FAKE CURRENCY DETECTION USING MATLAB

M.Vijay Kumar - (18261A04F5)

M.Chandrika - (18261A04F6)

M.V.N.Umesh - (18261A04F7)



Department of Electronics and Communication Engineering

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad, T.S.)

Chaitanya Bharathi P.O., Gandipet, Hyderabad- 500075

2022

FAKE CURRENCY DETECTION USING MATLAB

MINI PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT

OF THE RECQUIREMENTS FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

BY

M VIJAY KUMAR - 18261A04F5

M CHANDRIKA - 18261A04F6

M.V.N UMESH - 18261A04F7



Department of Electronics and Communication Engineering

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad, T.S.)

Chaitanya Bharathi P.O., Gandipet, Hyderabad- 500075

2022

**MAHATMA GANDHI INSTITUTE OF TECHNOLOGY**

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad, T.S.)   
  
Chaitanya Bharathi P.O., Gandipet , Hyderabad- 500075



**Department of Electronics and Communication Engineering**

# CERTIFICATE

Date: February 2022

This is to certify that the Mini project entitled “**Fake Currency Detection using MATLAB** ” is a bonafide work carried out by

**M.Vijay Kumar (18261A04F5)**

**M.Chandrika (18261A04F6)**

**M.V.N.Umesh (18261A04F7)**

in partial fulfillment of the requirements for the degree of **BACHELOR OF TECHNOLOGY** in **ELECTRONICS AND COMMUNICATION ENGINEERING** by the Jawaharlal Nehru Technological University, Hyderabad during the academic year 2021-22.

The results embodied in this report have not been submitted to any other University or Institution for the award of any degree or diploma.

|  |  |
| --- | --- |
| (Signature)  Dr.G.Madhavi  Assistant Professor | (Signature)  Dr. S P Singh  Head of Department |

**MAHATMA GANDHI INSTITUTE OF TECHNOLOGY**

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad, T.S.)

Chaitanya Bharathi P.O., Gandipet , Hyderabad- 500075



**Department of Electronics and Communication Engineering**

# DECLARATION

Date: February 2022

We do declare that the thesis work entitled “**FAKE CURRENCY DETECTION USING MATLAB**” submitted to the department of **Electronics and Communication Engineering (ECE)**, **Mahatma Gandhi Institute of Technology**, Hyderabad, in partial fulfilment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY** in **Electronics and Communication Engineering (ECE)** is a bonafide record of the mini project report carried out under the supervision of **Dr. M.FAYAZUR RAHAMAN**, Assistant Professor,

**Mr.D .V.S.NAGENDRA KUMAR**, Assistant Professor, **Dr. D. VENKAT REDDY,** Senior Professor, MGIT.

Also, we declare that the matter embodied in this thesis has not been submitted by me in full or in any part thereof for the award of any degree/diploma of any other Institution or university previously.

**PLACE**: Hyderabad

**DATE**: /02/2022

# ACKNOWLEDGEMENT

We wish to express our sincere thanks to **Dr. S. P. Singh**, Head of the Department of Electronics and Communication Engineering, M.G.I.T., for permitting us to pursue our Project and encouraging us throughout the Project.

We are highly indebted to our Faculty **Dr.M.Fayazur Rahaman**, Senior Assistant Professor,

**Dr. D.V.S.Nagendra Kumar**, Senior Assistant Professor, **Dr.D. Venkat Reddy**, Assistant Professor, Electronics and Communication Engineering Department, who has given us all the necessary technical guidance in carrying out this Project.

We express our deep sense of gratitude to our Guide, **Dr.G.Madhavi**, Assistant Professor, for her valuable guidance and encouragement in carrying out our Project.

Finally, we thank all the people who have directly or indirectly help us through the course of our Project.

M.VIJAY KUMAR  
M.CHANDRIKA

M.V.N.UMESH

# ABSTRACT

Counterfeit notes are one of the biggest problems occurring in cash transactions. For country like India, it is becoming big hurdle. Because of the advances in printing, scanning technologies it is easily possible for a person to print fake notes with use of latest hardware tools. Detecting fake notes manually becomes time-consuming and untidy process hence there is need of automation techniques with which currency recognition process can be efficiently done.

The advancement of color printing technology has increased the rate of fake currency note printing and duplicating the notes on a very large scale. Few years back, the printing could be done in a print house, but now anyone can print a currency note with maximum accuracy using a simple laser printer. As a result the issue of fake notes instead of the genuine ones has been increased very largely. India has been unfortunately cursed with the problems like corruption and black money . And counterfeit of currency notes is also a big problem to it. This leads to design of a system that detects the fake currency note in a less time and in a more efficient manner. The proposed system gives an approach to verify the Indian currency notes. Verification of currency note is done by the concepts of image processing. This article describes extraction of various features of Indian currency notes. MATLAB software is used to extract the features of the note. The proposed system has got advantages like simplicity and high performance speed. The result will predict whether the currency note is fake or not.

**Keywords:** MATLAB, Feature extraction, Analysis, Detection

# TABLE OF CONTENTS

[CERTIFICATE iii](#_Toc94796008)

[DECLARATION iv](#_Toc94796009)

[ACKNOWLEDGEMENT v](#_Toc94796010)

[ABSTRACT vi](#_Toc94796011)

[TABLE OF CONTENTS vii](#_Toc94796012)

[LIST OF FIGURES ix](#_Toc94796013)

[CHAPTER 01 INTRODUCTION 1](#_Toc94796014)

[CHAPTER 02 SOFTWARE DESCRIPTION 4](#_Toc94796015)

[2.1 Introduction 4](#_Toc94796016)

[2.2 Typical uses 4](#_Toc94796017)

[2.3 Starting and Quitting MATLAB 6](#_Toc94796018)

[2.4 The M-file 8](#_Toc94796019)

[2.5 Images: 10](#_Toc94796020)

[CHAPTER 03 PROPOSED SYSTEM 16](#_Toc94796021)

[CHAPTER 04 FEATURES OF REAL NOTE 18](#_Toc94796022)

[CHAPTER 05 METHODOLOGY AND IMPLEMENTATION 20](#_Toc94796023)

[5.1 Methodology 20](#_Toc94796026)

[5.2 Implementation 22](#_Toc94796027)

[CHAPTER 06 APPLICATIONS AND ADVANTAGES 23](#_Toc94796029)

[6.1 Applications 23](#_Toc94796031)

[6.2 Advantages 23](#_Toc94796032)

[CHAPTER 07 SOURCE CODE 24](#_Toc94796033)

[7.1 Main.m 24](#_Toc94796040)

[7.2 TestImage.m 27](#_Toc94796041)

[7.3 Subtest.m 33](#_Toc94796042)

[7.4 Call.m 41](#_Toc94796043)

[7.5 Compare.m 43](#_Toc94796044)

[7.6 Features.m 43](#_Toc94796045)

[7.7 Edgesdetection.m 44](#_Toc94796046)

[CHAPTER 08 RESULTS 45](#_Toc94796047)

[CHAPTER 09 CONCLUSION 48](#_Toc94796048)

[REFERENCES 49](#_Toc94796049)

# LIST OF FIGURES

[Figure 1.1 2000 rupees note with its various real identification mark 2](file:///C:\Users\User\Desktop\We%20see%20you\MINI\Fake%20Currency%20Detection%20(F5,F6,F7)\MINIreport.docx#_Toc94619889)

[Figure 2.1 MATLAB Window 7](#_Toc94619890)

[Figure 2.2 Example of M file 9](file:///C:\Users\User\Desktop\We%20see%20you\MINI\Fake%20Currency%20Detection%20(F5,F6,F7)\MINIreport.docx#_Toc94619891)

[Figure 2.3 Example of M file 11](file:///C:\Users\User\Desktop\We%20see%20you\MINI\Fake%20Currency%20Detection%20(F5,F6,F7)\MINIreport.docx#_Toc94619892)

[Figure 2.4 Bitmap and Gray Scale Image 12](#_Toc94619893)

[Figure 2.5 JPEG and Grayscale Images 12](#_Toc94619894)

[Figure 2.6 M file for saving Image 13](file:///C:\Users\User\Desktop\We%20see%20you\MINI\Fake%20Currency%20Detection%20(F5,F6,F7)\MINIreport.docx#_Toc94619895)

[Figure 3.1 Flow Chart of Proposed System 17](#_Toc94619896)

[Figure 4.1 Security Features of a note 18](file:///C:\Users\User\Desktop\We%20see%20you\MINI\Fake%20Currency%20Detection%20(F5,F6,F7)\MINIreport.docx#_Toc94619897)

[Figure 5.1 Acquired Image 20](file:///C:\Users\User\Desktop\We%20see%20you\MINI\Fake%20Currency%20Detection%20(F5,F6,F7)\MINIreport.docx#_Toc94619898)

[Figure 5.2 Gray Scale Conversion 20](#_Toc94619899)

[Figure 5.3 Image Segmentation 21](file:///C:\Users\User\Desktop\We%20see%20you\MINI\Fake%20Currency%20Detection%20(F5,F6,F7)\MINIreport.docx#_Toc94619900)

[Figure 5.4 Feature Extraction 21](#_Toc94619901)

[Figure 5.5 Edges Detection 22](#_Toc94619902)

[Figure 8.1 Main.m 45](file:///C:\Users\User\Desktop\We%20see%20you\MINI\Fake%20Currency%20Detection%20(F5,F6,F7)\MINIreport.docx#_Toc94619903)

[Figure 8.2Testimage.m 45](file:///C:\Users\User\Desktop\We%20see%20you\MINI\Fake%20Currency%20Detection%20(F5,F6,F7)\MINIreport.docx#_Toc94619904)

[Figure 8.3subtest.m 46](#_Toc94619905)

[Figure 8.4 Segmented images of scanned and real notes 47](file:///C:\Users\User\Desktop\We%20see%20you\MINI\Fake%20Currency%20Detection%20(F5,F6,F7)\MINIreport.docx#_Toc94619906)

[Figure 8.5 feature extraction using MSER algorithm 47](file:///C:\Users\User\Desktop\We%20see%20you\MINI\Fake%20Currency%20Detection%20(F5,F6,F7)\MINIreport.docx#_Toc94619907)

# CHAPTER 01 INTRODUCTION

The Reserve Bank is only one which has the sole authority to issue bank notes in India. Reserve Bank, like other central banks the world over, changes the design of bank notes from time to time. Traditionally, anticounterfeiting measures involved including fine detail with raised intaglio printing on bills which allows non-experts to easily spot forgeries. On coins, milled or marked with parallel grooves edges are used to show that none of the valuable metal has been scraped off. Reserve bank uses several techniques to detect fake currency. Manual testing of all notes in transactions is very time consuming and untidy process and also there is a chance of tearing while handing notes. Therefore, Automatic methods for bank note recognition are required in many applications such as automatic selling goods and vending machines. Extracting sufficient monetary characteristics from the currency image is essential for accuracy and robustness of the automated system.

This is a challenging issue to system designers. Every year RBI (Reserve bank of India) face the counterfeit currency notes or destroyed notes. Handling of large volume of counterfeit notes imposes additional problems. Therefore, involving machines (independently or as assistance to the human experts) makes notes recognition process simpler and efficient. Counterfeit money is imitation currency produced without the legal sanction of the state or government. Producing or using counterfeit money is a form of fraud or forgery. Counterfeiting is almost as old as money itself. Plated copies have been found of Lydian coins which are thought to be among the first western coins. Before the introduction of paper money, the most prevalent method of counterfeiting involved mixing base metals with pure gold or silver. A form of counterfeiting is the production of documents by legitimate printers in response to fraudulent instructions. Counterfeit money is imitation currency produced without the legal sanction of the state or government. Producing or using this fake money is a form of fraud or forgery. Counterfeiting is as old as money itself, and is sufficiently prevalent throughout history that it has been called "the world's second oldest profession.

This has led to the increase of corruption in our country hindering country’s growth. Common man became a scapegoat for the fake currency circulation, let us suppose that a common man went to a bank to deposit money in bank but only to see that some of the notes are fake, in this case he has to take the blame. Counterfeiting, of whatever kind, may be that has been occurring ever since humans grasped the concept of valuable items, and there has been an ongoing race between certifier like (banks, for example) and counterfeiter ever since. Some of the effects that counterfeit money has on society include a reduction in the value of real money; and inflation due to more money getting circulated in the society or economy which in turn dampen our economy and growth - an unauthorized artificial increase in the money supply; a decrease in the acceptability of paper money; and losses. And this some of the methods to detect fake currency are water marking, optically variable ink, security thread, latent image, techniques like counterfeit detection pen and using MATLAB.

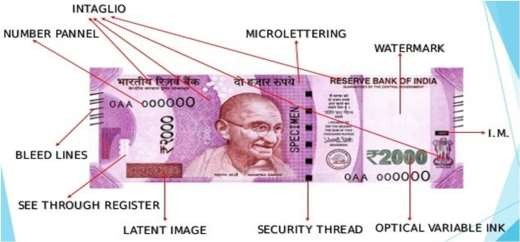


Figure .1 2000 rupees note with its various real identification mark

The currency counting machine or CCM is one of the miracles of the science. The CCM works on the principle on the breadth of the bundle of currency and there in an roller which has rods in an continuous pattern and the roller moves these rods with a particular speed. The speed remains constant as like in the ATM machine counting machine and these rollers moves on the bundle of the currency and just move out the single currency one by one at a constant and high speed and there is a transducer which detect that how many single currencies has passed out in front of it.

Different range of counting machines like Basic Note counter, Intelligent Counting cum counterfeit detection machines and Hi Speed Heavy duty cash counting machine are available to suit different type of customers. Highly dependable and ideal for Banks, Big & small business houses, Traders, retailers, jewelers and almost all types of business establishment can use them according to their suitability.

The machine meant for detection of fake notes as prime function invariably should be capable of not allowing any fake note to pass as genuine. It is possible only with the detectors specially developed considering the large number of intricacies concerning to Indian notes.

The kind of machines Indian Banks at cash counters needed are the machine which can verify not only the images but also can check the chemical and physical properties of papers, inks, resins and other materials used in production of note. The machine should be capable of not allowing any fake note to pass as genuine. It is possible only with the detectors specially developed considering the large number of intricacies concerning to Indian note.

# CHAPTER 02 SOFTWARE DESCRIPTION

## 2.1 Introduction

If you are new to MATLAB, you should start by reading Manipulating Matrices. The most important things to learn are how to enter matrices, how to use the: (colon) operator, and how to invoke functions. After you master the basics, you should read the rest of the sections below and run the demos. At the heart of MATLAB is a new language you must learn before you can fully exploit its power. You can learn the basics of MATLAB quickly, and mastery comes shortly after. You will be rewarded with high productivity, high-creativity computing power that will change the way you work.

Describes the components of the MATLAB system.

Development Environment - introduces the MATLAB development environment, including information about tools and the MATLAB desktop.

Manipulating Matrices - introduces how to use MATLAB to generate Matrices and perform mathematical operations on matrices.

Graphics - introduces MATLAB graphic capabilities, including information about plotting data, annotating graphs, and working with images.

Programming with MATLAB - describes how to use the MATLAB language to create scripts and functions, and manipulate data structures, such as cell arrays and multidimensional arrays. MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.

## 2.2 Typical uses

Math and computation

Algorithm Development   
Modeling, Simulation, and Prototyping  
Data analysis, Exploration, and Visualization Scientific and engineering graphics  
Application development,  
 including graphical user interface building   
MATLAB is an interactive system whose basic data element is an array that does not require dimensioning.

This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or FORTRAN.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

**MATLAB system**

The MATLAB system consists of five main parts:

* + - * Development Environment

This is the set of tools and facilities that help you use MATLAB function and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, and browsers for viewing help, the workspace, files, and the search path.

* + - * The MATLAB Mathematical Function Library

This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

* + - * The MATLAB Language

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "Programming in the large" to create complete large and complex application programs.

* + - * Handle Graphics

This is the MATLAB graphics system. It includes high-level commands for two- dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

* + - * The MATLAB Application Program Interface (API)

This is a library that allows you to write C and FORTRAN programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

* + - * Development environment

This chapter provides a brief introduction to starting and quitting MATLAB, and the tools and functions that help you to work with MATLAB variables and files. For more information about the topics covered here, see the corresponding topics under Development Environment in the MATLAB documentation, which is available online as well as in print.

## Starting and Quitting MATLAB

* + - * Starting MATLAB

On a Microsoft Windows platform, to start MATLAB, double-click the MATLAB shortcut icon on your Windows desktop. On a UNIX platform, to start MATLAB, type MATLAB at the operating system prompt.

After starting MATLAB, the MATLAB desktop opens - see MATLAB Desktop. You can change the directory in which MATLAB starts, define startup options including running a script upon startup, and reduce startup time in some situations.

* + - * Quitting MATLAB

To end your MATLAB session, select Exit MATLAB from the File menu in the desktop, or type quit in the Command Window. To execute specified functions each time MATLAB quits, such as saving the workspace, you can create and run a finish script.

* + - * MATLAB Desktop

When you start MATLAB, the MATLAB desktop appears, containing tools (graphical user interfaces) for managing files, variables, and applications associated with MATLAB.

In this chapter the software analysis was done by giving the flowcharts and their corresponding algorithms.

### Image Processing with MATLAB

The purpose of this tutorial is to gain familiarity with MATLAB’s Image Processing Toolbox. This tutorial does not contain all of the functions available in MATLAB. It is very useful to go to Help\MATLAB window if you have any questions not answered by this tutorial. Many of the examples in this tutorial are modified versions of MATLAB’s help examples. The help tool is especially useful in image processing applications since there are numerous filter examples.

1. Opening MATLAB in the microcomputer lab

Access the Start Menu, Proceed to Programs, Select MATLAB 14 from the MATLAB 14 folder --OR-- Open through C:\MATLAB6p5\bin\win32\matlab.exe

1. MATLAB When MATLAB opens, the screen should look something like what is pictured in below.

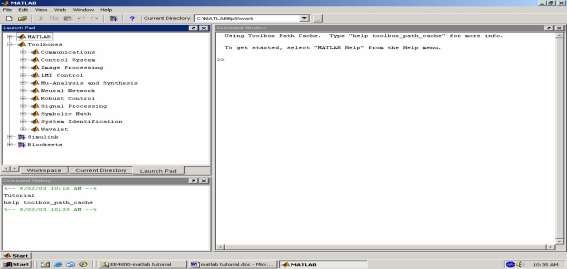


Figure 2.1 MATLAB Window

The Command Window is the window on the right-hand side of the screen. This window is used to both enter commands for MATLAB to execute, and to view the results of these commands. The Command History window, in the lower left side of the screen, displays the commands that have been recently entered into the Command Window. In the upper left hand side of the screen there is a window that can contain three different windows with tabs to select between them. The second window is the Workspace window, which displays which variables are currently being used and how big they are. The third window is the Launch Pad window, which is especially important since it contains easy access to the available toolboxes, of which, Image Processing is one. If these three windows do not all appear as tabs below the window space, simply go to View and select the ones you want to appear.

In order to gain some familiarity with the Command Window, try Example 2.1, below. You must type code after the >> prompt and press return to receive a new prompt. If you write code that you do not want to reappear in the MATLAB Command Window, you

must place a semi colon after the line of code. If there is no semi colon, then the code will print in the command window just under where you typed it.

Example 2.1

X = 1; %press enter to go to next line Y = 1; %press enter to go to next line

Z = X + Y %press enter to receive result

As you probably noticed, MATLAB gave an answer of Z = 2 under the last line of typed code. If there had been a semi colon after the last statement, the answer would not have been printed. Also, notice how the variables you used are listed in the Workspace Window and the commands you entered are listed in the Command History window. If you want to retype a command, an easy way to do this is to press the ↑ or ↓ arrows until you reach the command you want to renter.

## The M-file

M-file – An M-file is a MATLAB document the user creates to store the code they write for their specific application. Creating an M-file is highly recommended, although not entirely necessary. An M-file is useful because it saves the code the user has written for their application. It can be manipulated and tested until it meets the user’s specifications. The advantage of using an M-file is that the user, after modifying their code, must only tell MATLAB to run the M-file, rather than reenter each line of code individually

Saving – The next step is to save the newly created M-file. In the M-file window, select File\Save As….. Opening an M-file – To open the M-file by going to File\Open…, and selecting your file. Then, in order for MATLAB to recognize where your M-file is stored, you must go to File\Set Path… Click the Add Folder… button, then browse to find the folder that your M-file is located in, and press OK. Then in the Set Path window, select Save, and then Close. If you do not set the path, MATLAB may open a window saying your file is not in the current directory. In order to get by this, select the “Add directory to the top of the MATLAB path” button, and hit OK.. Writing Code – After creating and saving your M-file, the next step is to begin writing code.. Comments are declared by placing a % symbol before them. Comments appear in green in the M-file window.

Graphical user interface, text, application, email

Description automatically generated

Figure 2.2 Example of M-file

## Images:

Images – The first step in MATLAB image processing is to understand that a digital image is composed of a two- or three-dimensional matrix of pixels. Individual pixels contain a number or numbers representing what grayscale or color value is assigned to it. Color pictures generally contain three times as much data as grayscale pictures, depending on what color representation scheme is used. Therefore, color pictures take three times as much computational power to process. In this tutorial the method for conversion from color to grayscale will be demonstrated and all processing will be done on grayscale images. However, in order to understand how image processing works, we will begin by analyzing simple two dimensional 8-bit matrices. Loading an Image – Many times you will want to process a specific image, other times you may just want to test a filter on an arbitrary matrix. If you choose to do this in MATLAB you will need to load the image so you can begin processing. If the image that you have is in color, but color is not important for the current application, then you can change the image to grayscale.

This makes processing much simpler since then there are only a third of the pixel values present in the new image. Color may not be important in an image when you are trying to locate a specific object that has good contrast with its surroundings., below, demonstrates how to load different images.In some instances, the image in question is a matrix of pixel values. For example, you may need something to test a filter on, but you do not yet need a real image to test the filter. Therefore, you can simply create a matrix that has the characteristics wanted, such as areas of high and low frequency. See Example4. 6.1, for a demonstration of this. Other times a stored image must be imported into MATLAB to be processed. If color is not an important aspect then rgb2gray can be used to change a color image into a grayscale image.**C:\MATLAB6p5\toolbox\images\imdemos**. Therefore, it is a good idea to know how to load any image from any folder.

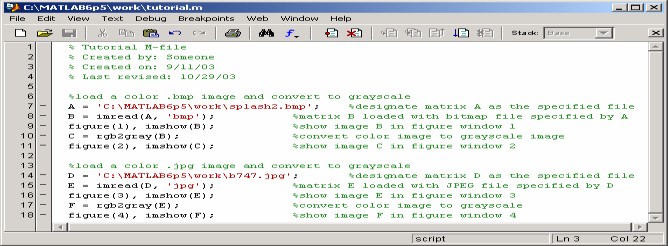


Figure 2.3 M file for loading Images

The class of the new image is the same as that of the color image. As you can see from the example M-file in Figure 4.1, MATLAB has the capability of loading many different image formats, two of which are shown. The function imread is used to read an image file with a specified format. Consult imread in MATLAB’s help to find which formats are supported. The function imshow displays an image, while figure tells MATLAB which figure window the image should appear in. If figure does not have a number associated with it, then figures will appear chronologically as they appear in the M-file. Figures 4.2, 4.3, 4.4 and 4.5, below, are a loaded bitmap file, the image in Figure 4.2 converted to a grayscale image, a loaded JPEG file, and the image in Figure 4.4 converted to a grayscale image, respectively. The images used in this example are both MATLAB example images. In order to demonstrate how to load an image file, these images were copied and pasted into the folder denoted in the M-file in Figure In Example 7.1, later in this tutorial, you will see that MATLAB images can be loaded by simply using the imread function.

A picture containing text

Description automatically generated

Figure 2.4 Bitmap and Gray Scale Image

Writing an Image – Sometimes an image must be saved so that it can be transferred to a disk or opened with another program. In this case you will want to do the opposite of loading an image, reading it, and instead write it to a file.

This can be accomplished in MATLAB using the imwrite function. This function allows you to save an image as any type of file supported by MATLAB, which are the same as supported by imread. below, contains code necessary for writing an image.

In order to save an image, you must use the imwrite function in MATLAB. The M-file in contains code for saving an image. This M-file loads the same bitmap file as described in the M-file pictured in. However, this new M-file saves the grayscale image created as a JPEG image. Just like the “splash2” bitmap picture must be moved into MATLAB’s work folder in order for the imread function to find it.

A picture containing text

Description automatically generated

Figure 2.5 JPEG and Grayscale Images

### Image Properties

* + - * Histogram:

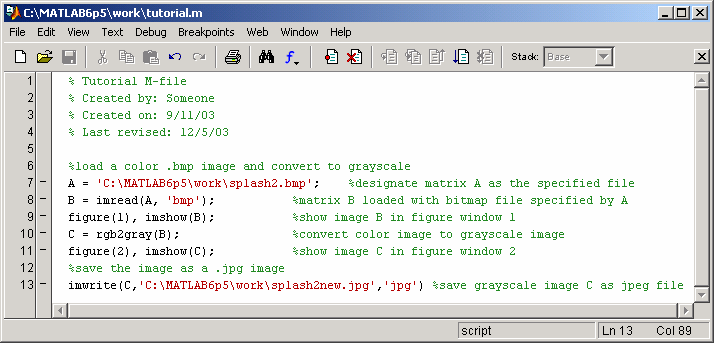
A histogram is bar graph that shows a distribution of data. In image processing histograms are used to show how many of each pixel value are present in an image. Histograms can be very useful in determining which pixel values are important in an image. From this data you can manipulate an image to meet your specifications. Data from a histogram can aid you in contrast enhancement and thresholding. In order to create a histogram from an image, use the imhist function. Contrast enhancement can be performed by the histeq function, while thresholding can be performed by using the gray thresh function and the im2bw function. See Example 5.1, for a demonstration of imhist, imadjust, gray thresh, and im2bw. If you want to see the resulting histogram of a contrast enhanced image, simply perform the imhist operation on the image created with histeq. Negative – The negative of an image means the output image is the reversal of the input image. In the case of an 8-bit image, the pixels with a value of 0 take on a new value of 255, while the pixels with a value of 255 take on a new value of 0. All the pixel values in between take on similarly reversed new values.

Figure 2.6 M file for saving Image

* + - * Frequency Domain

Fourier Transform – In order to understand how different image processing filters work, it is a good idea to begin by understanding what frequency has to do with images. An image is in essence a two-dimensional collection of discrete signals. For instance, if there is relatively little change in grayscale values as you scan across an image, then there is lower frequency content contained within the image. If there is wide variation in grayscale values across an image then there will be more frequency content associated with the image. This may seem somewhat confusing, so let us think about this in terms that are more familiar to us. From signal processing, we know that any signal can be represented by a collection of sine waves of differing frequencies, magnitudes and phases.

This transformation of a signal into its constituent sinusoids is known as the Fourier Transform. This collection of sine waves can potentially be infinite, if the signal is difficult to represent, but is generally truncated at a point where adding more signals does not significantly improve the resolution of the recreation of the original signal. In digital systems, we use a Fourier Transform designed in such a way that we can enter discrete input values, specify our sampling rate, and have the computer generate discrete outputs. This is known as the Discrete Fourier Transform, or DFT.

MATLAB uses a fast algorithm for performing a DFT, which is called the Fast Fourier Transform, or FFT, whose MATLAB command is fft. The FFT can be performed in two dimensions, fft2 in MATLAB. This is very useful in image processing because we can then determine the frequency content of an image. Still confused? Picture an image as a two- dimensional matrix of signals. If you plotted just one row, so that it showed the grayscale value stored within each pixel, you might end up with something that looks like a bar graph, with varying values in each pixel location. However, the Fourier Transform can determine which frequencies are present in the signal.

* + - * Filters

Filters – Image processing is based on filtering the content of images. Filtering is used to modify an image in some way. This could entail blurring, deblurring, locating certain features within an image, etc.… Linear filtering is accomplished using convolution, as discussed above. A filter, or convolution kernel as it is also known, is basically an algorithm for modifying a pixel value, given the original value of the pixel and the values of the pixels surrounding it. There are literally hundreds of types of filters that are used in image processing. However, we will concentrate on several common ones. Low Pass Filters – The first filters we will talk about are low pass filters. These filters blur high frequency areas of images. However, these filters do not discriminate between noise and edges, so they tend to smooth out content that should not be smoothed out.

Median Filters – Median Filters can be very useful for removing noise from images. A median filter is like an averaging filter in some ways. The averaging filter examines the pixel in question and its neighbor’s pixel values and returns the mean of these pixel values. The median filter looks at this same neighborhood of pixels, but returns the median value. In this way noise can be removed, but edges are not blurred as much, since the median filter is better at ignoring large discrepancies in pixel values.

Edge Detectors – Edge detectors are very useful for locating objects within images. There are many different kinds of edge detectors, but we will concentrate on two: the Sobel edge detector and the Canny edge detector. The Sobel edge detector is able to look for strong edges in the horizontal direction, vertical direction, or both directions. The Canny edge detector detects all strong edges plus it will find weak edges that are associated with strong edges. Both of these edge detectors return binary images with the edges shown in white on a black background.

* + - * Segmentation

Segmentation is the process of fractioning an image into its component objects. This can be accomplished in various ways in MATLAB. One way is to use a combination of morphological operations to segment touching objects within an image.

# CHAPTER 03 PROPOSED SYSTEM

We propose Fake Currency Detection, Counterfeit notes are one of the biggest problems occurring in cash transactions. For country like India, it is becoming big hurdle. Because of the advances in printing, scanning technologies it is easily possible for a person to print fake notes with use of latest hardware tools. Detecting fake notes manually becomes time-consuming and untidy process hence there is need of automation techniques with which currency recognition process can be efficiently done.

The advancement of color printing technology has increased the rate of fake currency note printing and duplicating the notes on a very large scale. Few years back, the printing could be done in a print house, but now anyone can print a currency note with maximum accuracy using a simple laser printer. As a result the issue of fake notes instead of the genuine ones has been increased very largely. India has been unfortunately cursed with the problems like corruption and black money . And counterfeit of currency notes is also a big problem to it. This leads to design of a system that detects the fake currency note in a less time and in a more efficient manner.

The proposed system gives an approach to verify the Indian currency notes. Verification of currency note is done by the concepts of image processing. This article describes extraction of various features of Indian currency notes. MATLAB software is used to extract the features of the note. The proposed system has got advantages like simplicity and high performance speed. The result will predict whether the currency note is fake or not.



Result

comparison

Feature extraction

Image segmentation

Edge detection

RGB to Gray conversion

Image acquisition

Figure 3.1 Flow Chart of Proposed System

# CHAPTER 04 FEATURES OF REAL NOTE

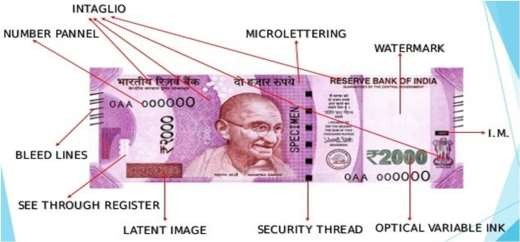
The Mahatma Gandhi Series of banknotes contain the Mahatma Gandhi watermark with a light and shade effect and multi- directional lines in the watermark window.This is a new feature included in the Rs.1000 and Rs.500 notes with revised color scheme introduced in November 2000. The numeral 1000 and 500 on the obverse of Rs.1000 and Rs.500 notes respectively is printed in optically variable ink viz., a color- shifting ink. The colour of the numeral 1000/500 appears green when the note is held flat but would change to blue when the note is held at an angle.Number panels of the notes are printed in fluorescent ink. The notes also have optical fibers. Both can be seen when the notes are exposed to ultra-violet lamp.The Rs.5, Rs.10, Rs.20 and Rs.50 notes contain a readable, fully embedded windowed security thread with the inscription “Bharat” (in Hindi), and “RBI”. The security thread appears to the left of the Mahatma's portrait.The portrait of Mahatma Gandhi, the Reserve Bank seal, guarantee and promise clause,Ashoka Pillar Emblem on the left, RBI Governor's signature are printed in intaglio i.e. in raised prints, which can be felt by touch, in Rs.20, Rs.50, Rs.100, Rs.500 and Rs.1000 notes.

Figure 4.1 Security Features of a note

On the obverse side of Rs.1000, Rs.500, Rs.100, Rs.50 and Rs.20 notes, a vertical band on the right side of the Mahatma Gandhi’s portrait contains a latent image showing the respective denominational value in numeral. The latent image is visible only when the note is held horizontally at eye level.This feature appears between the vertical band and Mahatma Gandhi portrait. It always contains the word “RBI” in Rs.5 and Rs.10. The notes of Rs.20 and above also contain the denominational value of the notes in micro letters. This feature can be seen well under a magnifying glass.Each note has a unique mark of it. This feature is in different shapes for various denominations (100-Triangle, Rs.500-Circle, and Rs.1000- Diamond) and helps the visually impaired to identify the denomination.

# CHAPTER 05 METHODOLOGY AND IMPLEMENTATION



## Methodology



### 5.1.1 Image Acquisition

The real-time image of currency is captured using camera is taken as the input. We can use webcam with hardware setup to capture the currency. The acquired image is later converted into the respective RGB histograms.



Figure 5.1 Acquired Image

### 5.1.2 RGB to Grayscale Conversion

The real-time image of currency is captured using camera is taken as the input. We can use webcam with hardware setup to capture the currency.



Figure 5.2 Gray Scale Conversion

### 5.1.3 Image segmentation:

Text

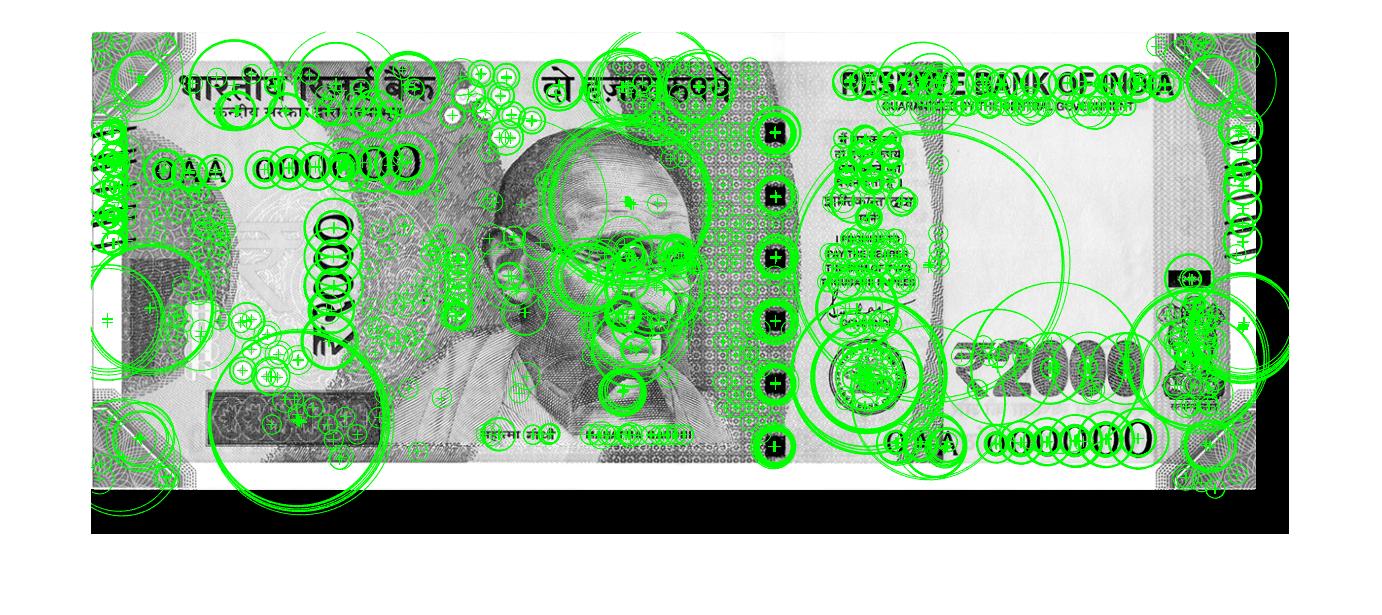
Description automatically generated with medium confidenceA close-up of a person's face

Description automatically generated with low confidence  
 It’s the process of dividing image into multiple parts by cropping it.

Figure 5.3 Image Segmentation

### 5.1.4 Feature extraction:

Now the features are extracted using MSER feature extraction.





**Figure 5.4 Feature Extraction**

### 5.1.5 Edge Detection:

Here we detected the edges of the image with the help of Canny and Prewitts algorithms.



Figure 5.5 Edges Detection

## 5.2 Implementation

## 5.2.1 Software Used

The necessary program regarding private PC that comprises configuration as specified as follows:

Windows 7(64-bit) operating system. MATLAB 7.14 Version R2012a

### Image Processing Toolbox

Image processing device box permits carrying out image improvement, deblurring of image, characteristic identification, decreasing of noise, image segmentation, arithmetical alteration, as well as registration of image. Image processing device intended for the execution regarding methods proposed are specified below: Fundamental import as well as export Display.

# CHAPTER 06 APPLICATIONS AND ADVANTAGES



## 6.1 Applications

* + - * Fake currency detection system can be utilized in shops.
      * Fake currency detection system can be utilized in banks counter.
      * Fake currency detection system can be utilized in computerized teller machine, auto merchant machines.

## 6.2 Advantages

* + - * Stops inflation.
      * Helps in maintaining stable economy.
      * Maintain actual flow of real notes.
      * The circulation of a large amount of fake currency can be reduced.

# CHAPTER 07 SOURCE CODE



## Main.m

function varargout = main(varargin)

% MAIN MATLAB code for main.fig

% MAIN, by itself, creates a new MAIN or raises the existing

% singleton\*.

%

% H = MAIN returns the handle to a new MAIN or the handle to

% the existing singleton\*.

%

% MAIN('CALLBACK',hObject,eventData,handles,...) calls the local

% function named CALLBACK in MAIN.M with the given input arguments.

%

% MAIN('Property','Value',...) creates a new MAIN or raises the

% existing singleton\*. Starting from the left, property value pairs are

% applied to the GUI before main\_OpeningFcn gets called. An

% unrecognized property name or invalid value makes property application

% stop. All inputs are passed to main\_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 main

% Last Modified by GUIDE v2.5 05-Jan-2022 13:00:48

% Begin initialization code - DO NOT EDIT

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @main\_OpeningFcn, ...

'gui\_OutputFcn', @main\_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 main is made visible.

function main\_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 main (see VARARGIN)

% Choose default command line output for main

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

% UIWAIT makes main wait for user response (see UIRESUME)

% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

function varargout = main\_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 pushbutton1.

function pushbutton1\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

Testimage;

% --- 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)

sub\_test;

% --- If Enable == 'on', executes on mouse press in 5 pixel border.

% --- Otherwise, executes on mouse press in 5 pixel border or over text2.

% --- Executes during object creation, after setting all properties

% --- Executes when uipanel2 is resized.

% --- Executes during object creation, after setting all properties.

function axes2\_CreateFcn(hObject, eventdata, handles)

% hObject handle to axes2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

imshow('C:\Users\User\Desktop\We see you\MGIT.png');

% Hint: place code in OpeningFcn to populate axes2

% --- Executes on button press in pushbutton4.

## TestImage.m

function varargout = Testimage(varargin)

% TESTIMAGE MATLAB code for Testimage.fig

% TESTIMAGE, by itself, creates a new TESTIMAGE or raises the existing

% singleton\*.

%

% H = TESTIMAGE returns the handle to a new TESTIMAGE or the handle to

% the existing singleton\*.

%

% TESTIMAGE('CALLBACK',hObject,eventData,handles,...) calls the local

% function named CALLBACK in TESTIMAGE.M with the given input arguments.

%

% TESTIMAGE('Property','Value',...) creates a new TESTIMAGE or raises the

% existing singleton\*. Starting from the left, property value pairs are

% applied to the GUI before Testimage\_OpeningFcn gets called. An

% unrecognized property name or invalid value makes property application

% stop. All inputs are passed to Testimage\_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 Testimage

% Last Modified by GUIDE v2.5 05-Jan-2022 12:35:38

% Begin initialization code - DO NOT EDIT

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @Testimage\_OpeningFcn, ...

'gui\_OutputFcn', @Testimage\_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 Testimage is made visible.

function Testimage\_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 Testimage (see VARARGIN)

% Choose default command line output for Testimage

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

% UIWAIT makes Testimage wait for user response (see UIRESUME)

% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

function varargout = Testimage\_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;

function edit1\_Callback(hObject, eventdata, handles)

% hObject handle to edit1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text

% str2double(get(hObject,'String')) returns contents of edit1 as a double

% --- Executes during object creation, after setting all properties.

function edit1\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edit1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes on button press in pushbutton1.

function pushbutton1\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global Iscaned

[imname,impath]=uigetfile({'\*.jpg;\*.png'},'Provide Currency for Testing');

Iscaned=imread([impath,'/',imname]);

axes(handles.axes1);

imshow(Iscaned);

setappdata(0,'Iscaned',Iscaned);

% --- Executes during object creation, after setting all properties.

function figure1\_CreateFcn(hObject, eventdata, handles)

% hObject handle to figure1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

function edit5\_Callback(hObject, eventdata, handles)

% hObject handle to edit5 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text

% str2double(get(hObject,'String')) returns contents of edit5 as a double

% --- Executes during object creation, after setting all properties.

function edit5\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edit5 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes on button press in popupmenu.

function popupmenu\_Callback(hObject, eventdata, handles)

% hObject handle to popupmenu (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global Ireal;

c=get(hObject,'Value');

switch(c)

case 2

Ireal=imread('C:\Users\User\Desktop\We see you\MINI\Fake Currency Detection (F5,F6,F7)\RBI\100.jpg');

axes(handles.axes2);

imshow(Ireal);

setappdata(0,'Ireal',Ireal);

note=100;

case 3

Ireal=imread('C:\Users\User\Desktop\We see you\MINI\Fake Currency Detection (F5,F6,F7)\RBI\200RBI.jpg');

axes(handles.axes2);

imshow(Ireal);

setappdata(0,'Ireal',Ireal);

note=200;

case 4

Ireal=imread('C:\Users\User\Desktop\We see you\MINI\Fake Currency Detection (F5,F6,F7)\RBI\500RBI.jpg');

axes(handles.axes2);

imshow(Ireal);

setappdata(0,'Ireal',Ireal);

note=500;

case 5

Ireal=imread('C:\Users\User\Desktop\We see you\MINI\Fake Currency Detection (F5,F6,F7)\RBI\2000RBI.png');

axes(handles.axes2);

imshow(Ireal);

setappdata(0,'Ireal',Ireal);

note=2000;

end

% --- 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)

global Ireal;

global Iscaned

compare(Ireal,Iscaned);

% --- Executes on button press in view2.

function view2\_Callback(hObject, eventdata, handles)

% hObject handle to view2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

sub\_test;

function edit7\_Callback(hObject, eventdata, handles)

% hObject handle to edit7 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text

% str2double(get(hObject,'String')) returns contents of edit7 as a double

% --- Executes during object creation, after setting all properties.

function edit7\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edit7 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes during object creation, after setting all properties.

## Subtest.m

function varargout = sub\_test(varargin)

% SUB\_TEST MATLAB code for sub\_test.fig

% SUB\_TEST, by itself, creates a new SUB\_TEST or raises the existing

% singleton\*.

%

% H = SUB\_TEST returns the handle to a new SUB\_TEST or the handle to

% the existing singleton\*.

%

% SUB\_TEST('CALLBACK',hObject,eventData,handles,...) calls the local

% function named CALLBACK in SUB\_TEST.M with the given input arguments.

%

% SUB\_TEST('Property','Value',...) creates a new SUB\_TEST or raises the

% existing singleton\*. Starting from the left, property value pairs are

% applied to the GUI before sub\_test\_OpeningFcn gets called. An

% unrecognized property name or invalid value makes property application

% stop. All inputs are passed to sub\_test\_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 sub\_test

% Last Modified by GUIDE v2.5 05-Jan-2022 12:34:01

% Begin initialization code - DO NOT EDIT

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @sub\_test\_OpeningFcn, ...

'gui\_OutputFcn', @sub\_test\_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 sub\_test is made visible.

function sub\_test\_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 sub\_test (see VARARGIN)

% Choose default command line output for sub\_test

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

% UIWAIT makes sub\_test wait for user response (see UIRESUME)

% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

function varargout = sub\_test\_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)

h=findobj('tag','view2');

% Get default command line output from handles structure

varargout{1} = handles.output;

% --- Executes on button press in Segmentation.

function Segmentation\_Callback(hObject, eventdata, handles)

% hObject handle to Segmentation (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global Ireal;

global Iscaned;

Ireal=getappdata(0,'Ireal');

Iscaned=getappdata(0,'Iscaned');

blackStripReal = Ireal(:,210:325,:);

blackStripscaned = Iscaned(:,210:325,:);

figure;

subplot(1,3,1);

imshow(blackStripReal);

title('Original');

subplot(1,3,2);

imshow(blackStripscaned);

title('scaned Description');

%%//Convert into grayscale then threshold

blackStripReal = rgb2gray(blackStripReal);

blackStripscaned = rgb2gray(blackStripscaned);

figure;

subplot(1,3,1);

imshow(blackStripReal);

title('Original');

subplot(1,3,2);

imshow(blackStripscaned);

title('scaned Description');

%leftmost part

blackStripReal = Ireal(:,10:110,:);

blackStripscaned = Iscaned(:,10:110,:);

subplot(1,3,1);

imshow(blackStripReal);

title('Original');

subplot(1,3,2);

imshow(blackStripscaned);

title('scaned');

%%//Convert into grayscale then threshold

blackStripReal = rgb2gray(blackStripReal);

blackStripscaned = rgb2gray(blackStripscaned);

figure;

subplot(1,3,1);

imshow(blackStripReal);

title('Original');

subplot(1,3,2);

imshow(blackStripscaned);

title('scaned');

%%//Extract Gandhi

blackStripReal = Ireal(:,400:650,:);

blackStripscaned = Iscaned(:,400:650,:);

figure;

subplot(1,3,1);

imshow(blackStripReal);

title('Original');

subplot(1,3,2);

imshow(blackStripscaned);

title('scaned');

%%//Convert into grayscale then threshold

blackStripReal = rgb2gray(blackStripReal);

blackStripscaned = rgb2gray(blackStripscaned);

figure;

subplot(1,3,1);

imshow(blackStripReal);

title('Original');

subplot(1,3,2);

imshow(blackStripscaned);

title('scaned');

blackStripReal = Ireal(:,600:775,:);

blackStripscaned = Iscaned(:,600:775,:);

figure;

subplot(1,3,1);

imshow(blackStripReal);

title('Original');

subplot(1,3,2);

imshow(blackStripscaned);

title('scaned');

%%//Convert into grayscale then threshold

blackStripReal = rgb2gray(blackStripReal);

blackStripscaned = rgb2gray(blackStripscaned);

figure;

subplot(1,3,1);

imshow(blackStripReal);

title('Original');

subplot(1,3,2);

imshow(blackStripscaned);

title('scaned');

%%//2000 Below

blackStripReal = Ireal(:,850:1100,:);

blackStripscaned = Iscaned(:,850:1100,:);

subplot(1,3,1);

imshow(blackStripReal);

title('Original');

subplot(1,3,2);

imshow(blackStripscaned);

title('scaned');

%%//Convert into grayscale then threshold

blackStripReal = rgb2gray(blackStripReal);

blackStripscaned = rgb2gray(blackStripscaned);

figure;

subplot(1,3,1);

imshow(blackStripReal);

title('Original');

subplot(1,3,2);

imshow(blackStripscaned);

title('scaned');

%%//Ashoka emblem

blackStripReal = Ireal(:,1050:1145,:);

blackStripscaned = Iscaned(:,1050:1145,:);

figure;

subplot(1,3,1);

imshow(blackStripReal);

title('Original');

subplot(1,3,2);

imshow(blackStripscaned);

title('scaned');

%%//Convert into grayscale then threshold

blackStripReal = rgb2gray(blackStripReal);

blackStripscaned = rgb2gray(blackStripscaned);

figure;

subplot(1,3,1);

imshow(blackStripReal);

title('Original');

subplot(1,3,2);

imshow(blackStripscaned);

title('scaned');

%%// latent

blackStripReal = Ireal(350:519,110:400,:);

blackStripscaned = Iscaned(350:519,110:400,:);

figure;

subplot(1,3,1);

imshow(blackStripReal);

title('Original');

subplot(1,3,2);

imshow(blackStripscaned);

title('scaned');

%%//Convert into grayscale then threshold

blackStripReal = rgb2gray(blackStripReal);

blackStripscaned = rgb2gray(blackStripscaned);

figure;

subplot(1,3,1);

imshow(blackStripReal);

title('Original');

subplot(1,3,2);

imshow(blackStripscaned);

title('scaned');

%%Entire left part

blackStripReal = Ireal(:,110:400,:);

blackStripscaned = Iscaned(:,110:400,:);

figure;

subplot(1,3,1);

imshow(blackStripReal);

title('Original');

subplot(1,3,2);

imshow(blackStripscaned);

title('scaned');

%%//Convert into grayscale then threshold

blackStripReal = rgb2gray(blackStripReal);

blackStripscaned = rgb2gray(blackStripscaned);

figure;

subplot(1,3,1);

imshow(blackStripReal);

title('Original');

subplot(1,3,2);

imshow(blackStripscaned);

title('scaned');

% --- Executes on button press in post\_analysis.

function post\_analysis\_Callback(hObject, eventdata, handles)

% hObject handle to post\_analysis (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

global Ireal;

global Iscaned;

Ireal=getappdata(0,'Ireal');

Iscaned=getappdata(0,'Iscaned');

% handles structure with handles and user data (see GUIDATA)

call(Ireal,Iscaned);

% --- Executes on button press in Pre\_analysis.

function Pre\_analysis\_Callback(hObject, eventdata, handles)

% hObject handle to Pre\_analysis (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

%%//Pre-analysis

global Ireal;

global Iscaned;

Ireal=getappdata(0,'Ireal');

Iscaned=getappdata(0,'Iscaned');

hsvImageReal = rgb2hsv(Ireal);

hsvImagescaned = rgb2hsv(Iscaned);

figure;

imshow([hsvImageReal(:,:,1) hsvImageReal(:,:,2) hsvImageReal(:,:,3)]);

title('Real pre process');

figure;

imshow([hsvImagescaned(:,:,1) hsvImagescaned(:,:,2) hsvImagescaned(:,:,3)]);

title('scaned');

% --- Executes on button press in Histogram.

function Histogram\_Callback(hObject, eventdata, handles)

% hObject handle to Histogram (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global Ireal;

global Iscaned;

Ireal=getappdata(0,'Ireal');

Iscaned=getappdata(0,'Iscaned');

figure;

subplot(2,4,1);

imshow(Ireal);

subplot(2,4,2);

I1=Ireal(:,:,1);

imhist(I1);

subplot(2,4,3);

I2=Ireal(:,:,2);

imhist(I2);

subplot(2,4,4);

I1=Ireal(:,:,3);

imhist(I1);

subplot(2,4,5);

imshow(Iscaned);

subplot(2,4,6);

I1=Iscaned(:,:,1);

imhist(I1);

subplot(2,4,7);

I1=Iscaned(:,:,2);

imhist(I1);

subplot(2,4,8);

I1=Iscaned(:,:,3);

imhist(I1);

% --- Executes on button press in Features.

function Features\_Callback(hObject, eventdata, handles)

% hObject handle to Features (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global Ireal

global Iscaned

Features(Ireal,Iscaned)

function Edgedetection\_Callback(hObject, eventdata, handles)

% hObject handle to Edgedetection (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global Ireal

global Iscaned

edgesdetection(Ireal,Iscaned);

## Call.m

function call(Ireal,Iscaned)

hsvImageReal = rgb2hsv(Ireal);

hsvImagescaned = rgb2hsv(Iscaned);

croppedImageReal = hsvImageReal(:,90:95,:);

croppedImagescaned = hsvImagescaned(:,93:98,:);

satThresh = 0.4;

valThresh = 0.3;

BWImageReal = (croppedImageReal(:,:,2) > satThresh & croppedImageReal(:,:,3) < valThresh);

figure;

subplot(1,2,1);

imshow(BWImageReal);

title('Real initial seg');

BWImagescaned = (croppedImagescaned(:,:,2) > satThresh & croppedImagescaned(:,:,3) < valThresh);

subplot(1,2,2);

imshow(BWImagescaned);

title('scaned');

%%//Post-process

se = strel('line', 6, 90);

BWImageCloseReal = imclose(BWImageReal, se);

BWImageClosescaned = imclose(BWImagescaned, se);

figure;

subplot(1,2,1);

imshow(BWImageCloseReal);

title('Real post process');

subplot(1,2,2);

imshow(BWImageClosescaned);

title('scaned');

%%//Area open the image

figure;

areaopenReal = bwareaopen(BWImageCloseReal, 15);

subplot(1,2,1);

imshow(areaopenReal);

title('Real area open image');

subplot(1,2,2);

areaopenscaned = bwareaopen(BWImageClosescaned, 15);

imshow(areaopenscaned);

title('scaned');

%//Count how many objects there are

[~,countReal] = bwlabel(areaopenReal);

[~,countscaned] = bwlabel(areaopenscaned);

disp(['The total number of black lines for the real note is: ' num2str(countReal)]);

disp(['The total number of black lines for the scaned note is: ' num2str(countscaned)]);

grt = rgb2gray(Ireal);

grs = rgb2gray(Iscaned);

% contrast enhance the gray image to emphasize dark lines in lighter background

grt = imadjust(grt);

grs = imadjust(grs);

% close rgb. choose a larger k. idea is to remove the dark line

k = 7;

se = ones(k);

Irealcl = imclose(Ireal, se);

Iscanedcl = imclose(Iscaned, se);

% convert closed image to gray scale

grtcl = rgb2gray(Irealcl);

grscl = rgb2gray(Iscanedcl);

% take the difference (closed-gray-scale - contrast-enhanced-gray-scale)

difft = grtcl - grt;

diffs = grscl - grs;

% take the projection of the difference

pt = sum(difft');

pf = sum(diffs');

% smooth the projection

ptfilt = conv(pt, ones(1, k)/k, 'same');

pffilt = conv(pf, ones(1, k)/k, 'same');

% threshold (multiplication by max element is just for visualization)

tht = (pt > graythresh(pt))\*max(pt);

thf = (pf > graythresh(pf))\*max(pf);

% get the number of segments.

[lblt, nt] = bwlabel(tht);

[lblf, nf] = bwlabel(thf);

figure,

subplot(2, 1, 1), imshow(difft'), title('difference image for solid line')

subplot(2, 1, 2), imshow(diffs'), title('difference image for broken line')

figure,

subplot(2, 1, 1), plot(1:length(pt), pt, 1:length(pt), ptfilt, 1:length(pt), tht),

title('solid line image'),

legend('projection', 'smoothed', 'thresholded', 'location', 'eastoutside')

subplot(2, 1, 2), plot(1:length(pf), pf, 1:length(pf), pffilt, 1:length(pf), thf),

title('broken line image'),

legend('projection', 'smoothed', 'thresholded', 'location', 'eastoutside')

## Compare.m

function compare(Ireal,Iscaned)

I1=im2double(Ireal);

I1 = rgb2gray(I1);

I2=im2double(Iscaned);

I2 = rgb2gray(Iscaned);

corners1 = detectMSERFeatures(I1);

% [features1, valid\_corners1] = extractFeatures(I1, corners);

% figure; imshow(I1); hold on

% plot(valid\_corners1);

c=length(corners1)

corners2 = detectMSERFeatures(I2);

% [features1, valid\_corners2] = extractFeatures(I2, corners);

% figure; imshow(I2); hold on

% plot(valid\_corners2);

d=length(corners2)

t=0.75\*c

if(d>t)

h=msgbox("The note is Real")

else

h=msgbox("The note is fake")

end

## Features.m

function Features(Ireal,Iscaned)

I1=im2double(Ireal);

I1 = rgb2gray(I1);

I2=im2double(Iscaned);

I2 = rgb2gray(Iscaned);

corners = detectMSERFeatures(I1);

[features1, valid\_corners1] = extractFeatures(I1,corners);

figure; imshow(I1); hold on

plot(valid\_corners1);

corners2 = detectMSERFeatures(I2);

[features2, valid\_corners2] = extractFeatures(I2, corners);

figure; imshow(I2); hold on

plot(valid\_corners2);

## Edgesdetection.m

function edgesdetection(Ireal,Iscaned)

figure;

subplot(2,2,1);

imshow(Ireal);

I1=rgb2gray(Ireal);

title('Original Image');

subplot(2,2,2);

BW1 = edge(I1,'Canny');

BW2 = edge(I1,'Prewitt');

imshowpair(BW1,BW2);

title('Detected Edges');

subplot(2,2,3);

imshow(Iscaned);

I2=rgb2gray(Iscaned);

title('Original Image');

subplot(2,2,4);

BW1 = edge(I2,'Canny');

BW2 = edge(I2,'Prewitt');

imshowpair(BW1,BW2);

title('Detected Edges');

# CHAPTER 08 RESULTS

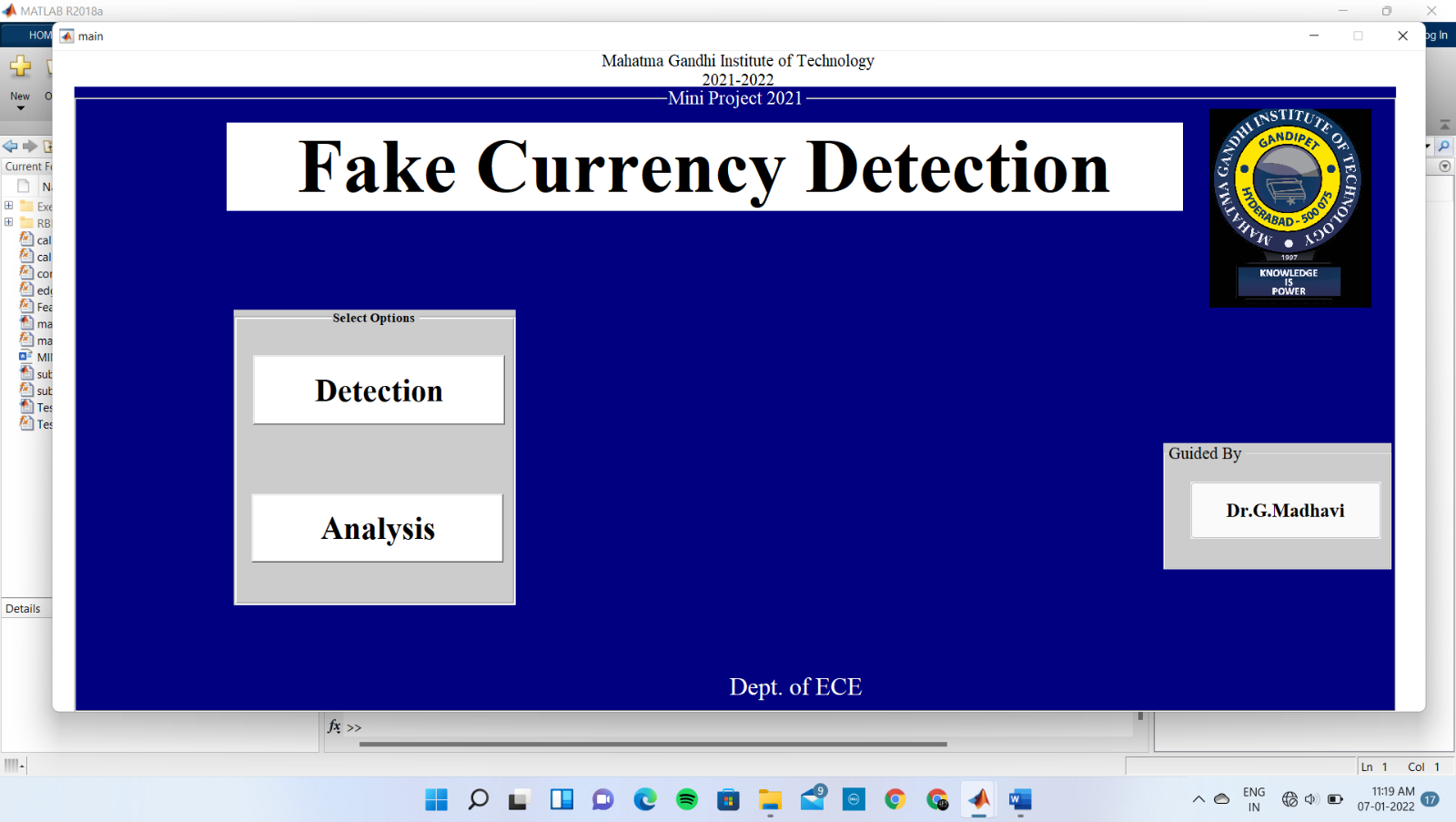
****

Figure 8.1 Main.m

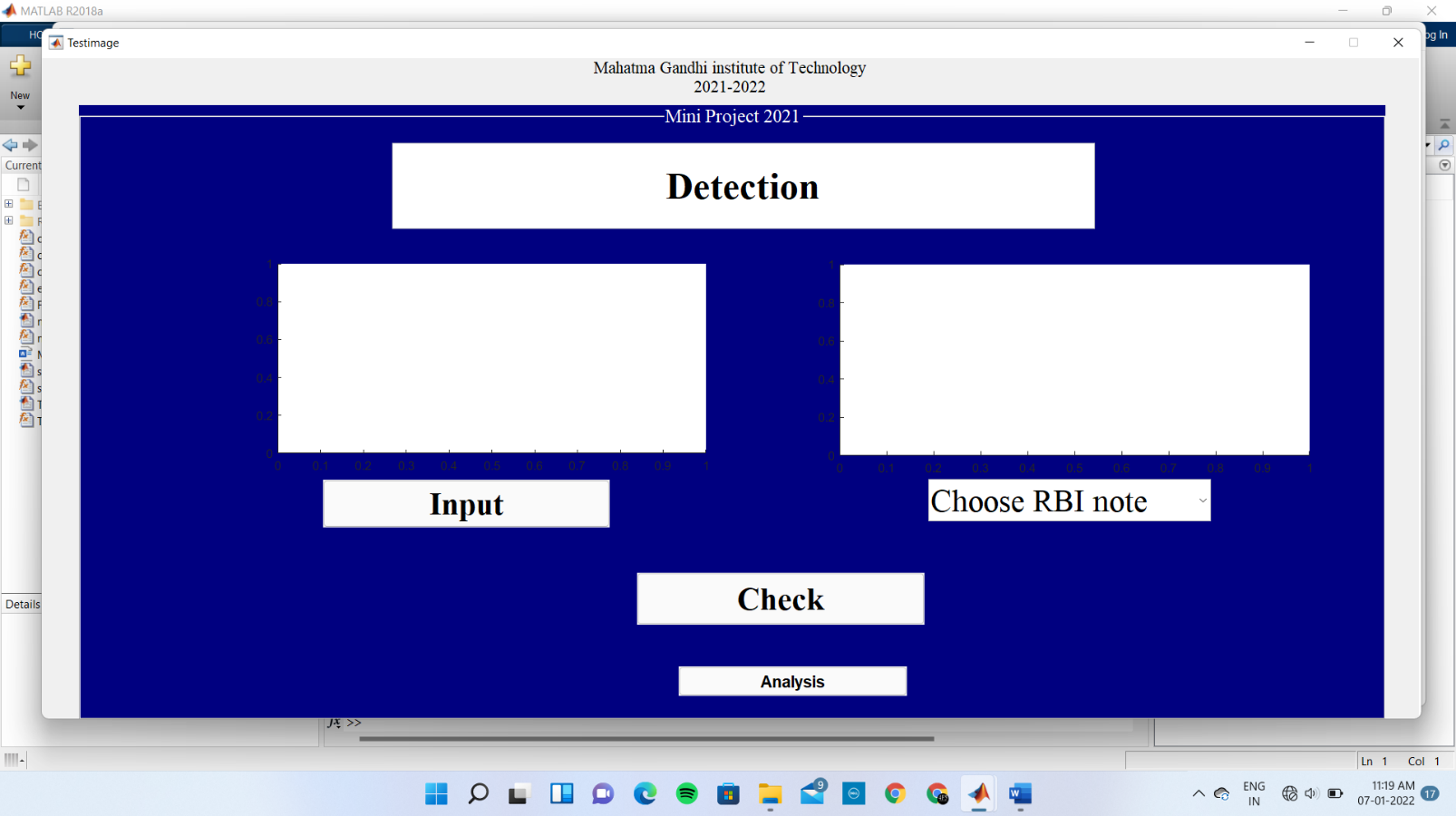
****

Figure 8.2 Testimage.m

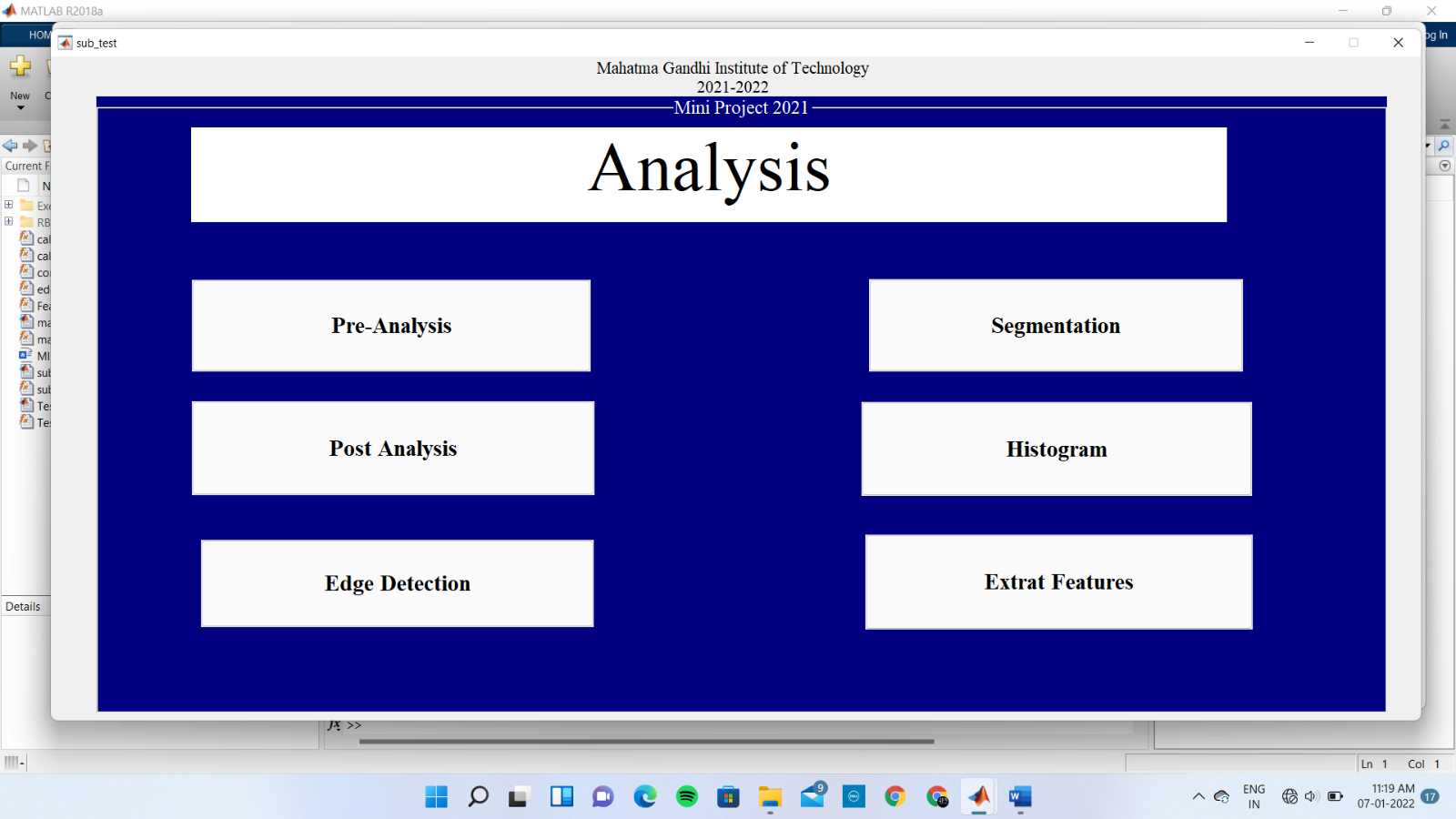
****

Figure 8.3 0.1subtest.m

A picture containing text

Description automatically generatedA picture containing text

Description automatically generated

**8.4(a)**

Text

Description automatically generated with medium confidence

**8.4(b)**

Letter

Description automatically generated with medium confidence

**8.4(c)**

**8.4(d)**

A close-up of a person's face

Description automatically generated with low confidence

Text

Description automatically generated with medium confidence

**8.4(f)**

**8.4(e)**

A picture containing text

Description automatically generatedA picture containing text, businesscard

Description automatically generated

**8.4(h)**

**8.4(g)**

Figure 8.4(a-h) Segmented images of scanned and real notes

Figure 8.50.2 feature extraction using MSER algorithmFigure 8.4(a-h) Segmented images of scanned and real notes

Figure 8.50.3 feature extraction using MSER algorithm

Figure 8.50.4 feature extraction using MSER algorithmFigure 8.4(a-h) Segmented images of scanned and real notes

Figure 8.50.5 feature extraction using MSER algorithmFigure 8.4(a-h) Segmented images of scanned and real notes

**Map

Description automatically generated with low confidenceA close-up of a dollar bill

Description automatically generated with medium confidence**

Figure 8.50.6 feature extraction using MSER algorithm

# CHAPTER 09 CONCLUSION

MATLAB software is used to extract the features of the note. The proposed system has got advantages like simplicity and high-performance speed. The result will predict whether the currency note is fake or not. In image processing part, the different features present in the testing input note is identified and compared with the real note and results as the given note is fake or not. This all modules are implemented in MATLAB. We have implemented a fake note detection unit with image processing algorithms. The simulation results indicate that the results achieved are nearly accurate, since the features extracted are compared and output is generated.

# REFERENCES

1. Sanjana, Manoj Diwakar, Anand Sharma, "An Automated recognition of Fake or Destroyed Indian currency notes in Machine vision", IJCSMS, Vol. 12, April 2012.
2. Hanish Agarwal, Padam Kumar, "Indian currency note denomination recognition in color images", IJACEC, Vol. 1, ISSN 2278–5140.
3. Foresti G.L, Regazzoni C,"A hierarchical approach to feature extraction and grouping” IEEE Trans Image Processing, 2000; 9(6):1056-74.
4. Euisun Choi, Jongseok Lee and Joonhyun Yoon,"Feature Extraction for Bank Note Classification Using Wavelet Transform”2006 @ISBN ISSN: 1051-4651, 0-7695-2521- 0, IEEE.
5. Peng Wang and Peng Liu, ″Invariant Features Extraction for Banknote Classification “Proceedings of the 11th Joint Conference on Information Sciences@2008.
6. Ahmadi and S.Omatu,” A Methodology to Evaluate and Improve Reliability in Paper Currency Neuro-Classifiers” Proceedings 2003 IEEE International Symposium on Computational intelligence in Robotics and Automation July 16-20,2003, Kobe, Japan.
7. E.H.Zhang, B. Jiang, J. H. Duan, Z. Z. Bian,“ Research on Paper Currency Recognition by Neural Networks”, Proceedings of the 2nd Int. Conference on Machine Learning and Cybernetics, 2003
8. Jae-Kang Lee and Hwan Kim,“New Recognition Algorithm for Various Kinds of Euro Banknotes” 0-7803-7906-3/03/ 2003 IEEE.
9. F.H Kong, J.Quab Ma , J.Feng Liu,“Paper Currency Recognition Using Gaussian Mixture Models based on Structural Risk Minimization” Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, Dalian, 13-16 August 2006.
10. Ji Qian,Dongping Qian,Mengjie Zhang,″A Digit Recognition System for Paper Currency Identification Based on Virtual Instruments”1-4244-0555-6/06 © 2006 IEEE.
11. Nadim Jahangir and Ahsan Raja Chowdhury,“Bangladeshi Banknote Recognition by Neural Network with Axis Symmetrical Masks” 1-4244-1551-9/07/$25.00 ©2007 IEEE.
12. H.Hassanpouri, A.Yaseri, G.Ardeshiri, “Feature Extraction for Paper Currency Recognition” 1-4244-0779-6/07/2007 IEEE.