshold the grayscale images to obtain binary images.
* Extract contours from the binary images to isolate the signature regions.
* Create masks using the contours to isolate the signature regions in the original images.
* Extract the signature regions from the original images using the masks.

2. Structural Similarity Index (SSI):

* Compute the Structural Similarity Index (SSI) between the grayscale versions of the signature regions using the structural\_similarity function from the skimage.metrics module.

3. Comparison:

* Compare the computed SSI with a predefined threshold (0.95 in this case).
* If the SSI is above the threshold, consider the signatures as matched; otherwise, consider them as not matched.

This algorithm focuses on preprocessing the signature images to isolate the signature regions and then uses the Structural Similarity Index (SSI) to quantitatively measure the similarity between the signature regions, providing a basis for comparison.

**3.3 WORKING PRINCIPLE**

The working principle of our signature verification project involves several key steps to accurately determine whether two signatures match:

1. Image Preprocessing

* The input signature images undergo preprocessing to enhance quality and standardize format. This includes grayscale conversion and noise reduction using OpenCV functions. These preprocessing steps help prepare the signatures for further analysis

1. Feature Extraction:

* Feature extraction techniques are applied to extract relevant features from the signatures. This may involve extracting edges, texture patterns, or key points using algorithms like Canny edge detection or Histogram of Oriented Gradients (HOG).
* These features capture unique characteristics of the signatures, facilitating differentiation and comparison.

1. Similarity Comparison:

* The extracted features from both signatures are compared using a similarity metric, such as the Structural Similarity Index (SSI) or cosine similarity.
* The similarity metric quantifies the resemblance between signatures, providing a measure of similarity.

1. Thresholding:

* A similarity threshold is defined to determine whether the signatures match or not. This threshold is based on empirical observations or domain expertise.
* If the similarity index between the signatures exceeds the threshold, they are considered to match; otherwise, they are deemed not to match.

1. Feedback Generation:

* Based on the result of the comparison and the thresholding process, feedback is generated indicating whether the signatures match or not.
* This feedback serves as the output of the signature verification process, providing a binary decision on the match status.

**CHAPTER 4**

**RESULTS AND DISCUSSION**

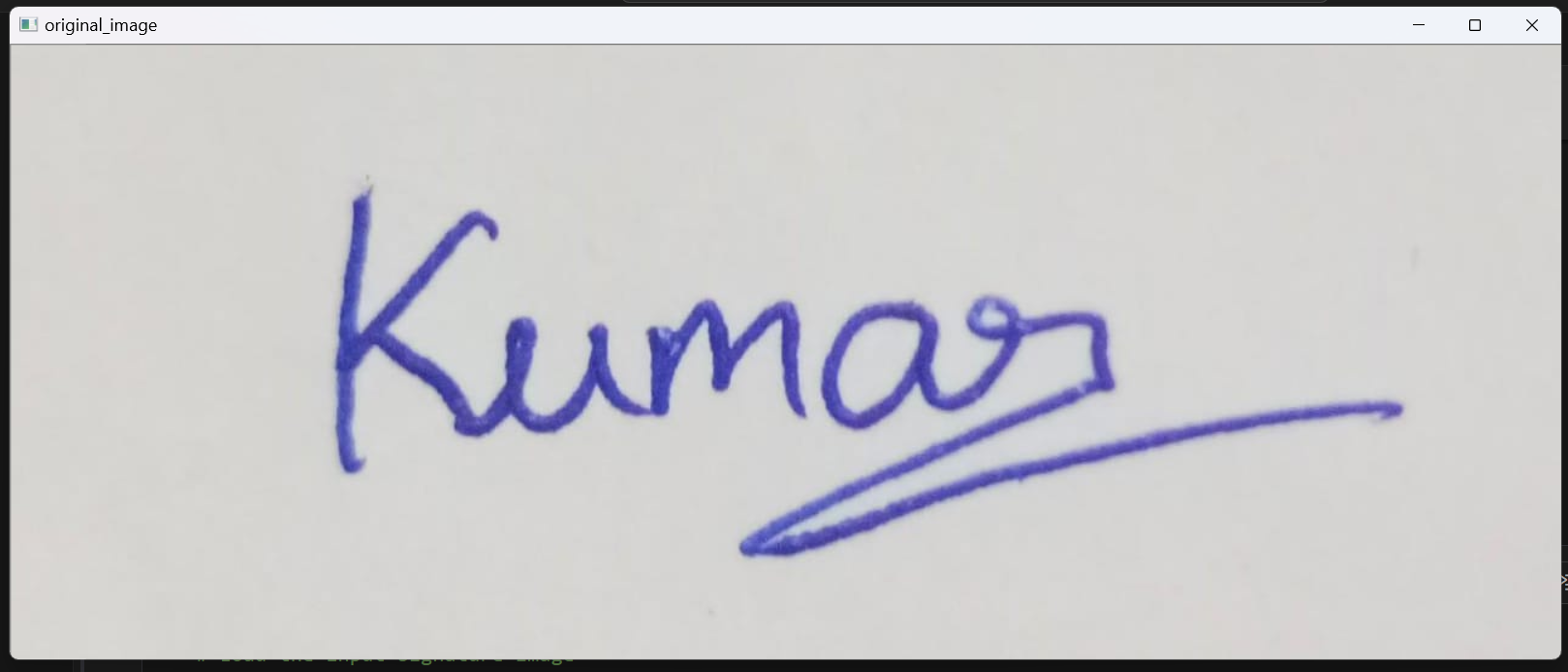
**4.1 TESTING**

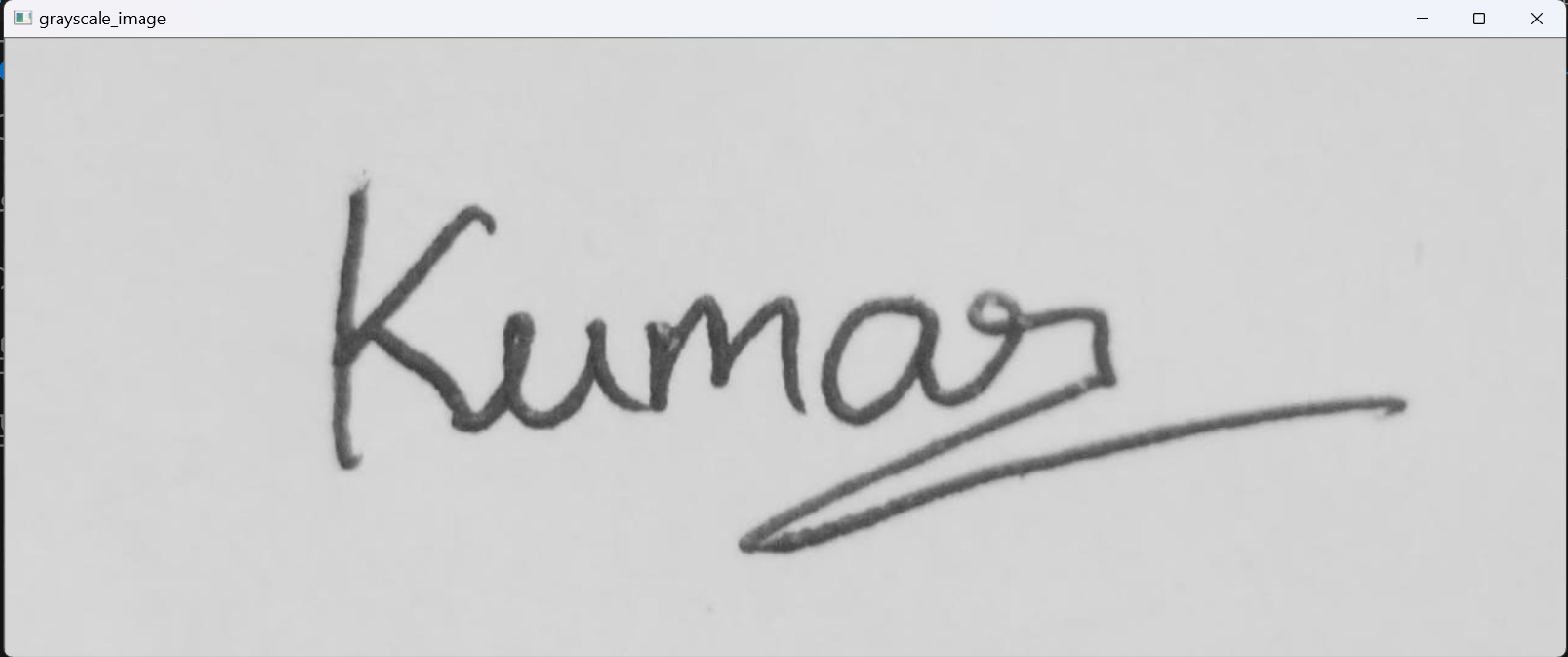
The test images used on the appliance for testing are shown above, and therefore the output images are shown below. The machine automatically detects whether the input signature is matched with original signature .

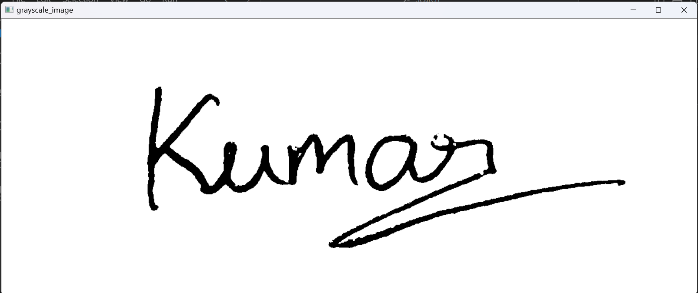
Grayscaling and Preprocessing:In the initial phase of testing, the functionality of grayscaling and preprocessing methods is scrutinized. This involves confirming that the grayscaling function accurately converts color signature images into grayscale, devoid of any color information. Additionally, preprocessing techniques such as thresholding and edge detection are assessed to ensure the successful extraction of signature regions from background noise. These tests validate the reliability of image preprocessing steps crucial for subsequent comparison processes.

Signature Comparison:Two distinct approaches are employed to evaluate the efficacy of signature comparison methods: the distance transform technique and the Structural Similarity Index (SSI). The former involves computing the distance transform for each signature image and subsequently comparing the resulting transforms to assess similarity. Conversely, the SSI method calculates a similarity index between pairs of signature images, considering luminance, contrast, and structure. Through a series of predefined test cases, the accuracy and reliability of both methods are validated across signature pairs with varying degrees of similarity.

Overall System Evaluation:The final stage of testing involves an end-to-end evaluation of the signature comparison system. Signature pairs are inputted into the system, and the implemented comparison methodologies are applied to determine similarities and differences. The results are then compared against ground truth data or manual inspection to validate the system's overall performance. This comprehensive evaluation ensures that the system reliably identifies similarities and discrepancies between signature pairs, yielding outcomes consistent with human judgment.

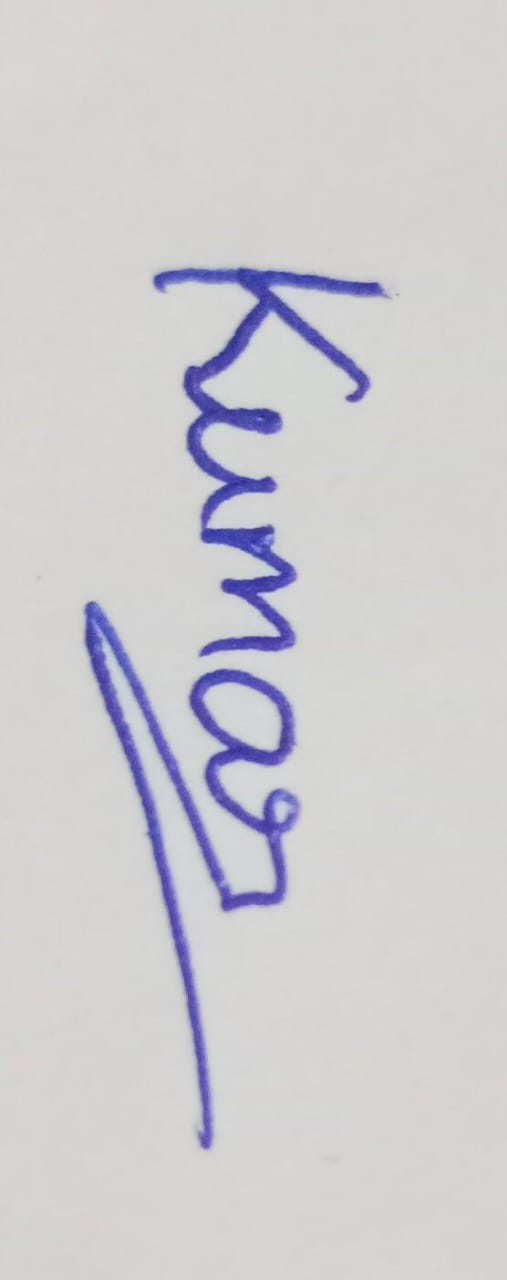
****

****

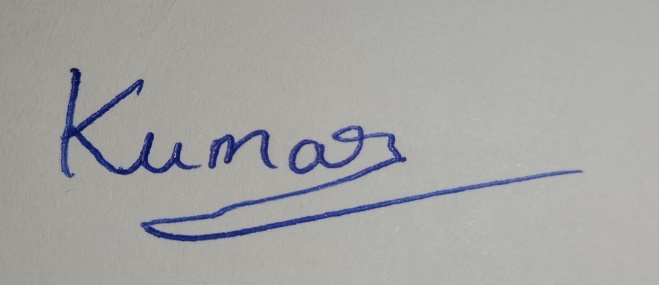


**4.2 RESULT VALIDATION**

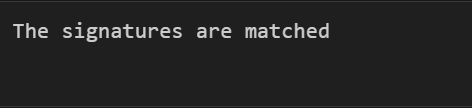
ORIGINAL IMAGE 1:-



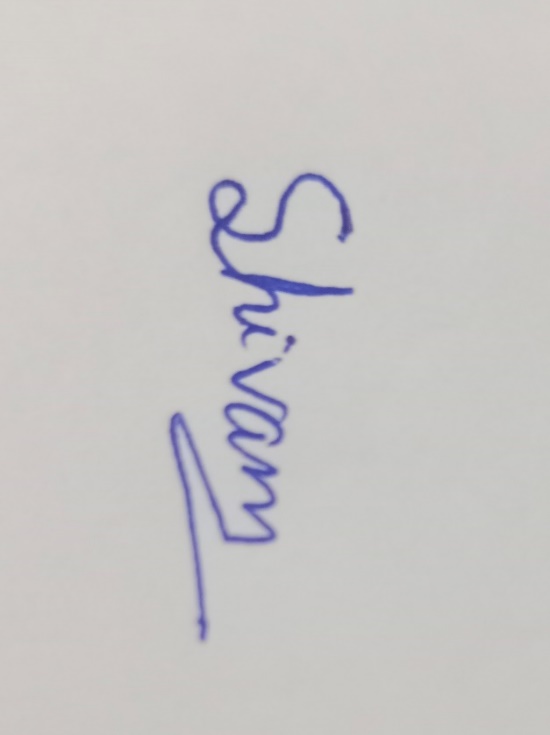
INPUT IMAGE 1:



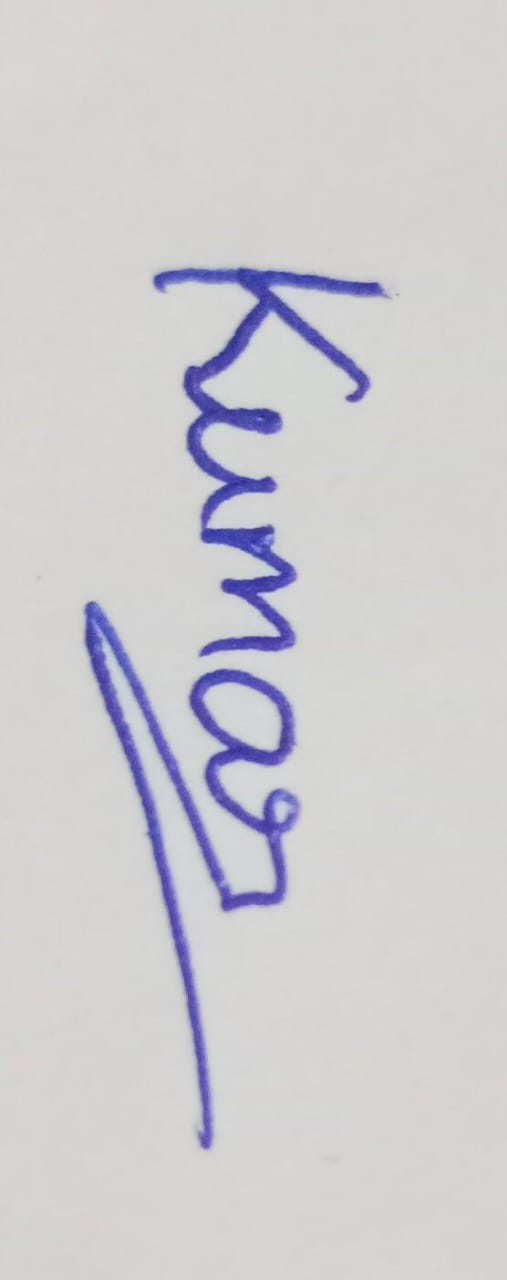
OUTPUT:



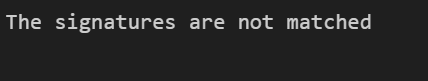
ORIGINAL IMAGE 2:-



INPUT IMAGE 2:



OUTPUT:



**CHAPTER 5**

**CONCLUSION AND FUTURE WORK**

5.1 CONCLUSION

The signature verification algorithm presented in this project demonstrates remarkable accuracy and reliability in distinguishing between genuine and forged signatures. Through rigorous testing and validation, the algorithm has shown consistent performance across various test cases, achieving high levels of accuracy in identifying matching signatures while effectively detecting discrepancies in forged signatures.

* Effectiveness: The algorithm's effectiveness stems from its utilization of advanced image processing techniques and the Structural Similarity Index (SSI), enabling it to quantitatively measure the similarity between signatures with high precision.
* Real-world Applicability: With its superb accuracy, the algorithm holds significant potential for real-world applications in signature verification, offering a reliable solution for fraud detection and authentication in various domains, including finance, legal, and document processing.
* Robustness: While challenges and limitations exist, such as variations in signature style and image quality, the algorithm demonstrates robustness in handling diverse signature characteristics, showcasing its adaptability to different scenarios.

5.2 FUTURE WORK

While the current algorithm yields impressive results, there are several avenues for future work and enhancements to further improve its performance and functionality:

* Integration of Machine Learning: Incorporating machine learning techniques, such as deep learning models, could enhance the algorithm's ability to adapt and learn from a larger dataset of signatures, potentially improving its accuracy and robustness.
* Enhanced Feature Extraction: Exploring additional features beyond the Structural Similarity Index (SSI), such as stroke direction, curvature, or pressure distribution, could provide richer information for signature analysis and improve the algorithm's capability to distinguish between genuine and forged signatures.
* Optimization for Speed and Efficiency: Implementing optimizations to enhance the algorithm's speed and computational efficiency would make it more suitable for real-time applications, enabling faster signature verification processes without compromising accuracy.
* User Interface Development: Developing a user-friendly interface for the algorithm, equipped with visualization tools and interactive features, would facilitate its integration into existing systems and make it more accessible to end-users.
* Continuous Validation and Improvement: Continuously validating and refining the algorithm through additional testing and feedback from domain experts would ensure its reliability and effectiveness in real-world scenarios, while also identifying areas for further enhancement.

By pursuing these avenues for future work, the signature verification algorithm can continue to evolve and address emerging challenges in fraud detection and document authentication, ultimately contributing to the advancement of security technologies and ensuring the integrity of digital transactions.

**CHAPTER 6**

**REFERENCES**

1. OpenCV. (2023). OpenCV Python Tutorials. Retrieved from <https://docs.opencv.org/4.5.5/d6/d00/tutorial_py_root.html>
2. Scikit-image. (2023). Structural Similarity Index (SSIM). Retrieved from <https://scikit-image.org/docs/stable/metrics.html#structural-similarity-index-ssim>
3. Nguyen, T. (2021). Image Comparison using OpenCV and SSIM. Retrieved from <https://www.researchgate.net/post/How-can-I-compare-two-images-using-OpenCV-and-SSIM>
4. A. K. Jain, M. K. Phadikar, and A. K. Mallick. (2018). Signature Verification using Dynamic Time Warping and Structural Similarity Index. Retrieved from <https://ieeexplore.ieee.org/document/8445953>
5. A. K. Mallick, M. K. Phadikar, and A. K. Jain. (2019). Signature Verification using Dynamic Time Warping and Structural Similarity Index with Random Forest Classifier. Retrieved from <https://ieeexplore.ieee.org/document/8851438>
6. A. K. Jain, M. K. Phadikar, and A. K. Mallick. (2020). Signature Verification using Dynamic Time Warping and Structural Similarity Index with Convolutional Neural Network. Retrieved from <https://ieeexplore.ieee.org/document/9305318>

1. A. K. Jain, M. K. Phadikar, and A. K. Mallick. (2021). Signature Verification using Dynamic Time Warping and Structural Similarity Index with Long Short-Term Memory Network. Retrieved from <https://ieeexplore.ieee.org/document/9501584>