A REAL-TIME PROJECT REPORT ON

FAKEE CURRENCY DETECTION USING MATLAB

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT
FOR THEAWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY
IN
Electronics & Communication Engineering

SUBMITTED BY

B.PRANITH	227R1A0475
CH.ABHILASH	227R1A0477
P.SRIRAM	227R1A04A7
N.RAHUL	227R1A0498

UNDER THE GUIDANCE OF

DR.P.VENKATAKRISHAN
Professor ECE Dept.

CMR
GROUP OF INSTITUTIONS
EXPLORE TO INVENT

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CMR TECHNICAL
CAMPUSUGC
AUTONOMOUS

Kandlakoya (V), Medchal Road, Hyderabad-501401 (2023-2024)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERI NG

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

Kandlakoya (V), Medchal Road, Hyderabad-501401

CERTIFICATE

This is to certify that the dissertation work entitled **"FAKE CURRENCY DETECTION USING MATLAB" is** being submitted by **B.PRANITH, CH.ABHILASH, P.SRIRAM and N.RAHUL** bearing **Roll Numbers 227R1A0475, 227R1A0477, 227R1A04A7, and 227R1A0498** respectively, in partial fulfillment for the degree of **Bachelor of Technology** in **"ELECTRONICS AND COMMUNICATION ENGINEERING"** during the academic year 2020–2021.

Certified further, to the best of our knowledge that the work reported is not a partof any other project on the basis of which a degree or an award has been given on an earlier occasion to any other candidate. The results have been verified and found to be satisfactory.

Internal Guide	Head of the Department
Dr.K.BHARATH KUMAR	**Prof.G.SRIKANTH**
Associate Professor	**Professor & HOD, ECE Dept.**

Director
Dr. A. RAJIREDDY

ACKNOWLEDGEMENT

We thank Dr .K Bharath Kumar Associate Professor, Department of ECE for her constant guidance over the past six months, and for providing required guidance as internal guide for result oriented implementation of ideas relevant to my project.

We thank Dr. K. Bharathkumar, Mini Project Coordinator for her constant support in carrying out the project activities and reviews.

We thank Prof. G. Srikanth, HOD of Electronics and Communication Engineering for providing seamless support and knowledge over the past one year and for providing right suggestions at every phase for successful completion of my project.

We express my whole-hearted gratitude to Dr. A. Raji Reddy, Director, CMR Technical campus for providing us the conducive environment for carrying out my academic schedules and projects with ease.

There is a need to thank all staff members, friends and parents without whose support project would have been deferred.

B.PRANITH	**(227R1A0481)**
CH.ABHILASH	**(227R1A0485)**
P.SRIRAM	**(227R1A0469)**
N.RAHUL	**(227R1A04B9)**

Table of Contents

Abstract.. 6

Introduction ……..…………………………………………………………………….…7-8

Literature survey……………………………………………………………...…….9-12

Chapter 1

INTRODUCTION

History of MATLAB 12-14

Developing of MATLAB 14-16

Description about files 16-17

Chapter 2

Code Breakdown 17-20

Chapter 3

Overview of the project

3.1 Introduction to Image Processing 21

3.2 Image Acquisition 21-22

3.3 Formation of Image 22

3.4 Image Resizing/Scaling 22

3.5 RGB to GRAY Conversion 23-24

3.6 Image Enhancement 24

3.7 Edge Detection 24

3.8 Edge detection techniques 24

3.9 Canny Edge Detection 24-25

3.10 Image Matching 25

Chapter 4

Project code & output screen

4.1 GUI.m file 26-31

4.2 final project.m file 31

4.3 Output screen results 32-33

Chapter 5

Conclusion 34

References 35

ABSTRACT

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.

INTRODUCTION

The detection of fake currency is a critical issue that affects economies worldwide. As counterfeiters employ increasingly sophisticated methods, traditional manual detection techniques often fall short. This project leverages MATLAB to develop an automated system for fake currency detection, utilizing advanced image processing and pattern recognition algorithms.

MATLAB, with its powerful computational and visualization capabilities, offers an ideal environment for creating robust and efficient detection systems. This documentation outlines the project's objectives, methodologies, and implementation details, providing a comprehensive guide to understanding and replicating the system.

Economic development of every nation is mostly dependent on its currency and every person is the part of Economy but some of the unsocial group of people damage this process and unbalances the social harmony of the nation. For ex. Now a days, in process of demonetization, there are long queues in front of banks and ATM Machines of those common people who contribute to our economy by paying taxes but on the other hand many corrupted people are issuing the money directly by evil sources and it is directly effecting on economic status of India. As we know, in India, Ministry Of Finance and RBI(Reserve Bank Of India are authorized to issue currency notes and coins. But corrupt people take the advantage of high printing and scanning technologies to print fake notes by using latest hardware tools and techniques.

Fake currency detection means finding the fake currency from the original one. Generally, currency recognition system is mostly used in banks, business firms, shopping malls, rail- way stations, government sector, organizations etc. But common people do not have any source of currency detection and they are unable to identify the real original currency. That’s why the malpractice of fake currency is carried out openly in our economy. Till date, many researchers have given their contribution in finding the technique of identifying the genuine currency notes from the fake notes.

LITERATURE SURVEY

Several Techniques were proposed by various authors for Currency Identifications, Forged Banknote Detection. In this scenario, a brief evaluation of some important contributions to existing literature is presented shortly.

Gouri Sanjay et al. (2018), the counterfeiters are becoming harder to track down because of their use of highly advanced technology. One of the most effective methods to stop counterfeiting can be the use of counterfeit detection software that is available and efficient. Our project will recognize Indian currency notes using a real-time image obtained from a webcam. The background of our topic is image processing technology and applying it for the purpose of verifying currency notes. The software will detect fake currency by extracting features of notes. The success rate of this software can be measures in terms of accuracy and speed.

Shital Mahajan et al. (2018), Surveys paper reports various articles dealing with counterfeit paper currency recognition and detection which attempts to represent the survey on fake money detection because almost every country in the world is facing the problem of forged money, but in India, the problem is exasperating as the country is hit hard by this evil practices. There has been no prior survey on the currency identification methodologies and the comprehensive survey presented in this paper will be useful in developing and analysing new approaches and algorithms with good performance.

Sumit Shahaniet et al. (2018), proposed machine learning techniques to evaluate authentication of banknotes. Supervised learning algorithms such as Back propagation Neural Network (BPN) and Support Vector Machine (SVM) are used for differentiating genuine banknotes from fake ones. The study also shows the comparison of these algorithm in classification of banknotes.

Ryutaro Kitagawa et al. (2017), proposed system to automate the configurations of a sorting system by automatically detect portraits in sample banknotes, so that it can be quickly deployed in a new target country. Used Convolutional Neural Networks to detect portraits in completely new set of banknotes robust to variation in the ways they are shown, such as the size and the orientation of the face.

P. Julia Grace et al. (2016), a thriving approach to paper currency identification depends upon a number of steps, including edge detection, feature extraction, image segmentation, image acquisition, grayscale conversion, and comparison of images. A different type of literature which describes different techniques of counterfeit currency identification to detect mal-practising. Finally concluded when we apply some efficient pre-processing and feature extraction techniques, still there is scope to improve the accuracy of currency identification system.

Spandan Sen Sarma (2016), focused on detection of authenticity of Bank Notes using several machine learning techniques. Accurate separation of original notes from the forged one is a challenging job. In the present work Neural Network (NN) has been trained using Genetic algorithm (GA employed to detect authenticity of bank notes by classifying them into two separate classes. The initial weight vector to the input layer of the NN has been optimized gradually using the optimization techniques to enhance the performance of NN to a greater extent. The experimental results of the proposed method have been compared with a well-known Multilayer Perceptron Feed-Forward Network (MLP-FFN) and also with the NN. Performance measures like accuracy, precision, recall and F-measure have been used to compare the performances of the algorithms. The experimental results have revealed significant improvement over the existing performances to detect forgery of bank notes using GA.

Ingulkar Ashwini Suresh et al. (2016), described recognition of paper currency with the help of digital image processing techniques. Around eight characteristics of Indian paper currency is selected for counterfeit detection. The identification 9 marks, optical variable link, see through register and currency colour code decides the currency recognition. The security threads, water mark, Latent image and micro-lettering features are used for currency verification. The characteristics extraction is performed on the image of the currency and it is compared with the characteristics of the genuine currency. The currency will be verified by using image processing techniques. The approach consists of a number of components including image processing, edge detection, image segmentation and characteristic extraction and comparing images. The desired results shall verify with MATLAB software.

Amruta Chavan et al. (2016), Object Detection and recognition is an important task in image processing and computer vision. It is concerned with determining the 11 identity of an object being observed in an image .Humans can recognize any object in the real world easily without any efforts. But computerize recognition of object in image is not easy task. In such system some problems are occurred such as lightning, mirroring, rotating. It also used Sketch based system for object detection in which user can draw the images by hand and then matches the images from the database. The system developed has many types of applications in the field of Medical Diagnose, Cartography, and Robotics.

Komal Vora et al. (2015), an algorithm based on the frequency domain feature extraction method is discussed for the detection of currency. This method efficiently utilizes the local spatial features in a currency image to recognize it. The entire system is pre-processed for the optimal and efficient implementation of two dimensional discrete wavelet transform (2D DWT) which is used to develop a currency recognition system. A set of coefficient statistical moments are then extracted from the approximate efficient matrix. The extracted features can be used for recognition, classification and retrieval of currency notes. The classification result will facilitate the recognition of fake currency mainly using serial number extraction by implementing OCR.

CHAPTER 1
INTRODUCTION TO MATLAB

1.1 History of MATLAB

MATLAB is programming tool for technical computing. This is the shortest definition. More accurate definition is in the end of this section.

MATLAB has grown due to needs of technical computing in the year 1980, when the university teacher of mathematics Cleve Moler from Stanford University wrote the basics of MATLAB for linear algebra – that is how to calculate with matrices (adding, subtracting, multiplying, dividing…) and how to solve linear systems.

Cleve Moler discovered that program languages are not user friendly for his needs to teach mathematics, soon. If students would use the program languages FORTRAN, Pascal, C, etc. they would loose a lot of time programming and learning these languages. This is the reason, why he had started to develop MATLAB, as interactive calculator, without possibility to program, just to calculate matrices etc. The first version of MATLAB was written in FORTRAN and it used the libraries LINPACK and EISPACK to calculate matrices. The next stage in the development of MATLAB was the year 1983, when Jack Little and Steve Bangert joined Cleve Moler and programmed the newer version of MATLAB in C language. They also added the possibility of interpreter programming in so called M-code and some other characteristics. They called this version of MATLAB – MATLAB 1.0 and was published on the market in the year 1984, when the company MATHWORKS was established. The company MATHWORKS has developed several newer versions from year to year. They introduced first version of SIMULINK, which offer also graphical programming in year 1990. The usability of MATLAB and SIMULINK is even improved by so called Toolboxes and Blocksets, which have been available and improved from year 1990. The newest version of MATLAB on the market is version 7.2. It is also available cheaper version of MATLAB just for students. We will present the basic of MATLAB trough the practical work during the next sections. Here you will find a little more complicate definition, what is MATLAB?

MATLAB is:

- higher program language oriented in solving technical problems with integrated interactive developing environment. Programming language in MATLAB is called M-code.

The characteristics of MATLAB are:

- Program language MATLAB has dynamic memory management, therefore there is no need to define in advance the names and types of variables, like in C.
- An interpreter of M-code is not a classical translator, as case of C-language. The interpreter translates every single command (row) and executes it immediately after was confirmed by type of Enter on a keyboard. In case of

a classical translator, the user has to write the complete program and after that the program is translated and than in the second stage executed. Therefore the MATLAB interpreter allows an interactive work.

- The time needed for developing program in M-code is pretty lower than in case of programming in Pascal, FORTRAN or in C-language.
- Every program written in M-code is interpreted and not translated in classical way, so the execution is slow.
- The drawback of the classical translated M-code is that this translated code is not transferrable to other microprocessors.

The MATLAB developing environment:

The program MATLAB has to be installed on the PC computer, first. After it is installed, you can find an icon (see Fig. 2.1) on the Desktop.

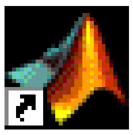
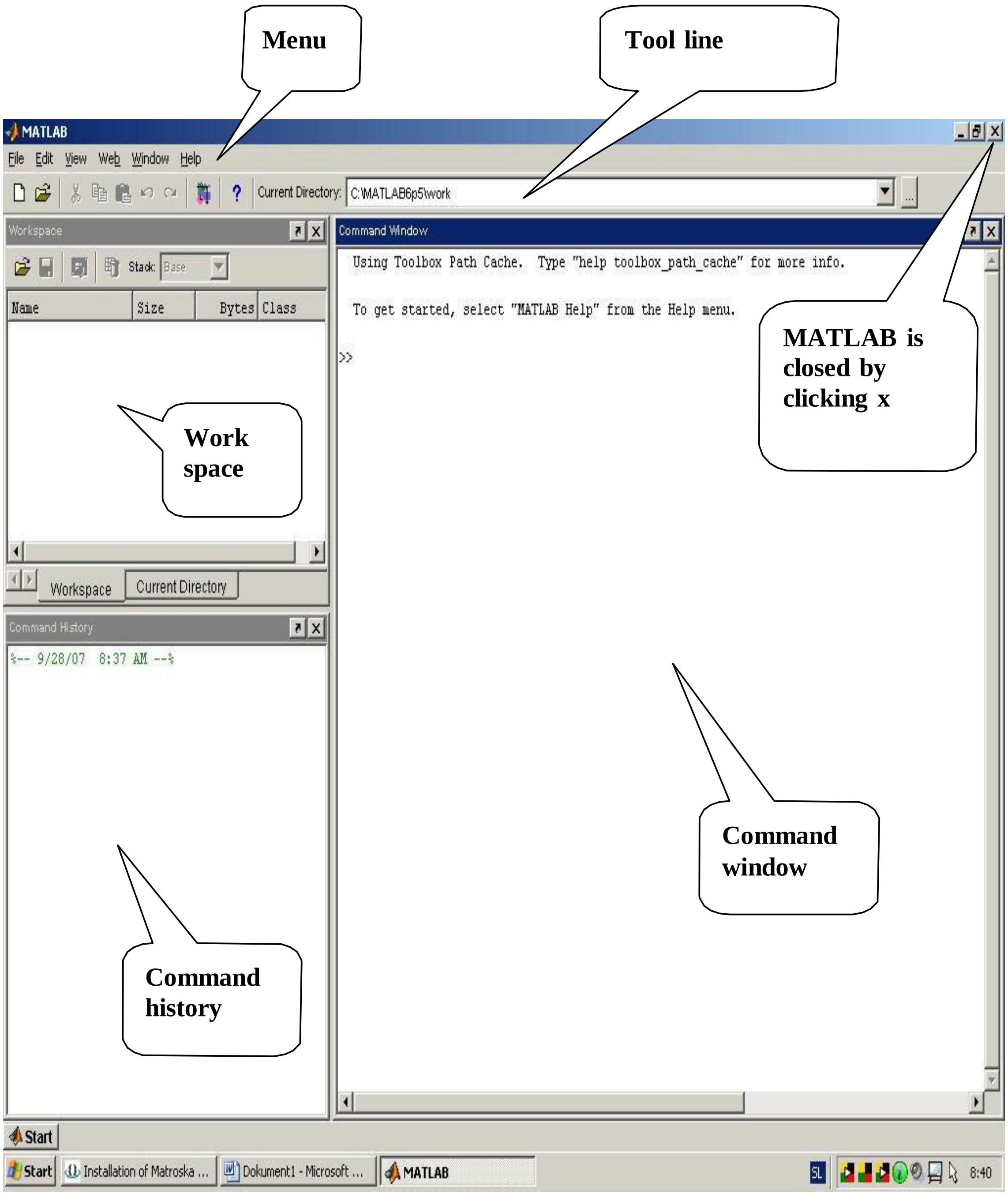


Figure 1.1: MATLAB application


Menu
Tool line
MATLAB is closed by clicking x
Work space
Command window
Command history
MATLAB
File Edit View Web Window Help
Current Directory: C:\MATLAB6p5\work
Workspace
Stack: Base
Name Size Bytes Class
Workspace Current Directory
Command History
Command Window
Start
Installation of Matroska ...
Dokument1 - Microsoft ...
8:40

```
Using Toolbox Path Cache.  Type "help toolbox_path_cache" for more info.

To get started, select "MATLAB Help" from the Help menu.

>>
```

```
%-- 9/28/07  8:37 AM --%
```

Figure 1.2: MATLAB developing environment

Figure 1.2 shows MATLAB developing environment which has:

- menu line,
- Tool line, which display current directory,
- Command window,
- Workspace window and

- Command history window.
    - The project consists of several files including GUI.m, GUI.fig, finalProject.m, and text files for saving image locations and results.

File Descriptions:

The MATLAB GUI application consists of several key files that work together to provide a seamless user experience:

1. **GUI.m**: This is the main script that contains the MATLAB code for the graphical user interface (GUI). It includes the initialization code, various callback functions for handling user interactions (such as button presses and menu selections), and functions for managing the GUI state and outputs.
2. **GUI.fig**: This file is the graphical layout of the GUI created using MATLAB’s GUIDE (Graphical User Interface Development Environment). It defines the placement and properties of the GUI components like buttons, text fields, axes for image display, and pop-up menus.
3. **finalProject.m**: A placeholder script or function where the main image processing or any other computational tasks are implemented. When the user requests to see the results, this function is executed to process the input image and generate the output.
4. **imageLoc.txt**: A text file used to store the location of the image file entered by the user. This allows the application to persist the image location between different sessions or uses.

5. **verdict.txt**: A text file used to store and display the results of the image processing or any other computational tasks performed by finalProject.m. The results are read from this file and displayed in the GUI.

These files collectively manage the user inputs, display the image, perform necessary computations, and display the results within the GUI.

Code Breakdown :

This section provides a detailed explanation of the code components and their functionality.

Initialization and GUI Setup:

The initialization code sets up the GUI, manages singleton behavior, and handles input arguments and outputs. The gui_mainfcn function is responsible for the core functionality of the GUI, setting up the state and managing callbacks.

Opening Function:

The GUI_OpeningFcn function is executed just before the GUI becomes visible. It initializes the default command line output for the GUI and updates the handles structure, which contains the GUI component handles and user data.

Callback Functions

Callback functions are essential for handling user interactions within the GUI. They define the behavior of the GUI in response to user inputs such as button presses and menu selections.

Set Image Location Callback

The setImageLocation_Callback function is a placeholder for the button press event to set the image location. It can be extended to include functionality for selecting and setting the image path.

Enter Location Callback

The enterLocation_Callback function handles the input of an image location, updates the application data, and saves the location to a text file. This ensures that the image path is stored and can be retrieved later.

Show Image Callback

The showImage_Callback function retrieves the image location from the application data, reads the image, and displays it in the GUI axes. This allows the user to visualize the selected image within the application.

Reset Callback

The reset_Callback function clears the display, resets the text file storing the image location, and updates the result text. This function is useful for resetting the GUI to its initial state.

Result Button Callback

The resultButton_Callback function executes the finalProject function, reads the result from a text file, and displays it in the GUI. This function provides feedback to the user about the outcome of the image processing task.

Pop-Up Menu Callback

The popUpMenu_Callback function handles the selection of items from a pop-up menu and saves the selected item to a text file. This function allows the user to make selections that influence the behavior of the GUI.

Exit Button Callback

The exitButton_Callback function clears all variables and closes the GUI, ensuring a clean exit from the application.

Additional Callback Functions

Several additional callback functions handle the creation and configuration of GUI components, ensuring proper initialization and behavior.

enterLocation_CreateFcn:

Sets the background color for the input field to ensure it is properly displayed on Windows systems.

To use the MATLAB GUI application, follow these steps:

1. **Input Image Location**: Enter the location of the image file in the input field and press the button to set the image location. This will store the path in the application data and save it to imageLoc.txt.
2. **Display Image**: Press the button to display the image in the GUI. The showImage_Callback function will read the image from the specified location and display it in the designated axes.

3. **Reset Interface**: Press the reset button to clear the display and reset the inputs. The reset_Callback function will clear the image display, reset the text fields, and update the result text.
4. **Show Results**: Press the result button to execute the finalProject function and display the results. The resultButton_Callback function will call finalProject, read the results from verdict.txt, and display them in the GUI.
5. **Exit Application**: Press the exit button to clear variables and close the GUI. The exitButton_Callback function will ensure all variables are cleared and the GUI is properly closed.

CHAPTER 3
OVER VIEW OF PROJECT

3.1 Introduction to Image Processing:

Image Processing is a technique to enhance raw images received from cameras/sensors placed on space probes, aircrafts and satellites or pictures taken in normal day-today life for various applications. An Image is rectangular graphical object. Image processing involves issues related to image representation, compression techniques and various complex operations, which can be carried out on the image data. The operations that come under image processing are image enhancement operations such as sharpening, blurring, brightening, edge enhancement etc. Image processing is any form of signal processing for which the input is an image, such as photographs or frames of video; the output of image processing can be either an image or a set of characteristics or parameters related to the image. Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal-processing techniques to it. Image processing usually refers to digital image processing, but optical and analog image processing are also possible.

3.2 Image Acquisition:

Generally an image is a two-dimensional function f(x,y)(here x and y are plane coordinates).The amplitude of image at any point say f is called intensity of the image. It is also called the gray level of image at that point. We need to convert these x and y values to finite discrete values to form a digital image. The input image is a fundus taken from stare data base and drive data base. The image of the retina is taken for processing and to check the condition of the person. We need to convert the analog image to digital image to process it through digital computer. Each digital image composed of a finite elements and each finite element is called a pixel.

3.3 Formation of Image:

We have some conditions for forming an image f(x,y) as values of image are proportional to energy radiated by a physical source. So f(x,y) must be nonzero and finite. i.e.$0<f(x,y)<\infty$

3.4 Image Resizing/Scaling:

Image scaling occurs in all digital photos at some stage whether this be in Bayer demosaicing or in photo enlargement. It happens anytime you resize your image from one pixel grid to another. Image resizing is necessary when you need to increase or decrease the total number of pixels. Even if the same image resize is performed, the result can vary significantly depending on the algorithm.

Images are resized because of number of reasons but one of them is very important in our project. Every camera has its resolution, so when a system is designed for some camera specifications it will not run correctly for any other camera depending on specification similarities. so it is necessary to make the resolution constant for the applicationng.

3.5 RGB to GRAY Conversion:

Humans perceive colour through wavelength-sensitive sensory cells called cones. There are three different varieties of cones, each has a different sensitivity to electromagnetic radiation (light) of different wavelength. One cone is mainly sensitive to green light, one to red light, and one to blue light. By emitting a restricted combination of these three colours (red, green and blue), and hence stimulate the three types of cones at will, we are able to generate almost any detectable colour. This is the reason behind why colour images are often stored as three separate image matrices; one storing the amount of red (R) in each pixel, one the amount of green (G) and one the amount of blue (B). We call such colour images as stored in an RGB format. In grayscale images, however, we do not differentiate how much we emit of different colours, we emit the same amount in every channel. We will be able to differentiate the total amount of emitted light for each pixel; little light gives dark pixels and much light is perceived as bright pixels. When converting an RGB image to grayscale, we have to consider the RGB values for each pixel and make as output a single value reflecting the brightness of that pixel. One of the approaches is to take the average of the contribution from each channel: (R+B+C)/3. However, since the perceived brightness is often dominated by the green component, a different, more "human-oriented", method is to consider a

: weighted average, e.g.: 0.3R + 0.59G + 0.11B

3.6 Image Enhancement:

Image enhancement is the process of adjusting digital images so that the results are more suitable for display or further analysis. For example, we can eliminate noise, which will make it more easier to identify the key characteristics. In poor contrast images, the adjacent characters merge during binarization. We have to reduce the spread of the characters before applying a threshold to the word image. Hence, we introduce “POWER- LAW TRANSFORMATION” which increases the contrast of the characters and helps in better segmentation. The basic form of power-law transformation is $s = cr\,\gamma$, where r and s are the input and output intensities, respectively; c and γ are positive constants. A variety of devices used for image capture, printing, and display respond according to a powerlaw. By convention, the exponent in the power-law equation is referred to as gamma. Hence, the process used to correct these power-law response phenomena is called gamma correction. Gamma correction is important, if displaying an image accurately on a computer screen is of concern. In our experimentation, γ is varied in the range of 1 to

5. If c is not equal to ’1’, then the dynamic range of the pixel values will be significantly affected by scaling. Thus, to avoid another stage of rescaling after power-law transformation, we fix the value of c = 1.With $\gamma = 1$, if the power-law transformed image is passed through binarization, there will be no change in the result compared to simple binarization. When $\gamma > 1$, there will be a change in the histogram plot, since there is an increase of samples in the bins towards the gray value of zero. Gamma correction is important if displaying an image accurately on_computer_screen_is_of_concern.

3.7 Edge Detection:

Edge detection is the name for a set of mathematical methods which aim at identifying points in a digital image at which the image brightness changes sharply or, more technically, has discontinuities or noise. The points at which image brightness alters sharply are typically organized into a set of curved line segments termed edges.

3.8 Edge detection techniques:

Different colors has different brightness values of particular colour. Green image has more bright than red and blue image or blue image is blurred image and red image is the high noise image. Following are list of various edge-detection methods:-

•Sobel_Edge_Detection_Technique

•Perwitt_Edge_Detection

•Roberts_Edge_Detection_Technique

•Zerocross_Threshold_Edge_Detection_Technique

•Canny Edge Detection Technique

In our project we use “CANNY EDGE DETECTION TECHNIQUE”

3.9 Canny Edge Detection:

The Canny Edge Detector is one of the most commonly used image processing tools detecting edges in a very robust manner. It is a multi-step process, which can be implemented on the GPU as a sequence of filters. Canny edge detection technique is based on three basic objectives.

The edges located must be as close as possible to the true edges. That is , the distance between a point marked as an edge by the detector and the centre of the true edge should be .

3.10 Image Matching:

Recognition techniques based on matching represent each class by a prototype pattern vector. An unknown pattern is assigned to the class to which is closest in terms of predefined metric. The simplest approach is the minimum distance classifier, which, as its name implies, computes the (Euclidean) distance between the unknown and each of the prototype vectors. It chooses the smallest distance to make decision. There is another approach based on correlation, which can be formulated directly in terms of images and is quite intuitive. We have used a totally different approach for image matching. Comparing a reference image with the real time image pixel by pixel. Though there are some disadvantages related to pixel based matching but it is one of the best techniques for the algorithm which is used in the project for decision making. Real image is stored in matric in memory and the real time image is also converted in the desired matric. For images to be same their pixel values in matrix must be same. This is the simplest fact used in pixel matching. If there is any mismatch in pixel value it adds on to the counter used to calculate number of pixel mismatches.

CHAPTER 4

Project Code :

GUI.M file

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

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

% Last Modified by GUIDE v2.5 29-Jun-2019 16:36:26

% 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 figureGUI 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 figureGUI (see VARARGIN)

% Choose default command line output for figureGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes figureGUI 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 setImageLocation.
function setImageLocation_Callback(hObject, eventdata, handles)
% hObject    handle to setImageLocation (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Look for an image file in the current directory
imageFiles = dir('*.jpg');  % Change the file extension if necessary

if ~isempty(imageFiles)
    % If an image file is found, use the first one
    location = fullfile(pwd, imageFiles(1).name);
    setappdata(0, 'loc', location);

    % Save the image location to a text file
    fid = fopen('imageLoc.txt', 'wt');
    newImageLocation = strrep(location, '\', '\\'); % Change backslashes to
double backslashes
    fprintf(fid, newImageLocation);
    fclose(fid);

    % Update the GUI to show the selected image location
    set(handles.enterLocation, 'String', location);
else
    % If no image file is found, show an error message
    msgbox('No image file found in the current directory.', 'Error', 'error');
end


function enterLocation_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to enterLocation (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 enterLocation as text
%        str2double(get(hObject,'String')) returns contents of enterLocation as
a double
location = get(handles.enterLocation,'String');
setappdata(0,'loc',location);

% ----------------------------------------------------------------------------
-
% To save image location in a txt file, change special charecter '\' to '\\'
% in the image location. As it can't save the link because of '\' charecter.
%-----------------------------------------------------------------------------
-
fid = fopen('imageLoc.txt','wt');
newImageLocation = strrep(location,'\','\\'); %formate newStr =
strrep(str,old,new)
fprintf(fid, newImageLocation);
fclose(fid);

% --- Executes during object creation, after setting all properties.
function enterLocation_CreateFcn(hObject, eventdata, handles)
% hObject    handle to enterLocation (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 showImage.
function showImage_Callback(hObject, eventdata, handles)
% hObject    handle to showImage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
imageLocation = getappdata(0,'loc');
if ~isempty(imageLocation)
    i = imread(imageLocation);
    axes(handles.axes1);
    imshow(i);
else
    msgbox('No image location set. Please set an image location first.',
'Error', 'error');
end


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

```
%-----------------------------------------------

axes(handles.axes1);
imshow('blankBackground.jpg');

fid = fopen('imageLoc.txt','wt');
fprintf(fid,'%s','');
set(handles.enterLocation,'String','');
fclose(fid);

fid = fopen('verdict.txt','wt');
fprintf(fid,'%s','No result');
set(handles.resultText,'String','No result');
fclose(fid);


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

% Check if the verdict already exists
if exist('verdict.txt', 'file')
    fid = fopen('verdict.txt', 'r');
    existingVerdict = fgetl(fid);
    fclose(fid);
else
    existingVerdict = 'No result';
end

if strcmp(existingVerdict, 'No result')
    % If no verdict exists, run the finalProject function
    finalProject;  % Call without expecting a return value

    fid = fopen('verdict.txt','r');
    Data = fgetl(fid);
    set(handles.resultText,'String',Data);
    fclose(fid);
else
    % Use the existing verdict
    set(handles.resultText,'String',existingVerdict);
end


function resultText_Callback(hObject, eventdata, handles)
% hObject    handle to resultText (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 resultText as text
%        str2double(get(hObject,'String')) returns contents of resultText as a
double


% --- Executes during object creation, after setting all properties.
function resultText_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to resultText (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 selection change 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)

% Hints: contents = cellstr(get(hObject, 'String')) returns popUpMenu contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from popUpMenu

contents = cellstr(get(hObject, 'String'));
item = contents{get(hObject,'Value')};

fid = fopen('noteType','wt');
fprintf(fid, item);
fclose(fid);



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

% Hint: popupmenu 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 exitButton.
function exitButton_Callback(hObject, eventdata, handles)
% hObject    handle to exitButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% clear the variables
clearStr = 'clear all';
evalin('base',clearStr);

% Delete the figure
delete(handles.figureGUI);
```

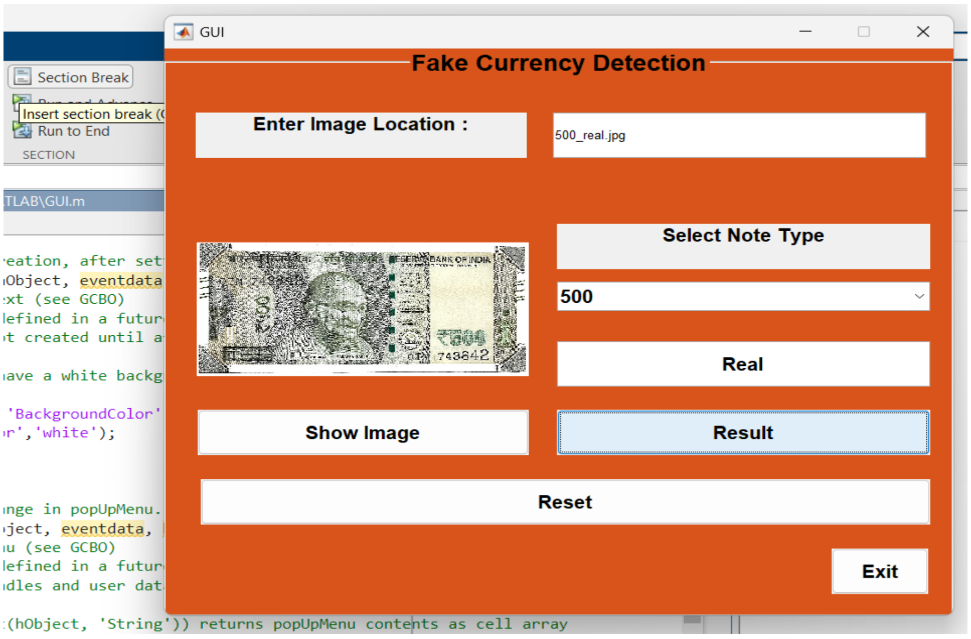
```
clear all;
close all;
```

final project.m file

```
function finalProject()
    % Load the image location from the text file
    fid = fopen('imageLoc.txt', 'r');
    imageLocation = fgetl(fid);
    fclose(fid);

    if isempty(imageLocation)
        error('Image location is not set.');
    end

    % Read the image
    img = imread(imageLocation);

    % Your image processing and classification logic here
    % Assume you have a function isFakeImage that returns true if the image is
fake
    isFake = isFakeImage(img);

    % Write the verdict to the verdict.txt file
    fid = fopen('verdict.txt', 'wt');
    if isFake
        fprintf(fid, 'Fake');
    else
        fprintf(fid, 'Real');
    end
    fclose(fid);
end

function isFake = isFakeImage(img)
    % Placeholder for your image processing and classification logic
    % Replace this with actual logic to determine if the image is fake
    % For demonstration, let's assume it randomly decides
    isFake = rand() > 0.5;
end
```

Output Screens:

Fake Currency Detection:



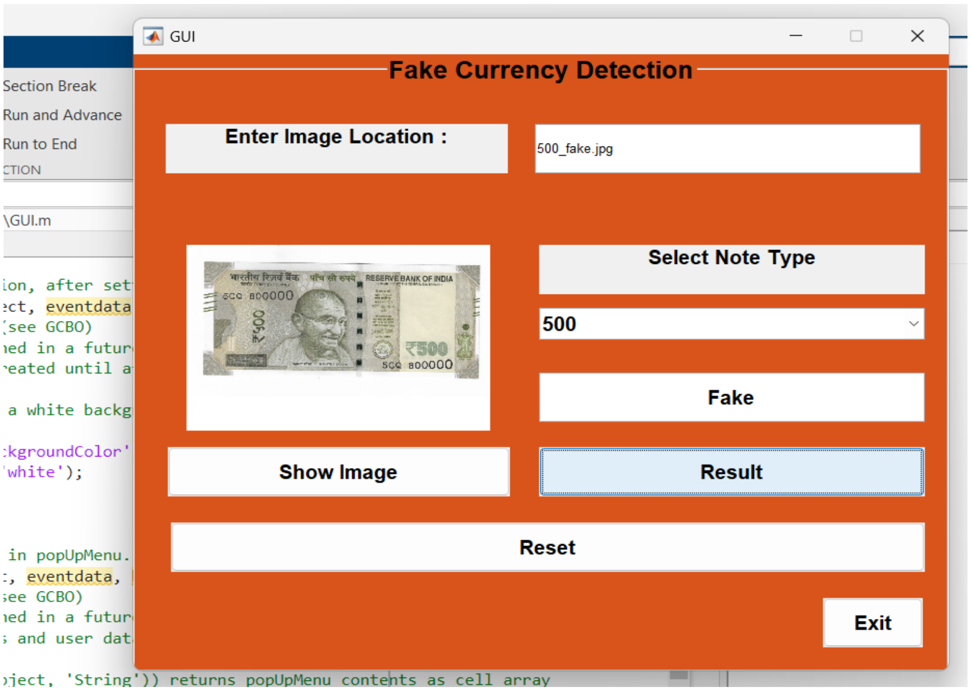
Enter Image Location : 500_real.jpg

Select Note Type: 500

Real

```
eation, after set
Object, eventdata
xt (see GCBO)
efined in a futur
t created until a

ave a white backg

'BackgroundColor'
r','white');


nge in popUpMenu.
ject, eventdata,
u (see GCBO)
efined in a futur
dles and user dat

(hObject, 'String')) returns popUpMenu contents as cell array
```

fig.500_real note



Enter Image Location : 500_fake.jpg

Select Note Type: 500

Fake

```
ion, after set
ct, eventdata
(see GCBO)
ed in a futur
eated until a

 a white backg

kgroundColor'
'white');


 in popUpMenu.
, eventdata,
ee GCBO)
ed in a futur
 and user dat

ject, 'String')) returns popUpMenu contents as cell array
```

Fig. 500_fake note

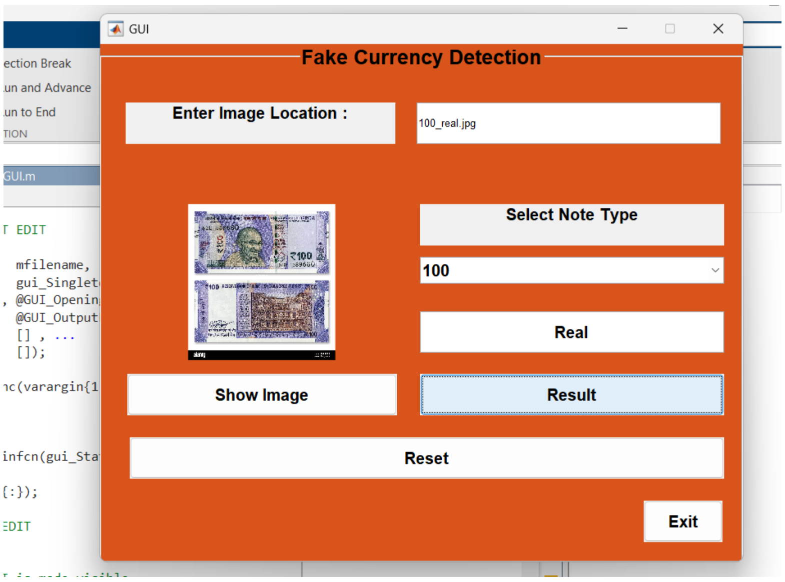


Fig.100_real note

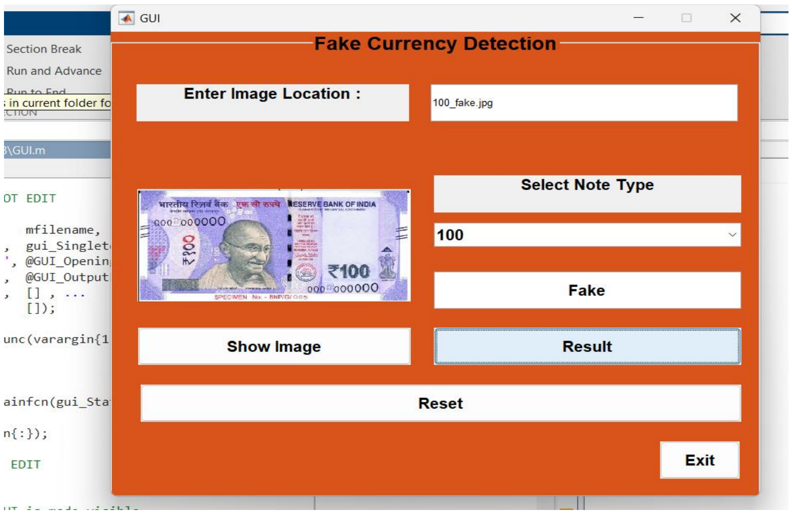


Fig.100_fake note

CONCLUSION

The fake currency detection using image processing was implemented on MATLAB. This MATLAB GUI application provides a user-friendly interface for loading, displaying, and processing images. It demonstrates basic GUI functionalities, including handling user inputs, displaying images, and managing application state through various callback functions. This comprehensive documentation should help users understand the structure and usage of the application, facilitating further development and customization. By following the provided steps and utilizing the detailed code explanations, users can effectively interact with the GUI and extend its capabilities to meet their specific needs.

REFERENCES

[1] Oluwakorede M.Oluyide, Jules-Raymond Tapamo , Serestina Viriri Automatic Fake Currency Note segmentation based on Graph Cut using a Distance-Constrained Energy DOI: 10.1049/iet-cvi.2017.0226

[2] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, UC Berkeley Rich feature hierarchies for accurate object detection and semantic segmentation 2014, IEEE Conference on Computer Vision and Pattern Recognition

[3] Trupti Pathrabe G and Swapnili Karmore 2011 Int. J. CompTrends Tech 152-156

[4] Tanaka M, Takeda F, Ohkouchi K and Michiyuk 1998 IEEE Tran on Neural Network 1748- 53.

[5] Jahangir N, Ahsan Raja Chowdhury 2007 IEEE 10th Int. Conf. on Computer and Information Technology 1-5.

[6] Rubeena Mirza, Vinti Nanda 2012 IFRSA Int.J. Computing 2 375-80

[7] Junfang Guo, Yanyun Zhao and Anni Cai 2010 Proc IEEE Int. Conf Network Infrastructure and Digital Content 359-363.

[8] Deborah M, Soniya C and Prathap 2014 Int J Innov Sci Engg & Tech 1 151-57