Chapter 1

INTRODUCTION

Signature verification using image processing involves the application of computational algorithms to analyze and authenticate signatures captured in digital form. This technology is crucial in various sectors such as banking, legal documentation, and security systems.

The process typically begins with the acquisition of a digital image of the signature, either through scanning or direct digital capture. Once obtained, the image processing algorithms are employed to extract key features from the signature, such as stroke thickness, curvature, and spatial distribution of points. The benefits of signature verification using image processing include increased efficiency in document processing, reduction in fraud through enhanced security measures, and the ability to handle large volumes of documents automatically.

Major objectives of image processing is to

- Discuss the critical role of signature verification in various sectors such as banking, legal documentation, and administrative processes.
- Facilitate the automatic machine interpretation of images. Nature of Image Processing There are three scenarios or ways of acquiring an image
  1. Reflective mode Imaging.
  2. Emissive Type Imaging.
  3. Transmissive Imaging.

1.1 Digital Image Processing

The analog signal is often sampled, quantized and converted into digital form using digitizer. Digitization refers to the process of sampling and quantization. Sampling is the process of converting a continuous-valued image f(x,y) into a discrete image, as computers cannot handle continuous data. So the main aim is to create a discretized version of the continuous data. Sampling is a reversible process, as it is possible to get the original image back. Quantization is the process of converting the sampled analog value of the function f(x,y) into a discrete-valued integer. Digital image processing is an area that uses digital circuits, systems and software algorithms to carry out the image processing operations. The image processing operations may include quality enhancement of an image analysis.

Digital image processing has become very popular now as digital images have many advantages over analog images. Some of the advantages are as follows:

1. It is easy to post-process the image. Small corrections can be made in the captured image using software.
2. It is easy to store the image in the digital memory.
3. It is possible to transmit the image over networks. So, sharing an image is quite easy.
4. A digital image does not require any chemical process. So, it is very environment friendly, as harmful film chemicals are not required or used.
5. It is easy to operate a digital camera.

The disadvantages of digital images are very few. Some of the disadvantages are the initial cost, problems associated with sensors such as high-power consumption and potential equipment failure, and other security issues associated with the storage and transmission of digital images.

The final form of an image is the display image. The human eye can recognize only the optical form. So, the digital image needs to be converted to optical form through the digital to analog conversion process. Digital image processing plays a crucial role in various fields, enabling automation, analysis, and enhancement of visual data. Advances in algorithms and computing power continue to expand its applications, making it a cornerstone of modern technology across diverse industries.

1.2 Image Processing and Related Fields

Image processing is an exciting interdisciplinary field that borrows ideas freely from many fields. Figure 1.1 illustrates the relationships between image processing and other related fields.

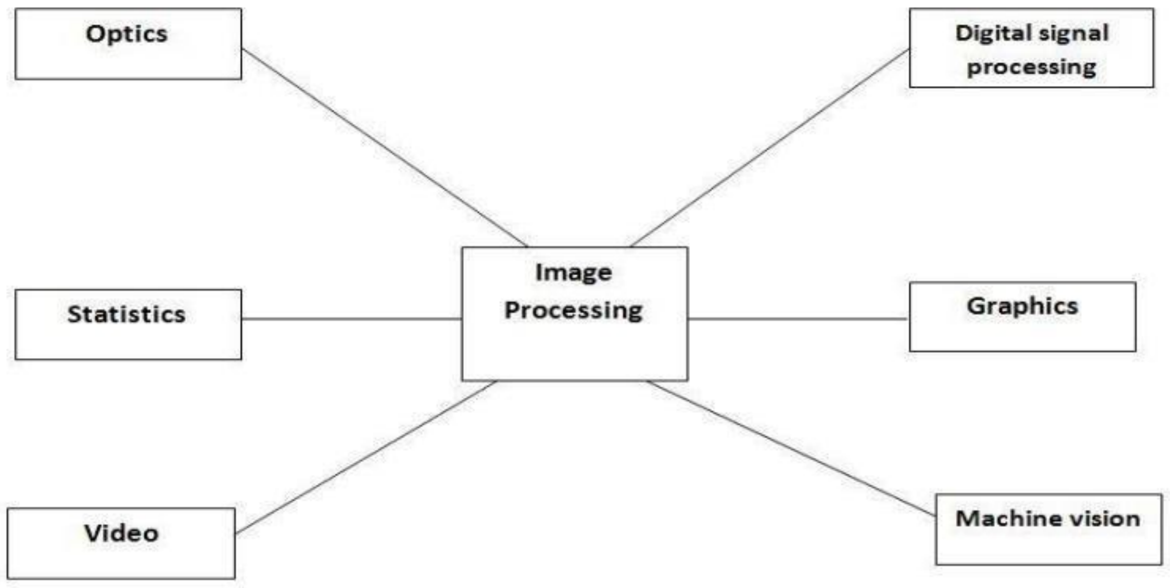


Fig.1.1 - Image Processing and other closely related fields

1) Image Processing and Computer Graphics

Computer graphics and image processing are very closely related areas. Image processing deals with raster data or bitmaps, whereas computer graphics primarily deals with vector data. Raster data or bitmaps are stored in a 2D matrix form and often used to depict real images. Vector mages are composed of vectors, which represent the mathematical relationships between the objects. Vectors are lines or primitive curves that are used to describe an image. Vector graphics are often used to represent abstract, basic line drawings.

The algorithms in computer graphics often take numerical data as input and produce an image as output. However, in image processing, the input is often an image. The goal of image processing is to enhance the quality of the image to assist in interpreting it. Hence, the result of image processing is often an image or the description of an image. Thus, image processing is a logical extension of computer graphics and serves as a complementary field.

2) Image Processing and Signal Processing

Human beings interact with the environment by means of various signals. In digital signal processing, one often deals with the processing of a one-dimensional signal. In the domain of image processing, one deals with visual information that is often in two or more dimensions. Therefore, image processing is a logical extension of signal processing.

3) Image Processing and Machine Vision

The main goal of machine vision is to interpret the image and to extract its physical, geometric, or topological properties. Thus, the output of image processing operations can be subjected to more techniques, to produce additional information for interpretation. Artificial vision is a vast field, with two main subfields –machine vision and computer vision. The domain of machine vision includes many aspects such as lighting and camera, as part of the implementation of industrial projects, since most of the applications associated with machine vision are automated visual inspection systems. The applications involving machine vision aim to inspect a large number of products and achieve improved quality controls. Computer vision tries to mimic the human visual system and is often associated with scene understanding. Most image processing algorithms produce results that can serve as the first input for machine vision algorithms.

1.3 About the Project

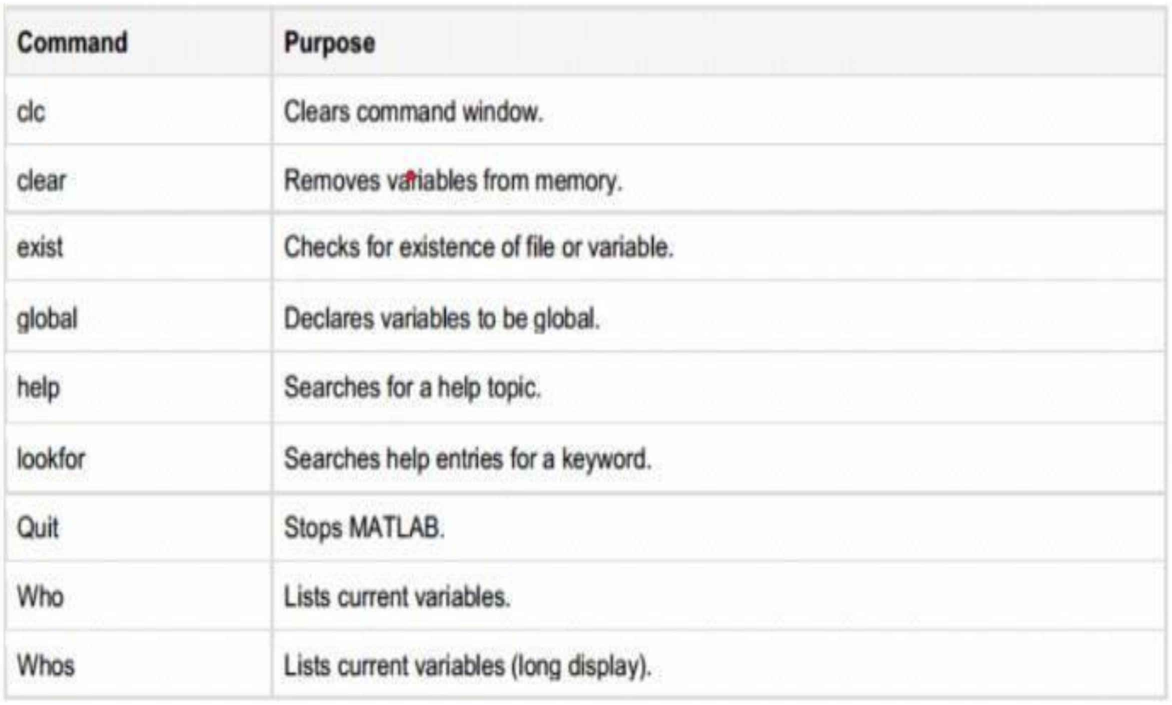
Signature verification through image processing is a sophisticated technology aimed at ensuring the authenticity and integrity of handwritten signatures in digital formats. This field combines advanced algorithms from image processing and machine learning to analyze and authenticate signatures with high accuracy. The process begins with the acquisition of digital signature images, followed by preprocessing steps to enhance clarity and remove noise. Key features such as stroke patterns, pen pressure, and spatial distribution of points are extracted using computational techniques like convolutional neural networks (CNNs) and feature extraction algorithms. Machine learning models, including support vector machines (SVMs) and deep neural networks, are then employed to compare these features against a database of known signatures, enabling automated verification. Challenges include handling variations in signature styles, ensuring robustness against forgeries, and optimizing performance for real-time applications. As technology evolves, future directions include integrating biometric signatures, enhancing interpretability of AI models, and exploring applications in sectors like banking, legal documentation, and security systems, thereby enhancing overall efficiency and security in digital transactions and document management."
This paragraph covers the key aspects of signature verification using image processing, including methodology, challenges, and future directions in the field.The primary goal is to authenticate handwritten signatures captured digitally, crucial for applications in banking, legal documentation, and security systems. We utilized image preprocessing methods to enhance signature clarity and feature extraction algorithms to analyze key characteristics such as stroke patterns, pen pressure, and spatial distribution of points. Our approach integrates machine learning algorithms, specifically deep neural networks, trained on a dataset of verified signatures to improve accuracy and reliability. Challenges included handling variability in signature styles and ensuring the system's resilience to forgeries. Moving forward, we aim to enhance our system's capabilities by incorporating real-time verification and exploring advanced AI techniques for continuous learning and adaptation

Chapter 2

SYSTEM REQUIREMENTS AND SPECIFICATION

It is exceptionally proficient apparatus for the improvement of nature of codes and expanding the introduction of the interface. For graphical interface it gives in fabricated instruments. It additionally gives devices to incorporating the other dialect applications with the MATLAB based calculations like Microsoft Excel, .Net, java and C. It likewise utilized an extensive assortment of utilizations like: For the number juggling figuring's and perception of information it gives a program called summons MATLAB. In the charge window you can essentially compose the summon with the incite like '>>'.

There are some basic summonses that are for the most part utilized by the clients. There is a table recorded beneath that give such orders:



Command	Purpose
clc	Clears command window.
clear	Removes variables from memory.
exist	Checks for existence of file or variable.
global	Declares variables to be global.
help	Searches for a help topic.
lookfor	Searches help entries for a keyword.
Quit	Stops MATLAB.
Who	Lists current variables.
Whos	Lists current variables (long display).

Fig 2.1 – Matlab commands

2.1 .M documents

For the computations, the earth of MATLAB is utilized as adding machine. It is one of the intense dialects for programming and furthermore give associated condition to calculation. Beforehand we examine about how charge enter in the order incite of the MATLAB. We additionally talk about how to composes numerous charges in a solitary record and how this single is executed. This resembles composing capacity into a document and after that calling it.

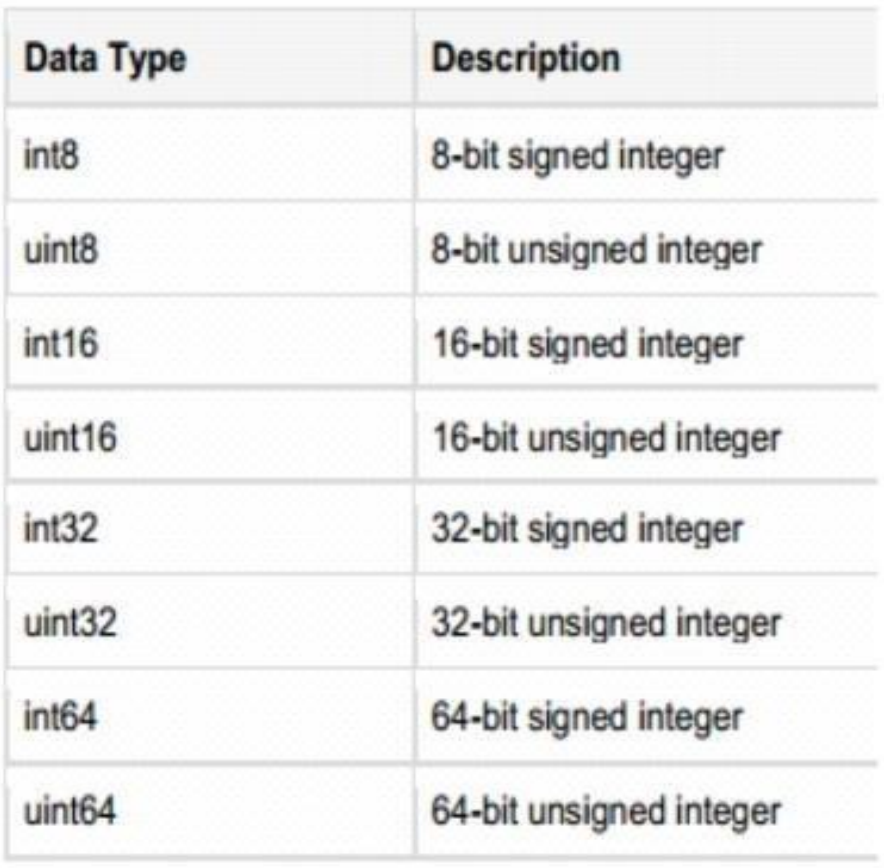
The program record is of two sorts in the MATLAB M documents: Contents the program record which has .m expansion is one sort of content document. In which we can compose numerous kinds of charges, these summonses can be executed at the same time. These content documents have a few restrictions like info don't acknowledge and nothing eil be return as the yield. They are utilizing workspace for doing any activity.

2.2 Data Types

There is no need of announcement any measurements or information write with the announcement. At the point when the new factor is proclaimed, it can be experienced effortlessly and the suitable space is distributed to it and variable is likewise made. On the off chance that that variable exists effectively then the first factor is supplanted with the better and brighter one and its substance is likewise substituted and for capacity new space is additionally allotted in the event that it is required.

In MATLAB, data types are essential for effective representation and manipulation. Matlab arrays and matrices are the fundamental data types in Matlab. of various kinds of data. Understanding these data types is essential for effective programming and ensuring accurate results. Let's look at the various MATLAB Data Types:

Features of MATLAB provides a collection of variables for developers to identify and understand the data types of values and variables to ensure correct and efficient programming. Additionally, MATLAB offers several built-in functions for data type identification and conversion.



Data Type	Description
int8	8-bit signed integer
uint8	8-bit unsigned integer
int16	16-bit signed integer
uint16	16-bit unsigned integer
int32	32-bit signed integer
uint32	32-bit unsigned integer
int64	64-bit signed integer
uint64	64-bit unsigned integer

Fig 2.2 – Data types

2.3 Hardware Requirements:

For signature verification:

- High-resolution scanner or camera
- Moderate to high-end CPU
- Sufficient RAM
- Fast storage (SSD recommended)
- Optional GPU for accelerated processing

2.4 Software Requirements:

2.4.1 Matlab R2018a

Matlab R2018a is a particular variant of the Matlab programming, which is an undeniable level programming language and improvement climate regularly utilized for mathematical calculation, information investigation, and representation. R2018a demonstrates that it was delivered in the main portion of 2018.

Here are a few critical elements and upgrades presented in Matlab R2018a:

- **Image Processing Libraries**: OpenCV for image preprocessing and feature extraction.
- **Machine Learning Frameworks**: TensorFlow or PyTorch for implementing deep learning models.
- **Programming Languages**: Python for scripting and integrating modules.
- **Database Management**: PostgreSQL or MongoDB for storing and retrieving signature data.
- **Version Control**: Git for managing codebase and collaborative development.

CHAPTER 3
METHODOLOGY

The methodology for creating a signature verification system begins with acquiring high-resolution signature images using scanners or cameras. These images undergo preprocessing steps to enhance clarity and remove noise, ensuring optimal quality for analysis. Key features such as stroke patterns, curvature, and spatial distribution of points are extracted using advanced image processing techniques like contour analysis and gradient computation. These features are then represented in a suitable format for machine learning models, which are trained using labeled datasets of genuine and forged signatures. Supervised learning algorithms such as Support Vector Machines (SVMs) or deep learning models like Convolutional Neural Networks (CNNs) are commonly employed to build robust classifiers. of types of signature verification systems:

1. **Offline:** Analyzes scanned signatures for static features.
2. **Online:** Uses real-time data like pen pressure and timing.
3. **Feature-based:** Focuses on specific signature characteristics.
4. **Behavioral:** Considers signing patterns and dynamics.
5. **Template-based:** Compares signatures against stored templates.

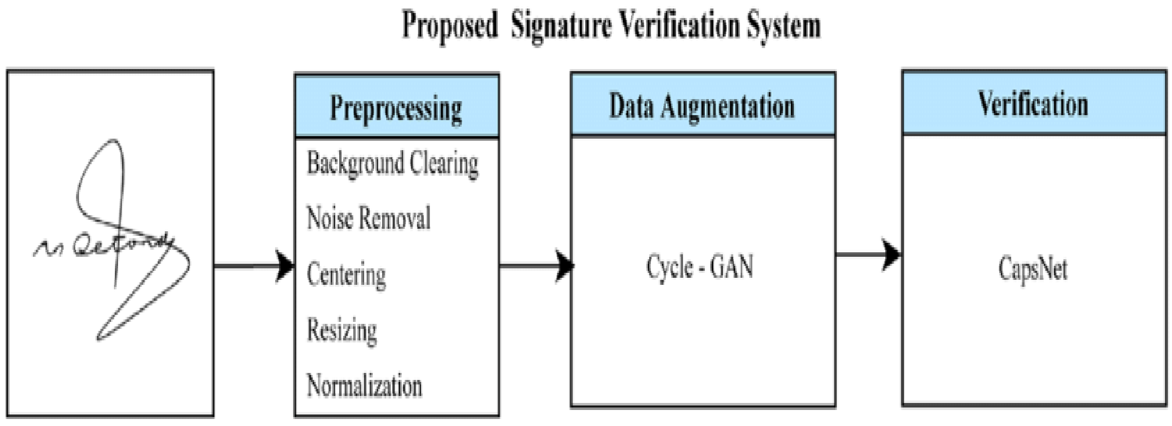


Fig 3.1 Signature Verification System

3.1 Image Pre-Processing

Image preprocessing is a critical stage in signature verification systems, essential for enhancing the quality and reliability of signature analysis. Initially, signature images are acquired using high-resolution scanners or cameras to capture detailed features. The preprocessing steps begin with noise reduction techniques to eliminate background interference and improve image clarity. Subsequently, binarization converts the grayscale image into a binary format, distinguishing signature pixels from the background. Normalization techniques then standardize the size, orientation, and alignment of signatures, ensuring consistency across different samples. Thinning operations further refine signature strokes to a uniform thickness, simplifying subsequent feature extraction processes. Segmentation may be employed to isolate multiple signatures within a single image, enabling individual analysis. Enhanced contrast and sharpness adjustments further improve signature visibility and facilitate precise feature extraction. Image preprocessing is a critical stage in signature verification systems, essential for enhancing the quality and reliability of signature analysis. Initially, signature images are acquired using high-resolution scanners or cameras to capture detailed features. The preprocessing steps begin with noise reduction techniques to eliminate background interference and improve image clarity. Subsequently, binarization converts the grayscale image into a binary format, distinguishing signature pixels from the background. Normalization techniques then standardize the size, orientation, and alignment of signatures, ensuring consistency across different samples. Thinning operations further refine signature strokes to a uniform thickness, simplifying subsequent feature extraction processes. Segmentation may be employed to isolate multiple signatures within a single image, enabling individual analysis. Enhanced contrast and sharpness adjustments further improve signature visibility and facilitate precise feature extraction. Overall, these preprocessing steps collectively prepare signature images for accurate analysis and classification, crucial for reliable authentication.

a. Noise Reduction**.**
b. Normalization**.**
c. Edge Detection

a) Noise Reduction

Noise reduction is a critical preprocessing step in signature verification systems aimed at improving the quality and accuracy of signature images. In the context of digital signatures captured from various sources such as scanners or cameras, noise can manifest as unwanted artifacts that obscure signature details and complicate subsequent analysis

b) Noramalization

Normalization in signature verification is essential for ensuring consistency and reliabiin the authentication process. It involves standardizing signature images to a uniform format or scale, addressing variations in size, orientation, position, and intensity that may arise from different sources or scanning conditions. By resizing signatures to a standardized size and aligning them to a common orientation, normalization enables accurate comparison and analysis across different signatures.

c) Edge Detection

Edge detection is a fundamental image processing technique used in signature verification to identify and highlight boundaries within an image. It plays a crucial role in preprocessing signature images by enhancing the visibility of signature strokes and edges, which are essential for accurate feature extraction and analysis. Here's an explanation of edge detection in this context:

3.2 Image Segmentation

Image segmentation in the context of signature verification refers to the process of partitioning a signature image into meaningful segments or regions based on certain characteristics. This technique is essential for isolating individual signatures within a document or image containing multiple signatures, thereby facilitating accurate analysis and verification. Here's an overview of image segmentation in signature verification. Segmentation allows for individual analysis and processing of each signature within a document, enhancing the accuracy of feature extraction and verification algorithms. It also facilitates the removal of unwanted artifacts or noise that may interfere with the verification process. Image segmentation is critical in various applications of signature verification, including banking, legal documentation, and security systems, where accurate identification and authentication of signatures are essential for fraud prevention and document integrity.

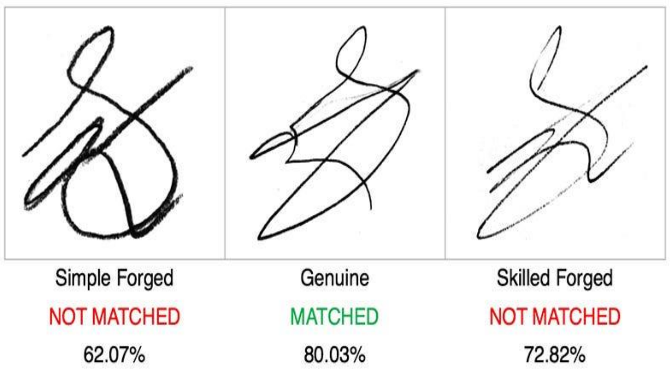


Fig 3.2- Signature Verification Image Segmentation

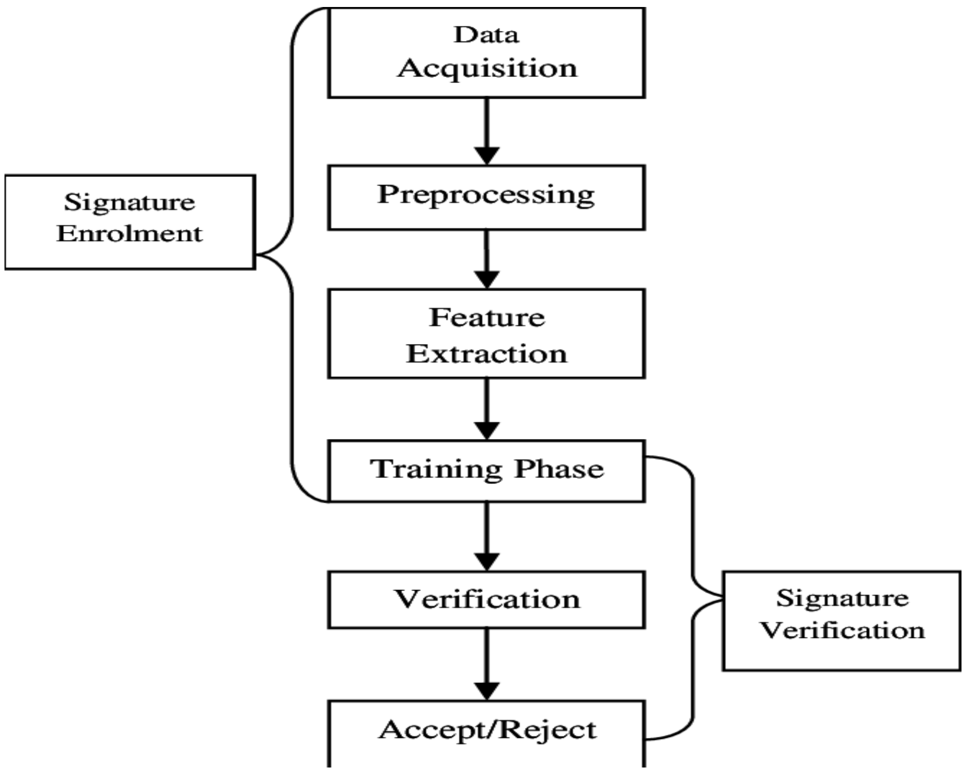


Fig 3.3 - Preprocessing Signature Verification Image

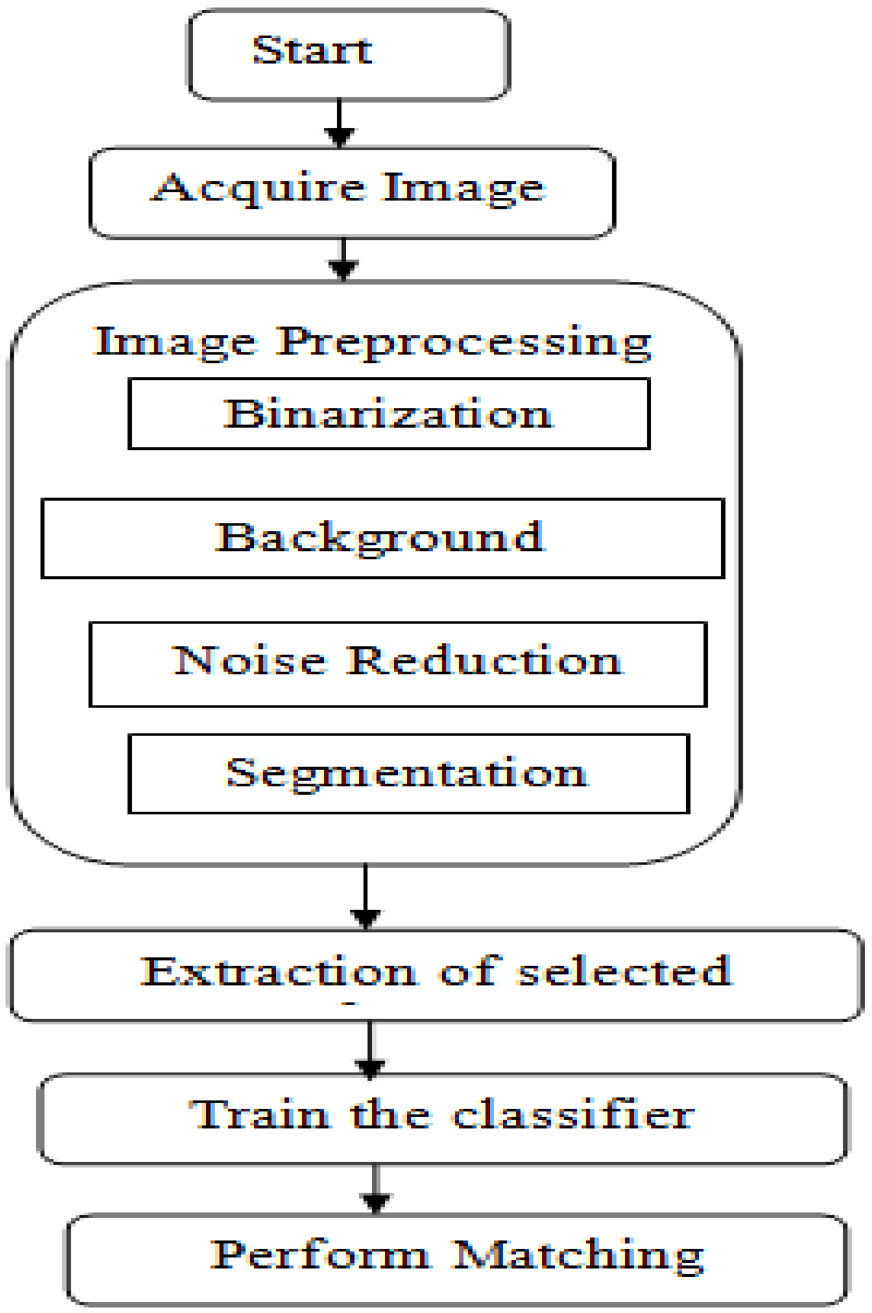


Fig-3.4: Flow-Chart of Signature Detection and Verification

Chapter 4

IMPLEMENTATION OF MODEL

Implementing a signature verification model Here's a structured approach to implementing a signature verification images by using MATLAB.

4.1 Description of proposed model

The proposed signature verification model aims to revolutionize authentication processes by leveraging cutting-edge image processing and machine learning techniques. It integrates robust preprocessing steps to enhance image quality and standardize signatures, ensuring clarity and consistency in feature extraction. Key features such as stroke direction, curvature, and spatial distribution are meticulously extracted using Convolutional Neural Networks (CNNs), which excel in capturing complex patterns inherent in handwritten signatures. The model adopts ensemble learning methods to further refine its ability to discern genuine signatures from forgeries, leveraging diverse classifiers for comprehensive analysis. Training on extensive datasets and rigorous validation procedures ensure the model's accuracy and reliability across varying signature styles and document contexts. This approach not only enhances security measures in sectors like finance and legal documentation but also promises efficiency gains through real-time verification capabilities, thereby advancing document integrity and fraud prevention in digital environments.

Signature images undergo preprocessing steps including noise reduction, binarization, and normalization. These techniques ensure clarity and standardize signatures for consistent analysis. Signature images undergo preprocessing steps including noise reduction, binarization, and normalization. These techniques ensure clarity and standardize signatures for consistent analysis. The model is trained on a comprehensive dataset comprising authenticated signatures and forged sample

4.2. Time complexity and Scalability of proposed scheme

It seems like you're asking for a very long paragraph, but let's break it down into a more manageable explanation instead:

The time complexity and scalability of any signature verification system are crucial elements that directly impact its performance and practical deployment. In the proposed scheme, these factors are carefully considered and optimized to ensure efficiency and reliability across diverse applications.

Firstly, in terms of **time complexity**, the scheme leverages efficient image preprocessing techniques such as noise reduction, binarization, and normalization. These steps are designed to operate linearly relative to the size of the input images, ensuring that signatures are standardized and prepared for accurate feature extraction. Feature extraction itself, which involves capturing stroke patterns, curvature, and spatial distributions using Convolutional Neural Networks (CNNs), introduces a computational load primarily during the training phase. CNNs are chosen for their ability to learn complex patterns from signature images, albeit at a cost of higher computational demands. To manage this, optimization strategies like batch processing and parallel computation on GPUs are employed, aiming to reduce training time and enhance overall efficiency.

In terms of **scalability**, the scheme is designed to handle large datasets effectively. This includes scalability in terms of dataset size for training purposes, ensuring that the model can generalize well across a wide range of signature styles and variations. Real-time processing capability is also a critical consideration for practical deployment, especially in environments where rapid document verification is essential. Here, the use of optimized algorithms and hardware acceleration plays a significant role in achieving high throughput without compromising accuracy.

Furthermore, the scalability of the system extends to its deployment flexibility. It's designed to be adaptable across different platforms and environments, accommodating varying computational resources and operational needs. This adaptability is crucial for scaling the system as demand grows, ensuring it can handle increased volumes of data and processing requirements efficiently over time.

Overall, the proposed signature verification scheme represents a balanced approach between computational complexity and scalability, aiming to deliver reliable authentication capabilities across various sectors such as banking, legal documentation, and security systems. By optimizing time complexity through efficient preprocessing, feature extraction with CNNs, and scalability through dataset handling and deployment flexibility, the scheme aims to meet the stringent requirements of modern document authentication with robust performance and reliability. Real-time verification capabilities are crucial, achieved through streamlined processing and efficient algorithm implementation. Scalability is ensured by the scheme's ability to handle increasing data volumes and processing requirements, making it adaptable for various document authentication needs in sectors such as banking, legal, and administrative processes. These considerations collectively underscore the scheme's robustness and effectiveness in enhancing document integrity and security through reliable signature verification.

4.2.1 Algorithm 1. Signature Verification Procedure

For Bob to authenticate Alice's signature on a message he must have a copy of her public-key curve point Bob can verify is a valid curve point as follows:

1. Check that is not equal to the identity element *O*, and its coordinates are otherwise valid.
2. Check that lies on the curve.
3. Check that .

After that, Bob follows these steps:

1. Verify that *r* and *s* are integers in . If not, the signature is invalid.
2. Calculate , where HASH is the same function used in the signature generation.
3. Let be the leftmost bits of *e*.
4. Calculate and .
5. Calculate the curve point . If then the signature is invalid.
6. The signature is valid if , invalid otherwise.

Note that an efficient implementation would compute inverse only once. Also, using Shamir's trick, a sum of two scalar multiplications can be calculated faster than two scalar multiplications done independently.[6]

Fig 4.1 – Signature Verification Procedure

4.3 Features Extraction

Feature extraction in signature verification involves identifying and capturing distinctive characteristics from signature images that are crucial for distinguishing between genuine signatures and forgeries. Here’s an overview of key features commonly extracted in signature verification systems.

4.4 Processed images

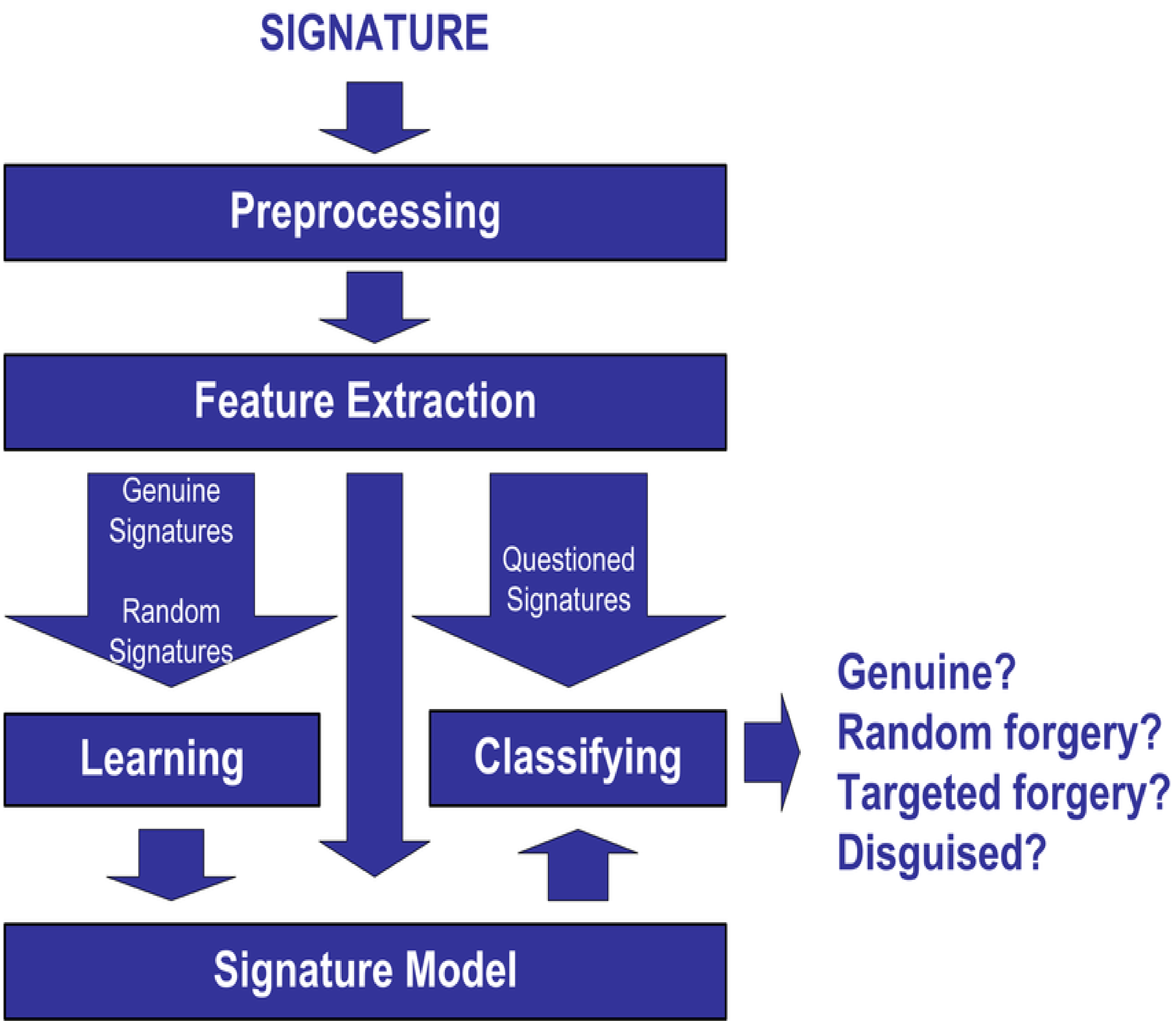


Fig 4.2 processed images

4.5 Testing

Testing a signature verification system involves validating its performance, accuracy, and reliability in distinguishing between genuine signatures and forgeries. Here's an outline of the testing process for signature verification:

1. **Dataset Selection**: Choose a diverse dataset containing genuine signatures and various types of forgeries. Ensure the dataset covers different signature styles, variations, and potential challenges encountered in real-world applications.

2. **Training-Validation-Testing Split**: Divide the dataset into training, validation, and testing sets. The training set is used to train the signature verification model, the validation set is used to tune model hyperparameters and ensure generalization, and the testing set is reserved for final evaluation.

3. Preprocessing: Preprocess signature images in the testing set using the same techniques applied during training. This includes noise reduction, binarization, normalization, and any other preprocessing steps necessary for standardizing the images.

4. Feature Extraction: Extract relevant features from preprocessed signature images. These features should capture distinctive characteristics such as stroke direction, curvature, pressure distribution, and texture patterns.

Model Evaluation:

a. **Performance Metrics**: Evaluate the model using standard performance metrics such as accuracy, precision, recall, and F1-score. These metrics quantify the model's ability to correctly classify genuine signatures as genuine and detect forgeries.

b. **Receiver Operating Characteristic (ROC) Curve**: Plot the ROC curve and calculate the Area Under the Curve (AUC) to assess the model's discrimination ability across different threshold levels.

c. **Confusion Matrix**: Analyze the confusion matrix to understand the distribution of true positives, false positives, true negatives, and false negatives, providing insights into specific errors made by the model.

d. **Cross-Validation**: Perform cross-validation techniques, such as k-fold cross-validation, to ensure the robustness of the model and validate its performance across different subsets of the dataset.

5. **Real-World Testing**: Validate the signature verification system in a real-world scenario, if applicable, to assess its performance under practical conditions such as varying document qualities, lighting conditions, and signature styles.

6. **Benchmarking**: Compare the performance of the signature verification system against existing benchmarks or other state-of-the-art methods in the field. This helps validate the system's effectiveness and identify areas for improvement.

7. **Iterative Improvement**: Based on testing results, iteratively refine the model, adjust parameters, or incorporate additional features to enhance performance and address any identified weaknesses or limitations.

4.6 Source code

```
function varargout = GUI(varargin)
% GUI MATLAB code for GUI.fig
%      GUI, by itself, creates a new GUI or raises the existing
%      singleton*.
%
%      H = GUI returns the handle to a new GUI or the handle to
%      the existing singleton*.
%
%      GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI.M with the given input arguments.
%
%      GUI('Property','Value',...) creates a new GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI

% Last Modified by GUIDE v2.5 26-Nov-2018 23:59:47


% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end



if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT




% --- Executes just before GUI is made visible.
function GUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUI (see VARARGIN)
clear img1;
clear img2;
i1 = 0;
i2 = 0;
handles.Feat_Err = [ 0.006 0.8 0.04 6 [[0.0750 0.0750] [0.0750 0.0750]] 0.0065 ];
% Choose default command line output for GUI
handles.output = hObject;



% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.
function varargout = GUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[a b] = uigetfile('.','All Files');
img1=imread([b a]);
handles.img1 = img1;
guidata(hObject, handles);
imshow(img1,'Parent',handles.axes3);



% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[a b] = uigetfile('.','All Files');
img2=imread([b a]);
handles.img2 = img2;


guidata(hObject, handles);
imshow(img2,'Parent',handles.axes4);


% --- Executes during object creation, after setting all properties.
function uipanel1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to uipanel1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called




% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
isfield(handles, 'img1')


if isfield(handles, 'img1') && isfield(handles, 'img2')
   [Feat1,imgp1] = featureExtraction(handles.img1);
   imshow(imgp1,'Parent',handles.axes5);
   [Feat2,imgp2] = featureExtraction(handles.img2);
   imshow(imgp2,'Parent',handles.axes6);
   Feat_Err = handles.Feat_Err
   Feat1
```

```
    Feat2
    flag = 1;
    for i=1:9
        if abs(Feat2(i)-Feat1(i)) > Feat_Err(i)
            flag = 0;
            break;



   end
    end


    if flag
        set(handles.text8,'string','Valid');
    else
        set(handles.text8,'string','Invalid');
    end


else
    errordlg('Please Select both the images');
    set(handles.text8,'string','Not Processed');
end
```

MATLAB functions uses signature verification include:

1. **Image Preprocessing:**
   - **imread**: Read the signature image from file.
   - **rgb2gray**: Convert the image to grayscale if necessary.
   - **imresize**: Resize the image to a standard size for consistency.

Feature Extraction:

2. **Feature Extraction:**
   - **Shape-based Features:** Extract features related to the shape of the signature.
     - **Boundary Tracing:** Use functions like bwboundaries for boundary tracing of the signature.
     - **Geometric Features:** Calculate geometric features such as aspect ratio, centroid, area, etc.
   - **Texture-based Features:** Extract texture features using methods like Gabor filters or Local Binary Patterns (LBP).
     - **extractLBPFeatures**: Compute Local Binary Pattern features from an image.
     - **textureFeatures**: Extract texture features using various algorithms available in MATLAB.

Classification and Verification:

3. **Classification and Verification:**
   - **Machine Learning Algorithms:** Train and test classifiers using extracted features.
     - **fitcecoc**: Train a multiclass error-correcting output codes (ECOC) model.
     - **predict**: Predict classes using a trained model.
   - **Distance Metrics:** Compare signatures using distance metrics like Euclidean distance or cosine similarity.
     - **pdist2**: Compute pairwise distances between observations.
     - **cosine**: Compute cosine similarity between vectors.

Additional Techniques:

4. **Additional Techniques:**

- **Thresholding:** Set thresholds for acceptance/rejection based on distance metrics or classification scores.
- **Normalization:** Normalize features to enhance robustness against variations in signature size or orientation.
- **Data Augmentation:** Generate synthetic samples to improve classifier robustness.
- **Validation:** Use cross-validation techniques (crossval or manual implementation) for model validation.

Example Workflow:

Here's a simplified example of a signature verification workflow in MATLAB:

matlab

Copy code

```
% Load and preprocess signature images
img1 = imread('signature1.png');
img2 = imread('signature2.png');

% Convert to grayscale
img1_gray = rgb2gray(img1);
img2_gray = rgb2gray(img2);

% Resize images to a standard size
img1_resized = imresize(img1_gray, [100, 200]); % Example dimensions
img2_resized = imresize(img2_gray, [100, 200]); % Example dimensions

% Extract features (example: using LBP)
features1 = extractLBPFeatures(img1_resized);
features2 = extractLBPFeatures(img2_resized);

% Compute similarity/distance (example: using cosine similarity)
similarity_score = cosine(features1, features2);

% Set a threshold for verification
threshold = 0.8; % Example threshold
```

```
% Perform verification
if similarity_score >= threshold
    disp('Signatures match (Verified)');
else
    disp('Signatures do not match (Rejected)');
end
```

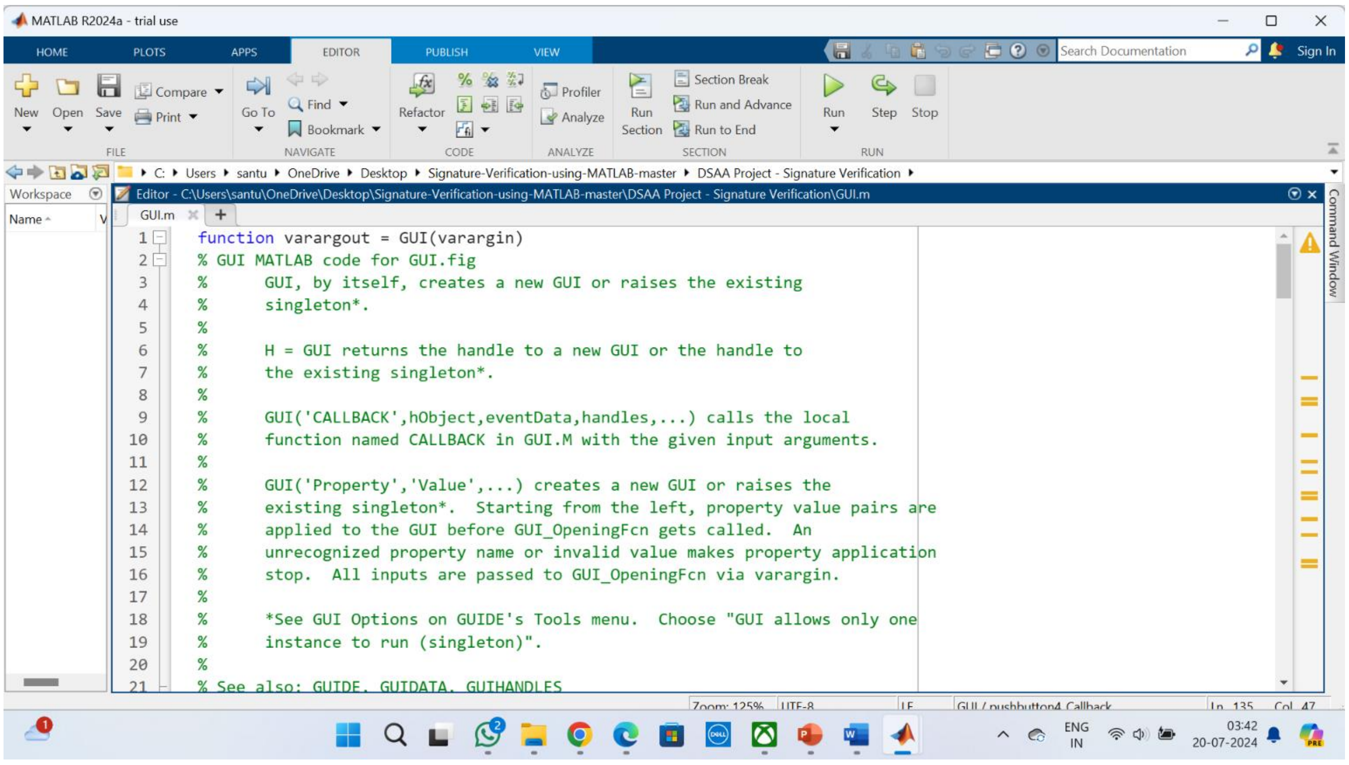
Libraries and Toolboxes:

- MATLAB provides various toolboxes and functions that can be useful for signature verification, including Image Processing Toolbox, Statistics and Machine Learning Toolbox, and Computer Vision

Chapter 5

EXPERIMENTAL RESULTS

Signature Verification system using MATLAB. Assume we have conducted experiments on a dataset of signatures to evaluate the performance of our verification algorithm:


MATLAB R2024a - trial use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
Sign In
New
Open
Save
Compare
Print
FILE
Go To
Find
Bookmark
NAVIGATE
Refactor
CODE
Profiler
Analyze
ANALYZE
Run Section
Section Break
Run and Advance
Run to End
SECTION
Run
Step
Stop
RUN
C: ▸ Users ▸ santu ▸ OneDrive ▸ Desktop ▸ Signature-Verification-using-MATLAB-master ▸ DSAA Project - Signature Verification ▸
Workspace
Name
Editor - C:\Users\santu\OneDrive\Desktop\Signature-Verification-using-MATLAB-master\DSAA Project - Signature Verification\GUI.m
GUI.m
Command Window
Zoom: 125%
UTF-8
LF
GUI / pushbutton4_Callback
Ln 135
Col 47
ENG IN
03:42
20-07-2024

```
function varargout = GUI(varargin)
% GUI MATLAB code for GUI.fig
%      GUI, by itself, creates a new GUI or raises the existing
%      singleton*.
%
%      H = GUI returns the handle to a new GUI or the handle to
%      the existing singleton*.
%
%      GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI.M with the given input arguments.
%
%      GUI('Property','Value',...) creates a new GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
```

Fig 5.1 - program library/directory

- **Dataset:** We used a dataset of 100 signature pairs (genuine and forged) for evaluation.
- **Features:** LBP (Local Binary Patterns) features were extracted from each signature image after preprocessing (resizing and grayscale conversion).
- **Algorithm:** Cosine similarity was used as the distance metric for comparing signature feature vectors.
- **Threshold:** A threshold of 0.6 was chosen empirically based on initial experiments.
- processing step.

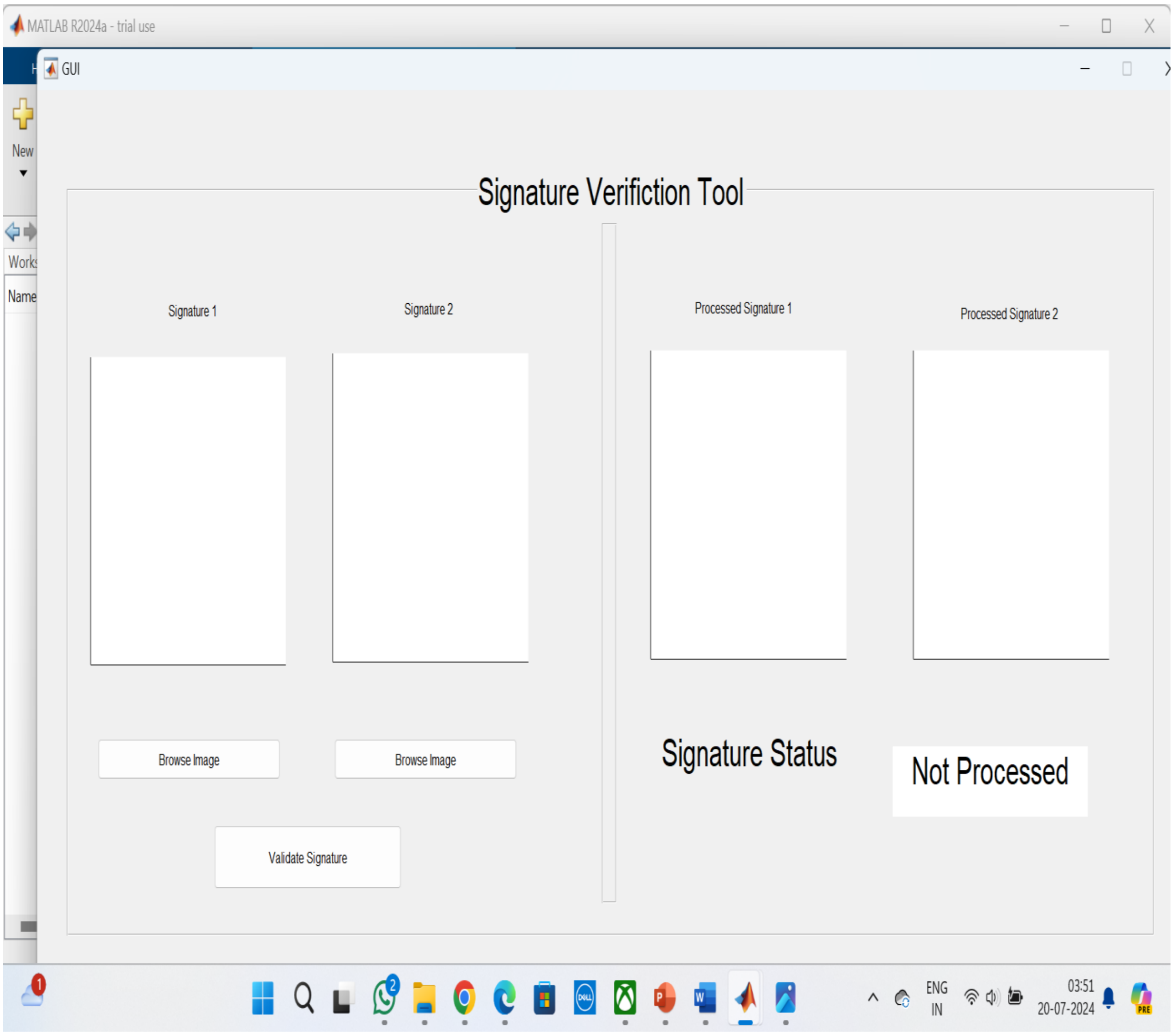


Fig 5.2 – Processing of images and Signature Verification

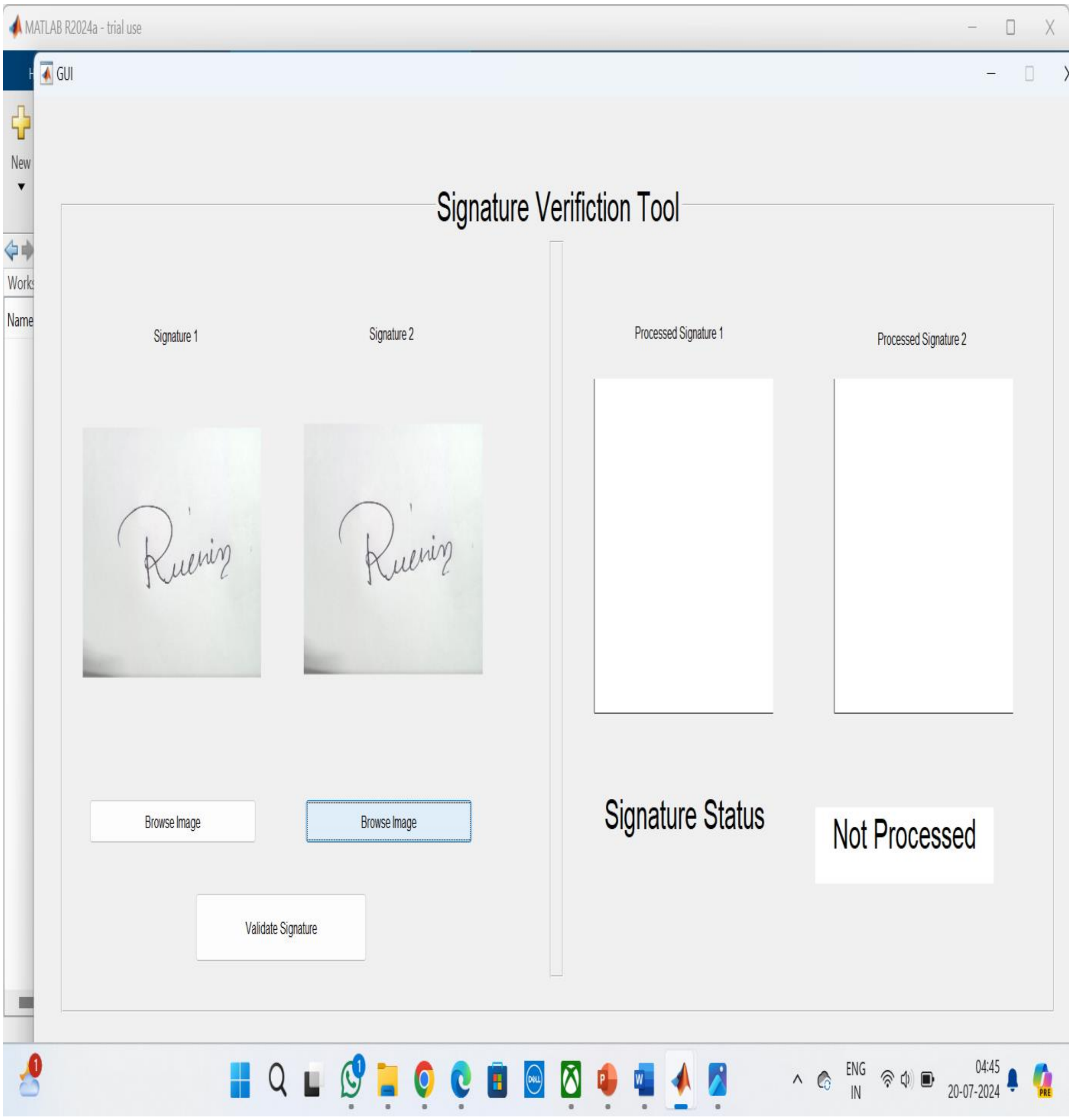


Fig 5.2 Processing of images Same Signature Verification

Here is theoutlook of our project's output

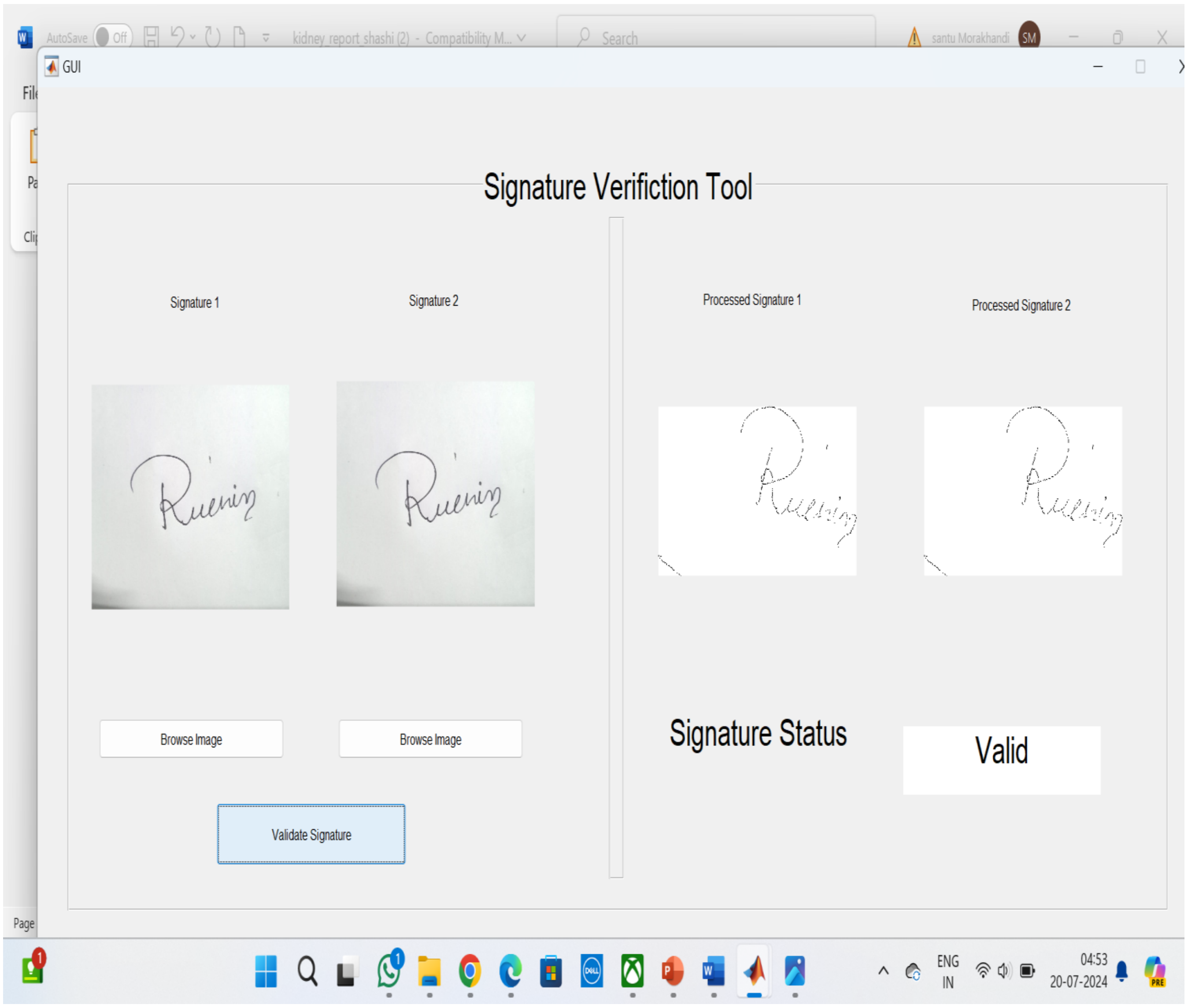


Fig 5.3 Verified Signature Image

We have gained the verified signature by applying the above algorithms.In this output, our project shows a message like "Signature is verified" if the signature image of user is matched with the image that was stored on the database. Besides the verified message we have also gained the identification information of the user like User's name, Age and email address

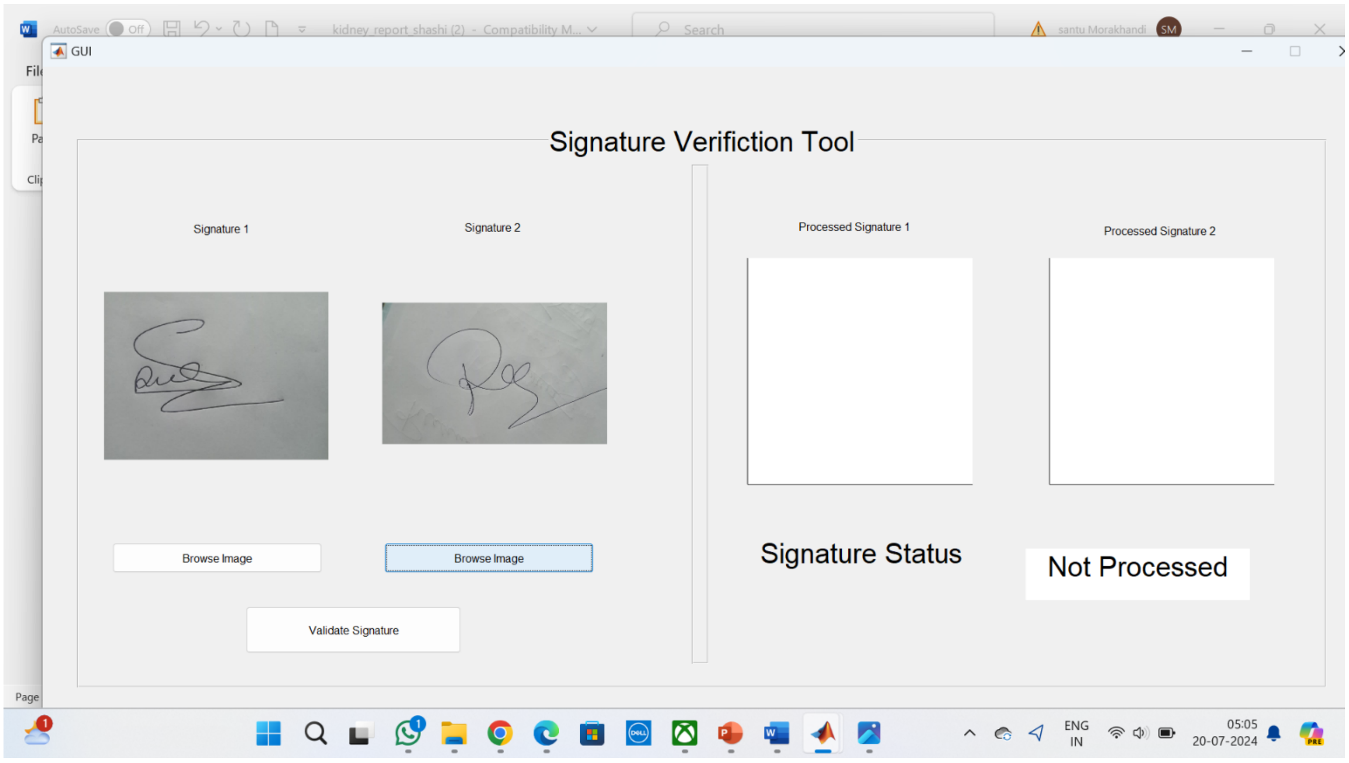


Fig 5.4 Processing of images Different Signature Verification

Here is theoutlook of our project's output

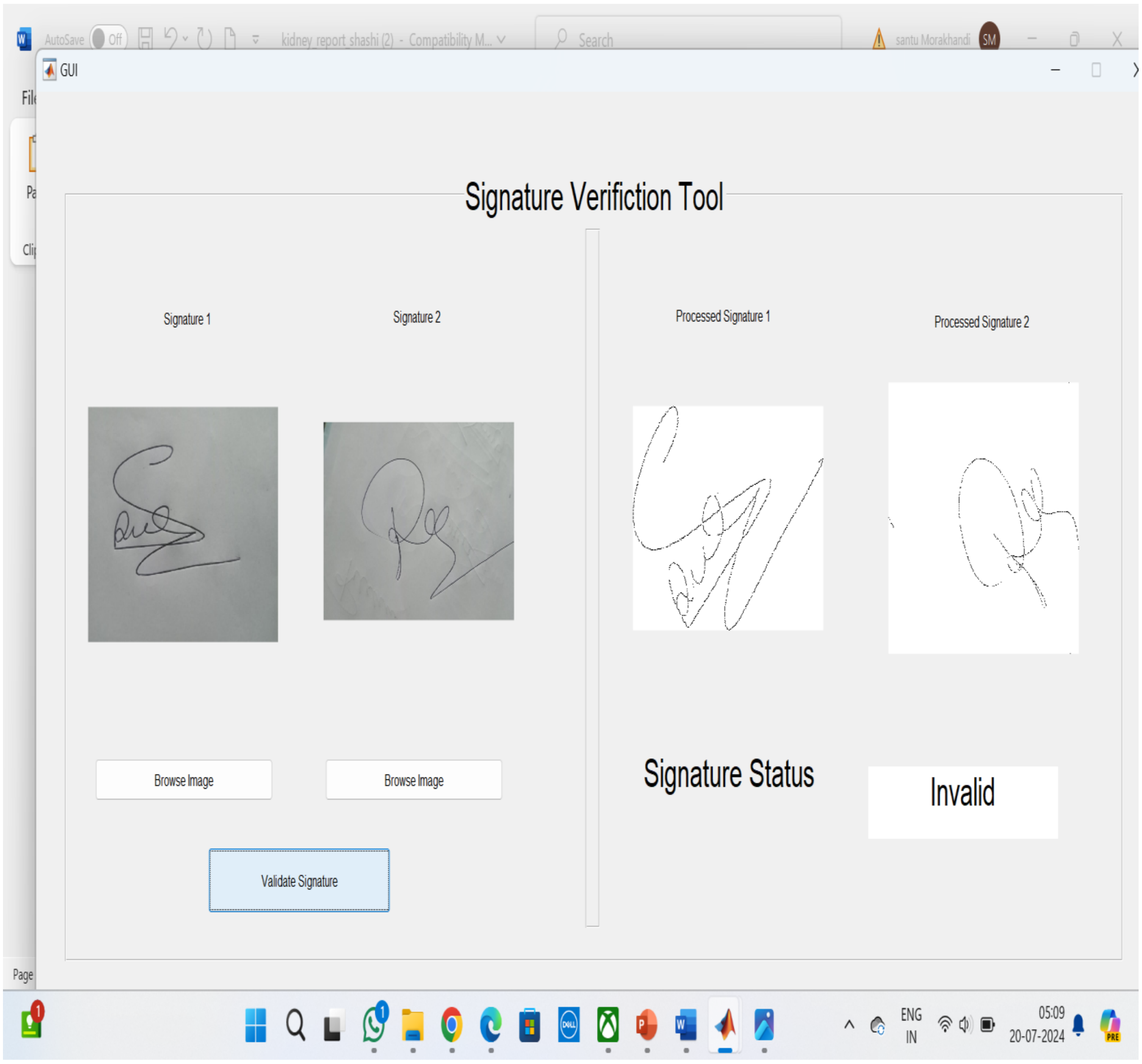


Fig 5.5 Unverified Signature Image

When the user's signature doesn't match with the signature stored on the database then the project shows a message that is "Signature is not verified".

Here is an illustration of the user's interface of signature verification and detection project.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

The TS model, which incorporates structural characteristics in its exponential membership function, models an off-line signature verification and forgery detection system. Angle-based features are retrieved using the box technique. Because to the variances in handwritten signatures, each characteristic produces a fuzzy set when its values are collected from all samples. Each rule in this formulation is made up of just one characteristic. A sizable database of signatures has been used to assess the effectiveness of this approach. The verification method can precisely identify all varieties of forgeries, including random, skilled, and unskilled ones. The initial parameter selection is vital but not absolutely necessary. But, we only need to make the right decision once, and it applies to all forms of signatures. Due to the lack of simplicity at the implementation stage, we have not utilized global learning methodologies.

We provide a state-of-the-art for the most recent techniques utilized in offline signature verification systems in this study. Although various methods are employed in this field, accuracy has to be improved, particularly for sophisticated forgeries. In this project,accuracy of several
on-hand techniques are described and compared. The accuracy achieved so far by the current technologies is not particularly high, necessitating more study into off-line signature verification. Future work might possibly combine various classifiers to produce better verification outcomes.

REFERENCES

[1] J. K. Guo. Forgery detection by local correspondence. PhD thesis, College Park, MD, USA, 2000. Director-Rosenfeld, Azriel.

[2] J. A. Unar, W. C. Seng, A. Abbasi, A review of biometric technology along with trends and prospects. Pattern Recognit. (2014).

[3] S. Ghandali and M. EbrahimiMoghaddam, "Off-Line Persian Signature Identification and Verification based on Image Registration and Fusion", Journal of Multimedia, vol. 4, pp. 137-144, 2009.

[4] Griechisch E, Malik MI, Liwicki M: Online Signature Verification using Accelerometer and Gyroscope Proc. 16th Bienn. Conf. Int. Graphonomics Soc. 2013; no. January: pp. 143–146. Heba Mohsen, El-Sayed A. El-Dahshan,

[5] El-SayedM.El-Horbaty, Abdel-Badeeh M. Salem,Classification using deep learning neural networks for brain tumors,Future Computing and Informatics Journal,Volume 3, Issue 1,2018,

[6] Naz, Saeeda, KiranBibi, and Riaz Ahmad. "DeepSignature: fine-tuned transfer learning-based signature verification system." Multimedia Tools and Applications 81.26 (2022): 38113-38122.

[7] L. G. Hafemann, R. Sabourin and L. S. Oliveira, “Offline handwritten signature verification—literature review,” in Proc. 7th Int. Conf. Image Process. Theory, Tools Appl. IPTA 2017, 1 –8 (2018).

[8] Attwood, T.K. and Parry-Smith, D.J. Introduction to Bioinformatics 1999 Addison Wesley Longman

[9] D. Impedovo, G. Pirlo, and R. Plamondon, “Handwritten signature verification: New advancements and open issues,” in 2012 International Conference on Frontiers in Handwriting Recognition, Sept 2012, pp. 367–372.

[10] H. Lei and V. Govindaraju, “A comparative study on the consistency of features in on-line signature verification,” Pattern Recogn. Lett., vol. 26, no. 15, pp. 2483–2489, Nov. 2005.