A PROJECT REPORT ON

FAKE CURRENCY DETECTION USING IMAGE PROCESSING AND MACHINE LEARNING

A

PROJECT REPORT

SUBMITTED TO

SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

in partial fulfillment for the Degree of

BACHLOER OF ENGINEERING

(INFORMATION TECHNOLOGY)

BY

Sharwari Gothe (B120398521)

Vrushal Joshi (B120398523)

Ajmal Khan (B120398524)

Kanchan Naik (B120398537)

GUIDED BY

Mrs. Priya Shelke

VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE

INFORMATION TECHNOLOGY DEPARTMENT

2018-19

DEPARTMENT OF INFORMATION TECHNOLOGY

Certificate

This is to certify that

Sharwari Gothe (B120398521)
Ajmal khan (B120398524)
Vrushal Joshi (B120398523)
Kanchan Naik (B120398537)

have successfully completed this project report entitled **"FAKE CURRENCY DETECTION"**, under my guidance in partial fulfillment of the requirements for the degree of Bachelor of Engineering, Department of Information Technology of Savitribai Phule Pune University during the academic year 2017-18.

Date:-
Place:-

Mrs. Priya Shelke	Mr. Narendra Pathak	Dr. B. S. Karkare
Project Guide	HOD,IT Department	Principal, VIIT, Pune

Examiners : 1
2.

Acknowledgement

It is our pleasure to develop this application and take a small initiative from our side for the betterment of common citizen of India and also in helping on a small scale in reducing the flow of fake currency notes in the market. We would like to thank our project guide Mrs. Priya Shelke in helping us from the scratch. It would have not been possible to achieve this success had it not been her who supported us in every possible way to reach this level in our project. We are grateful to Mr. Narendra Pathak, Head of Information Technology Department, VIIT for his support throughout. We would also like to thank the other teaching and non-teaching staff of Information Technology Department for the resources and guidance that we received from time to time. The VIIT, Library played a major role in helping us with the resources needed for the execution of the project. The results we have received today is an combined result of every individual of VIIT who have helped us in achieving this success.

Sharwari Gothe (B120398521)
Vrushal Joshi (B120398523)
Ajmal Khan (B120398524)
Kanchan Naik (B120398537)

ABSTRACT

Fake currency is imitation without the legal sanctions of the state and government. It is growing on a large scale as the technology is enhancing and detection of these notes is becoming difficult as the counterfeit process is improving day by day. With the development of modern banking services automatic method for fake currency detection recognition become important in many applications such as automated teller machines, automatic shopkeepers machine etc. Fake currency increase can cause reduction in value of real money and increase in prices as more money is being circulated in the economy. This is a loss on all fronts, economical as well as social. There are many methods of detecting fake currency bit. In this project we are using image processing method to detect fake currency. This project is an android application which will help people to identify the fake note in their day to day life. Method to detect fake note is to acquire the image of the note with the help of camera, pre-process the image, convert the image in gray scale image. Further processing will be done on this new image like edge detection, adaptive thresholding and feature extraction. Once the relevant features are extracted the note will be tested on the trained machine and the outcome will be displayed to the end user.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation and Social Impact

An article was published by Sharmad Mahajan not long ago in the year 2016 stating the statistics of the fake notes in the market. The numbers were unnerving in every sense. Due to the flow of such notes in the market the value of the money is going down tremendously. The inflation has risen and the corruption has no limit. This has led to the loss in social as well as economic sense. As a citizen of India we found it to be our duty to help reduce this in our own way. This was one the major motivation along with various other articles published on various other medias. This will reduce only when the people are made aware about it and every citizen takes an effort to educate himself about the ill effect of the fake currency circulating in the market. But to take an action after spreading awareness is to provide a means to bring it into action. We plan to provide this means to the citizens.

1.2 Need of Image Processing and Machine Learning

The currency notes that are taken under observation to verify the authenticity has to undergo many processes. These processes are important to increase the efficiency. The currency notes are captured through various angles and different qualities. They need to be brought in a standard dimensions and quality so as to bring stability to the application. Image processing helps in gaining these properties. Image processing is a general term for various operations that take place on any image. Image processing is a vast domain and has developed algorithm which give accurate results. Digital image processing has many advantages over analog image processing. It allows a wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and distortion during processing. Since images are defined over two dimensions digital image processing may be modeled in the form of multidimensional systems.

The machine is trained with well organized data set. Machine Learning helps to recognize the design pattern and analyze it. Machine Learning is the science of getting computers to act without being explicitly programmed. Machine Learning has given us self-driving cars, practical speech recognition,effective web search and a vastly improved understanding of the human genome. Machine Learning is so pervasive today that its been used probably dozens of times a day without knowing it. In this project the machine is trained with various notes of 100 rupees with some defects but are real. Also, the machine is trained with fake notes so that it will check the authenticity more accurately. Machine Learning is used because it is not possible to just give the dimensions and get the result. The technology has developed a lot in recent times and it is not possible for the naked eye to detect if the currency is real or fake. The machine needs to be perfectly trained to get the accurate results for which machine learning is required. Random Forest algorithm has been used in this project.

1.3 Goals and Objectives

The major goal and objective of this project is:

- To provide a means to the citizen of India to check the authenticity of the currency note
- To reduce the circulation of the currency note in the market at least on a small scale and help in reducing the inflation
- Our goal is to give maximum accurate result with minimal error rate

1.4 Statement of Scope and Outcomes

Developing an android application which will process and detect whether the currency note is authentic or duplicate using image processing technique. The camera will capture the image of currency note and further processing will be done by testing the respective currency note using the trained machine. This project is being developed for 100 rupee note however the scope can be further increased in the future by training the machine for other currency notes as well.

The outcome of this project is that it will help users to check the currency notes that they are handling on day to day basis and help them avoid getting deceived by the other people. It is easy to use and hence can be used even by a novice citizen.

1.5 Problem Statement

To develop an Android application to detect the authenticity of the currency notes being handled on day to day basis by the citizen of India.

Chapter 2

Literature Survey

2.1 Technologies available to cater the same services

We did the survey of certain systems which are used to check the authenticity of the currency notes. There are various mechanical ways through which banks, teller machines and huge organizations check the authenticity of the currency notes in bulk. However there are a very few, equivalent to null applications which helps the common man to detect the authenticity of the currency notes being handled by him.

2.1.1 Counterfeit Money Detector

Counterfeit money detector application as per the developers allows you to see the little marks and other security measures to probe a bill's authenticity under an UV light. Your cellphone screen isn't a UV lamp but we have to optimize it as much as possible. You may not be able to see anything in daylight so you have to rather so it in dark places.

Disadvantages of Counterfeit Money Detector:

- It shows you too many unwanted advertisements
- It forces you to watch those advertisements
- Accuracy is very less
- It needs UV light in phone

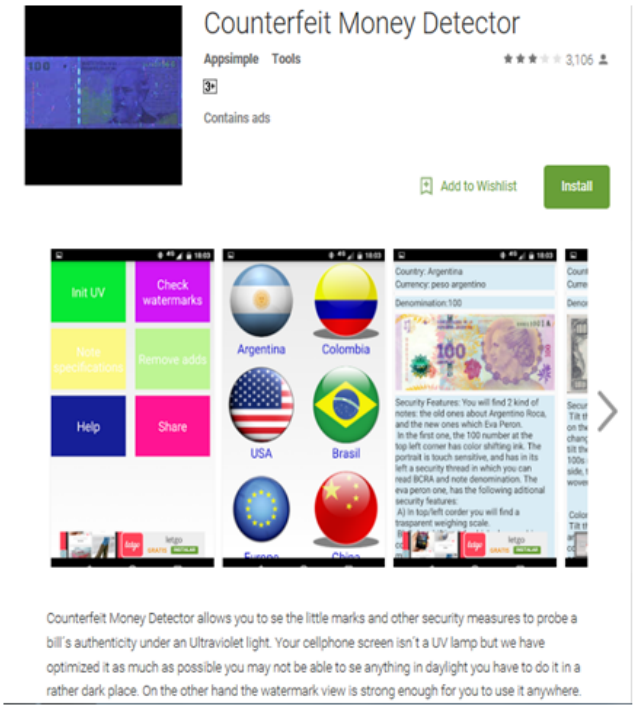

Figure 2.1: Existing Application

2.1.2 Real Indian Currency Detector

This application is more authentic than the previous. It does give results but its has too many flaws to consider using it.
Disadvantages of this application:

- Processing speed is very slow
- Results are not accurate
- Works for specific notes only

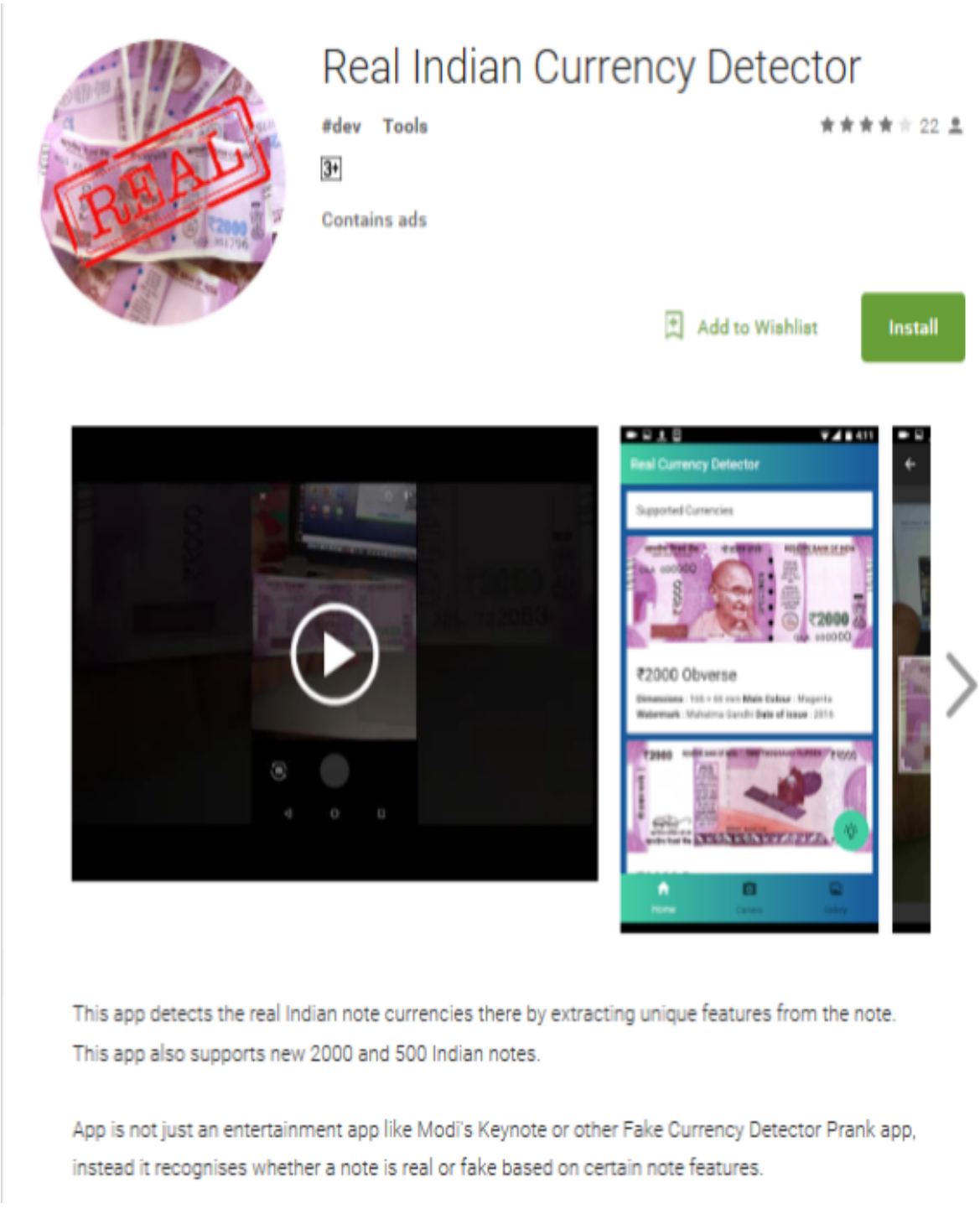

Figure 2.2: Real Indian Currency Detector

2.2 Review of Existing System

Thus we studied the above applications which detect fake notes. We noted all the drawbacks of each system and also noted the salient features and tried to make improvements in our application as follows: The Counterfeit Money detector requires UV light which is not kept as an constraint in our applications. Also our application works in daylight without any problems. The accuracy of the system is less compared to our application.

The Real Indian Currency Detector is very slow and is not feasible to use. It is one of the authentic applications but is not easily available on the Playstore. There are various issues with other applications as well. People have posted applications which are fake and are developed for malicious activities. No application is yet been developed which purely checks the authenticity without any ulterior motive. All the applications other than a selected few are a prank and deceive the users. Few applications help in detecting the features, however, they do not give the final results. The user has to himself figure out the final result. Hence, they are not completely automated. This reduces the efficiency on a large scale and is not very reliable.

Chapter 3

Project Statement

3.1 Purpose behind the Project

3.2 Technologies available to cater the same services

Today the technology has developed on a large scale. A lot of development in science and technology has led to growth in various sectors. It has helped in makings human life easy and fast. The benefits of this growth can be seen in every stream, every sector and it is highly appreciable as to how it has changed the way of life drastically. However, along with the pros there are various cons which have hampered the security and peace of mind. People are not using this technology only for good use and because of this a lot of malicious activities have taken place. People use technology to deceive people on regular basis and novice ignorant people become the victim.

Fake Currency Detector application is a small initiative to help common citizen of India to have a better understanding about the currency which is being used on day today basis and not get deceived easily. Due to advancement in technology the amount of fake notes that are being circulated in the market has increased on a large scale. These currency notes are extremely difficult to detect with naked eye. They look very identical to the authentic original notes and people get deceived very easily. With this application people will be able to get help on a small scale to avoid this circulation of notes and harming our economy drastically.

3.3 Reason for Android Application

The available applications in the market are mostly fake. They are just some malicious applications which do not give the result. A few of them are authentic however a lot of issues have been encountered with those application. These applications are slow and heavy. Thus they take time to give the result making them less efficient. Some applications are not

automated enough and hence the end user has to interpret the results themselves which again reduces the efficiency of the application. To overcome all these issues this application is being developed. Android phones have reached a large number of people. Around 6.4 billion people use android cell phones and hence the decision of using Android OS was taken. The major reason being that this would help us in reaching more and more people and increasing the usage of this application.

3.4 Market Survey

A proper market survey was done to see the behavior of people and preferences in case of technology related to the application. After a detailed survey the features of the applications were decided which would be convenient for maximum number of users. Also, the overall utility of the peoject was predicted with the help of this survey.

3.5 Uniqueness in the Project

As we have done the survey we came to know that there is no good option for common people in the form of application to help them detect the fake currency notes. Only a few organizations can check the authenticity and novice user is deceived on a regular basis. The applications which are developed before this are mostly fake or some sort of prank. A few authentic applications are extremely slow and do not give proper outcomes. Their reliability is very less. Our application is unique because the processing time is less compared to other applications. Also, we have not kept the constraint of UV light or night time as the criteria for the application to work. We have tried to give the best accuracy rate in our application.

3.6 Methodology

Fake currency is imitation of the authentic currency without any legal sanctions of the state or government. Counterfeit has increased on a large scale with the enhancement of technology and hence detection has become more difficult.

- This project looks forward to provide a facility to check the authenticity of the currency being handled on day to day basis by citizen of India.
- We are developing an android application which will process and detect whether the currency note is authentic or duplicate using image processing technique.
- The camera will click the image of the currency note and further processing will be done by testing the respective currency note using the trained machine.
- Our aim is to give maximum accurate result with minimal error rate.

Methods use in each step:

- Pre-processing
  - image adjustment
  - image smoothing – Median Filter
- Gray Scale conversion
- Edge Detection – Sobel Operator
- Segmentation – Edge Based Segmentation
- Feature Extraction

A fake currency detection stores the following information as shown below:

- Currency Database: It includes collection of various scanned denominations.
- Comparison Parameters (features): It includes various parameters (features) which should be checked to determine whether given sample is fake or not.

Level 1	**Level 2**	**Level 3**
Substrate Fidelity	Micro-text	Magnetic Ink
Print Fidelity	UV Glowing Ink	Screen Traps
Color Fidelity		Manufacture Anomalies
Acoustic Fidelity		Materials Interaction
Serial Number		Complicated Patterns
Holograms		Complicated Design
Watermark		Fluorescence
Security Thread		Texture Analysis
Security Fiber		Security Fiber
Color-Shifting Ink		Clear Window
Matching Sides		Latent Image

Features used in System:

- See Through Register
- Water Mark
- Security Thread
- RBI Mark

- Identification Mark
- Serial Number Extraction
- Governors Signature

3.7 Implementation

The approach in this project is to use a combination of image processing, and classification techniques to identify banknote denomination by currency features, and texture features. Banknote recognition is possible by extracting compatible features for computing, describing colour and the texture of acquired banknotes then comparing against the database of learned images. Once the banknote has been identified and classified, a similarity measurement is calculated and output is generated by side-by-side comparison. To test the algorithm a number of comparisons will be carried out to identify the most suitable feature vector, classifier combination.

1. **Acquisition:**

   Image is acquired by digital camera by applying the white back lighting against the paper currency so that the hidden attributes will appear on the image of the currency note.

2. **Pre-processing:**

   Pre-processing of image are those operations that are normally required prior to the main data analysis and extraction of information. Here image re-sizing is performed because the currency image is too large to process.

   - **Image Adjustment:**
     Re-size Function - The image is adjusted as per the required dimensions of the the application.
   - **Smoothing:**
     MedianBlur- The Function cv2.medianBlur() takes median of all the pixels under kernel area and central elements is replaced with this median value. This is highly effective against salt-and -pepper noise in the images. Interesting thing is that in the above filters, central element is a newly calculated value which may be a pixel value is the image or a new value. But in the median blurring, central element is always replaced by some pixel value in the image. It reduces the noise.

3. **Gray Scale Conversion:**

The image acquired is in RGB colour format. It is converted into gray scale because it carries only the intensity information which is easy to process instead of processing three components R(Red),G(Green), B(Blue).
textbfcvtcolor - It is an OpenCV function which converts from one color space to other.

4. **Edge Detection:**

Edge detection is one of the most frequently used techniques in digital image processing. The boundaries of object surfaces in a scene often lead to oriented localized changes in intensity of an image, called edges. This observation combined with a commonly held belief that edge detection is the first step in image segmentation, has fueled a long search for a good edge detection algorithm to use in image processing.

Adaptive Threshoding

In simple thresholding we use a global value as threshold value. But it may not be good in all the conditions where image has different lighting conditions in different areas. In that case, we go for adaptive thresholding. In this, the algorithm calculate the threshold for a small regions of the image. So we get different thresholds for different regions of the same image and it gives us better results for images with varying illumination.

It has three 'special' input parameters and only one output argument.

(a) **Adaptive Method** - It decides how thresholding value is calculated.
   - **cv.ADAPTIVE_THRESH_MEAN_C** : threshold value is the mean of neighbourhood area.
   - **cv.ADAPTIVE_THRESH_GAUSSIAN_C** : threshold value is the weighted sum of neighbourhood values where weights are a gaussian window.

(b) **Block Size** - It decides the size of neighbourhood area.

(c) **C**- It is just a constant which is subtracted from the mean or weighted mean calculated.

5. **Segmentation:**

Segmentation is the process of partitioning a digital image into multiple segments. It is typically used to distinguish objects from backgrounds. Here edge based segmentation is performed on the image.

6. **Decision Making:**

Finally decision is made by using following machine learning algorithm:

Random Forest:

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees habit of over fitting to their training set.

Random forest is like bootstrapping algorithm with Decision tree (CART) model. Say, we have 1000 observation in the complete population with 10 variables. Random forest tries to build multiple CART model with different sample and different initial variables. For instance, it will take a random sample of 100 observation and 5 randomly chosen initial variables to build a CART model. It will repeat the process (say) 10 times and then make a final prediction on each observation. Final prediction is a function of each prediction. This final prediction can simply be the mean of each prediction.

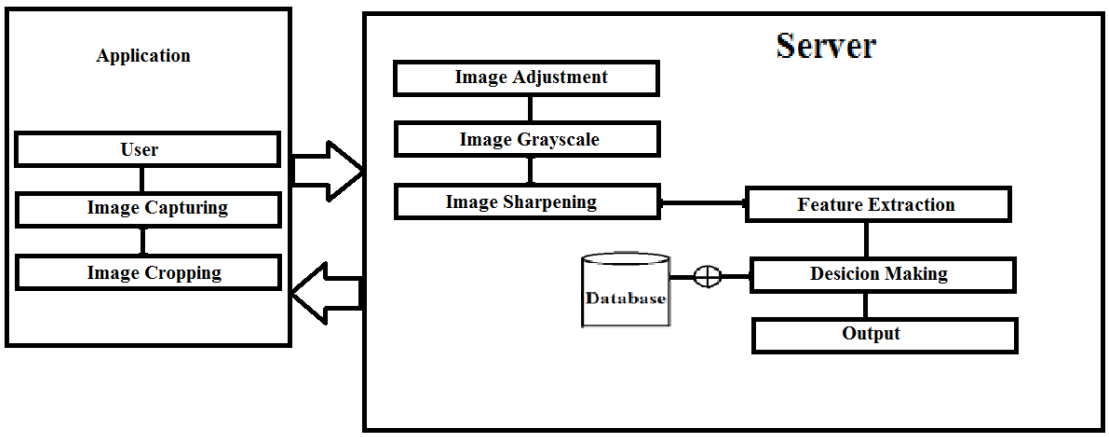

Figure 3.1: Block Diagram of proposed system

Chapter 4

System Requirement and Specification

4.1 Software requirements specifications

4.1.1 Introduction

Intended Audience and Reading Suggestions:
The intended audience includes all stakeholders as mentioned below:

- User: user should have basic knowledge of using android device.User will scan the image and ask for currency detection.
- Operating Environment
- This application will work on Android operating system, required android kitkat version or next version of Android till date.

4.1.2 Design and Implementation Constraints

- Server client connectivity.
- Quality of image should be maintained

4.1.3 Operating Environment

This is an android application and works on smart phones which have android operating system in it.

4.1.4 Assumptions and Dependencies

- Camera resolution should be good.
- Good lightning condition
- All user should have basic knowledge of using Android device.
- Sample to be tested might be damaged or old so 30-40 % error rate is allowed

4.2 External Interface Requirements

4.2.1 User Interfaces

- Front End : Android
- Back End : Machine Learning, Image Processing

4.2.2 Hardware Interfaces

- Android Device
- Inbuilt Device Camera: 16mp.

4.2.3 Network Interface

- Internet Connectivity
- Server

4.2.4 Software Interface

4.2.4.1 External Interface

Figure 4.1: External interface diagram

4.2.4.2 Internal Interface

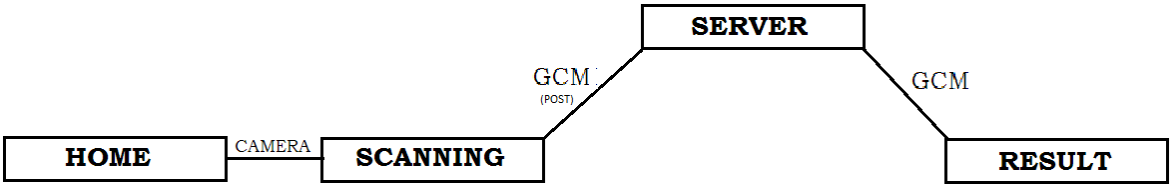

Figure 4.2: Internal Interface Diagram

4.3 Other Nonfunctional Requirements

4.3.1 Performance Requirements

Output time should be as low as possible. System should process the image and make the decision in optimal time. To improve the performance all the processing will be done on remote server using cloud technology

4.3.2 Accessibility

All the android device users should be able to use the application when the Internet connectivity is available. Server should be made available all the time in order to process the requests.

4.3.3 Efficiency

As application will run on small embedded mobile device memory consumed by the application should be as low as possible. Also processor utilization should be optimum.

4.3.4 Failure Management

If image is not captured properly then user will be given option of retaking the image. Also while processing the image it is noticed that image quality is poor then application will request user to retake the image.

4.3.5 Fault Tolerance

- 30%-40% of approximately error rate is allowed as currency note might be damaged or old one.

4.3.6 Other Requirements

- Old currency notes (Indian)
- Damaged currency notes (Indian)
- New currency notes (Indian)

4.4 System Specifications

4.4.1 Image Acquisition

Use Case Name	Image Acquisition
Trigger	The user will access application and click on capture image button
pre-recondition	The First page of application is opened correctly
Basic Path	The user will open the application Click on capture image button. Camera will be opened. Set the camera properly on currency note and click on capture button of camera.
Alternative Paths	None
Post-condition	Image is captured properly.
Exception Paths	The user may abandon the capture at any time.

4.4.2 Image Pre-processing

Use Case Name Trigger	image Pre-Processing The user click on detect method.
Precondition	The image of currency note is captured.
Basic Path	The image is adjusted in required dimensions. (using various interpolation methods). Noise from image is removed using median filter technique. In median filter image is scanned signal by signal then median of neighboring signal is taken. Original signal is then replaced with median.
Alternative Paths	None
Post-condition	Noise free image in required dimensions is obtained.
Exception Paths	None

4.4.3 Gray Scale Conversion

Use Case Name	Gray Scale Conversion
Trigger	Noise free image in require dimension is obtained
Precondition	Noise free image in require dimensions.
Basic Path	Input image is fetch. Gray scale conversion algorithm is implemented on input image. Gray image is obtained.
Alternative Paths	None
Post-condition	Gray image is obtained.

Edge Detection

Use Case Name	Edge Detection
Trigger	Gray image is obtained.
Precondition	Proper gray image is obtained
Basic Path	Gray scale image then subjected to edge detection algorithm (Adaptive Thresholding). Algorithm will find the threshold value for gray level intensity and edge will be detected.
Alternative Paths	None
Post-condition	Edges of objects are identified.

4.4.4 Segmentation

Use Case Name	Segmentation
Trigger	Image with proper edges is obtained.
Precondition	Edge detection should be done properly.
Basic Path	The image is subjected to segmentation algorithm. Various segments are separated from image. Features are extracted based on segments.
Alternative Paths	None
Post-condition	Various segments of image are obtained.

4.4.4.1 Decision Making

Use Case Name	Decision Making
Trigger	Segments of image are obtained.
Precondition	Segmentation and feature extraction should be done properly.
Basic Path	Extracted features are tested on trained detector. Detector will classify the features based on test cases. Threshold is checked. Decision will be taken whether the note is legitimate or not.
Alternative Paths	None
Post-condition	Decision is made for given currency note whether it is legitimate or not.

4.5 Technologies Used

- Android:

Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touch screen mobile devices such as smart phones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touch screen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on notebooks, game consoles, digital cameras, and other electronics. Android has the largest installed base of all operating systems of any kind. Android has been the bestselling OS on tablets since 2013, and on smart phones it is dominant by any metric. Initially developed by Android, Inc., which Google bought in 2005, Android was unveiled in 2007 along with the founding of the Open Handset Alliance a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices.

4.6 Libraries Used

- OpenCv:

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of- the-art computer vision and machine learning algorithms.

It has C++, C, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS.

Chapter 5

Project Analysis and Design

5.1 Diagrams

5.1.1 Use Case Diagram(Fuctional)

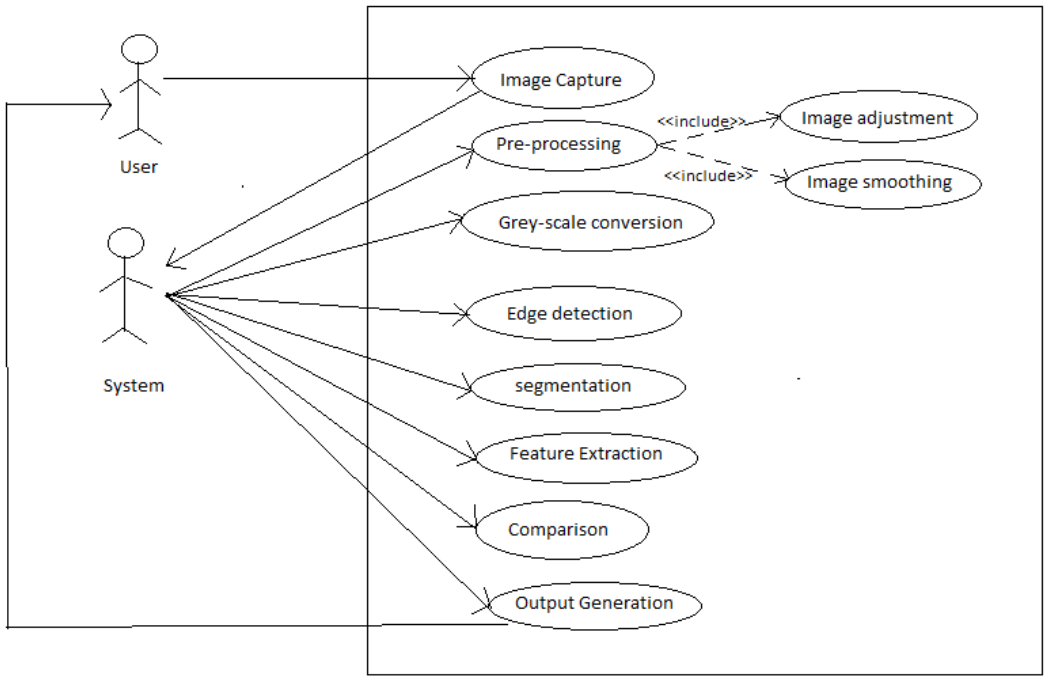

Figure 5.1: Use Case Diagram of Proposed System

5.1.2 Class Diagram

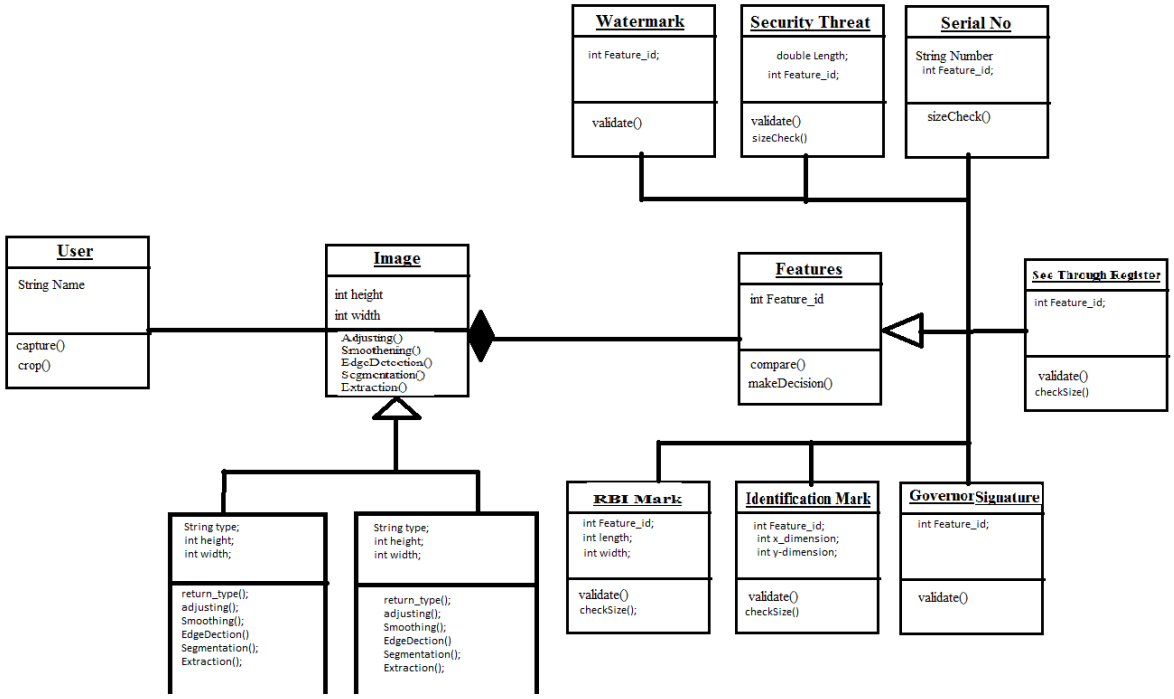

Figure 5.2: Class Diagram of Proposed System

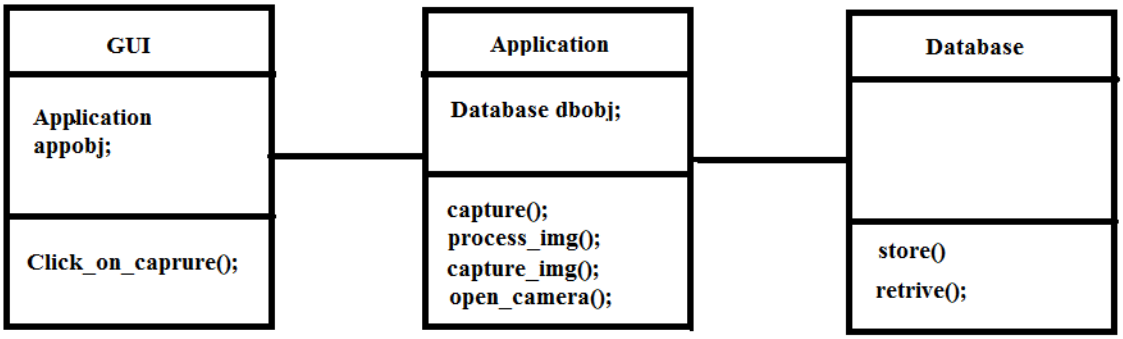

Figure 5.3: Advance Class Diagram of Proposed System

5.1.3 Activity and Sequence Diagram

5.1.3.1 Image Capture

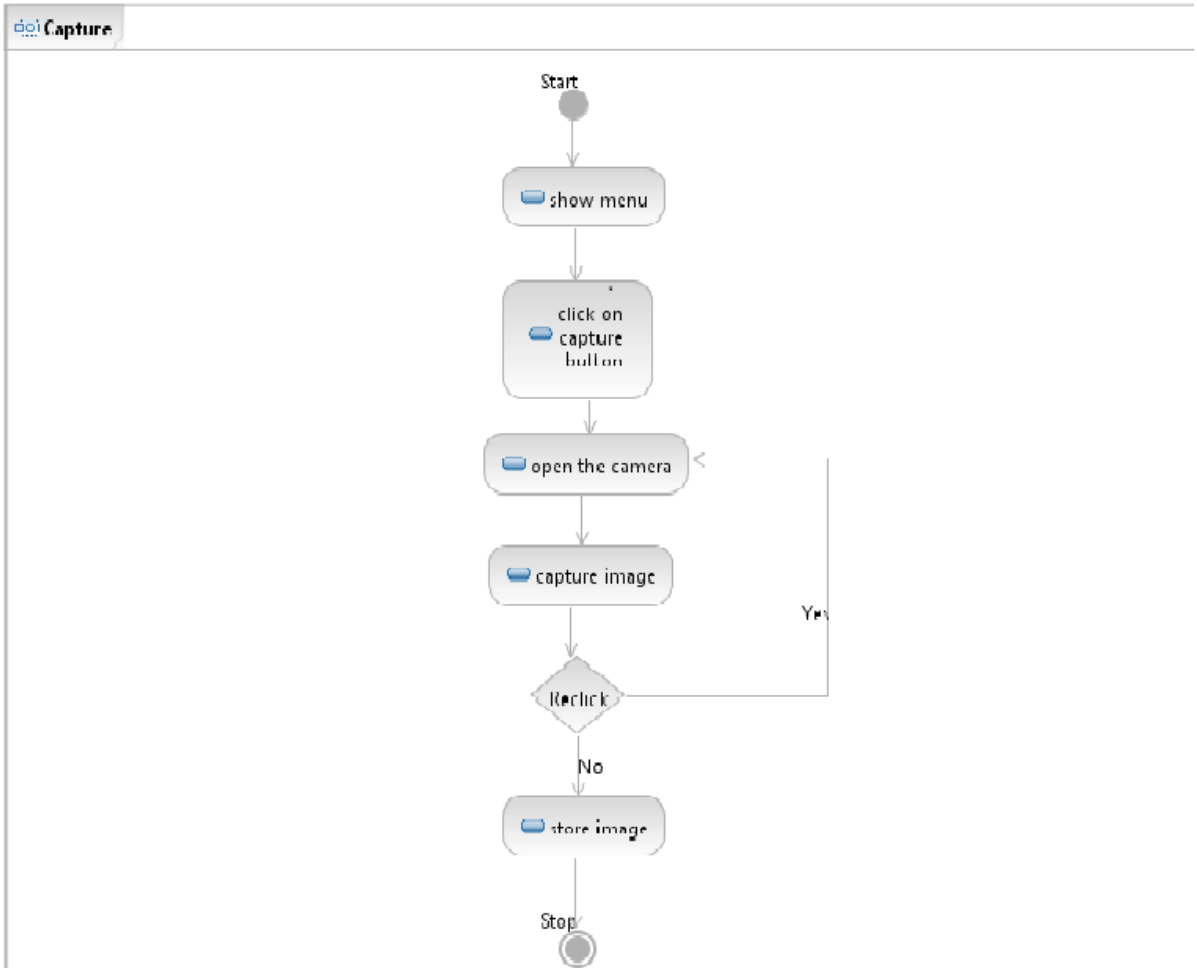

Figure 5.4: Image Capture Activity Diagram

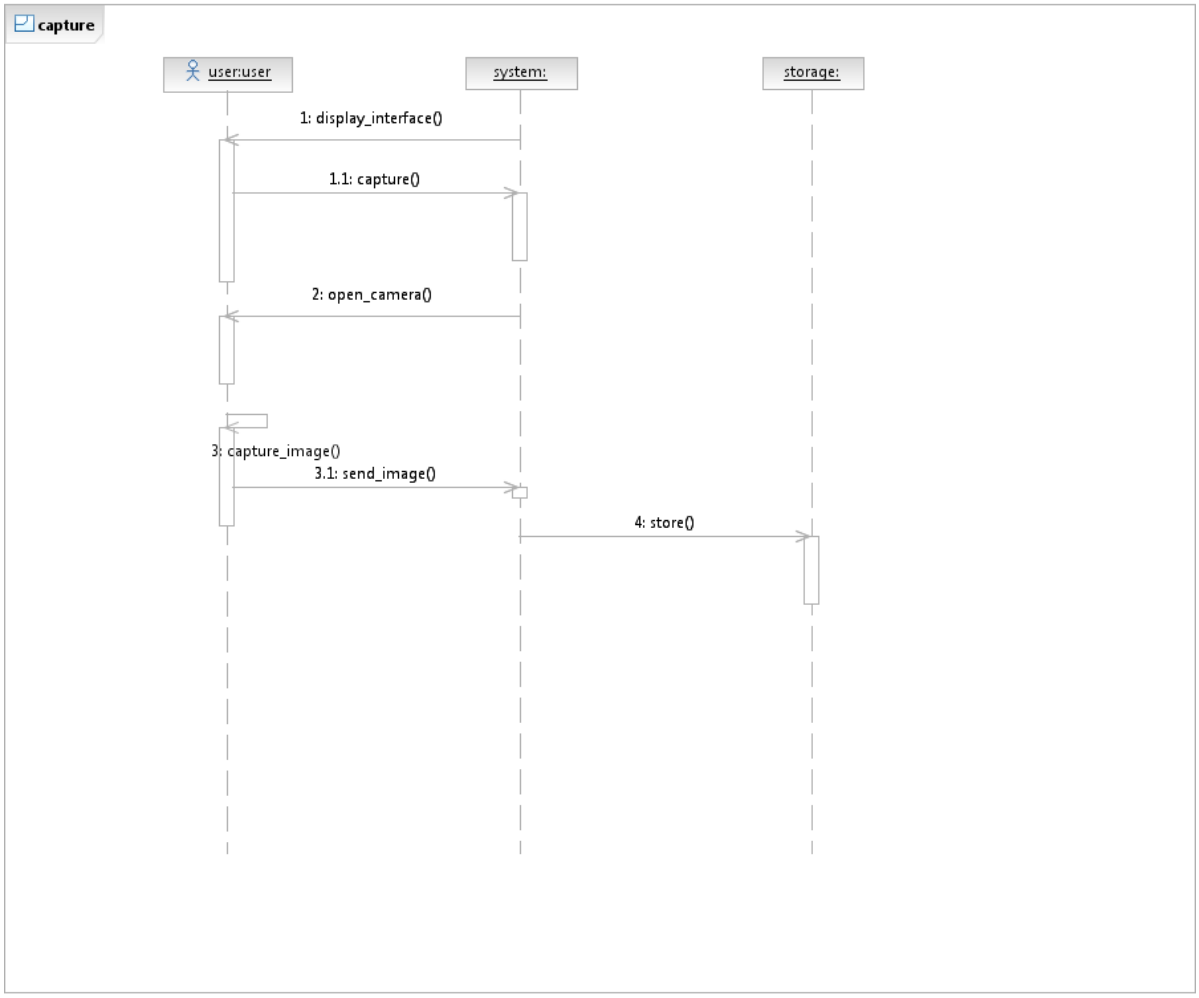

Figure 5.5: Image Capture Sequence Diagram

5.1.3.2 Pre-processing

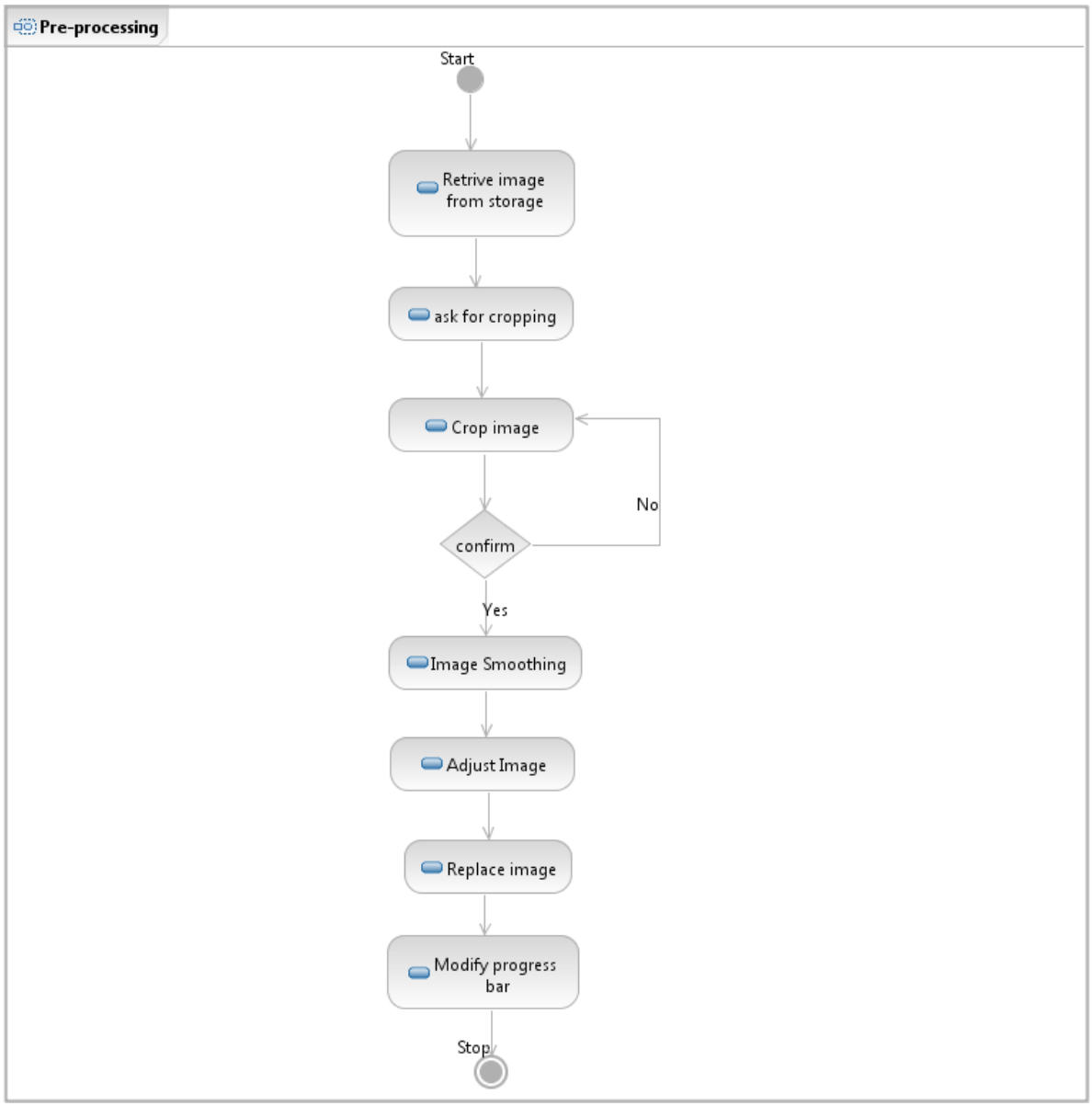

Figure 5.6: Image Pre-processing Activity Diagram

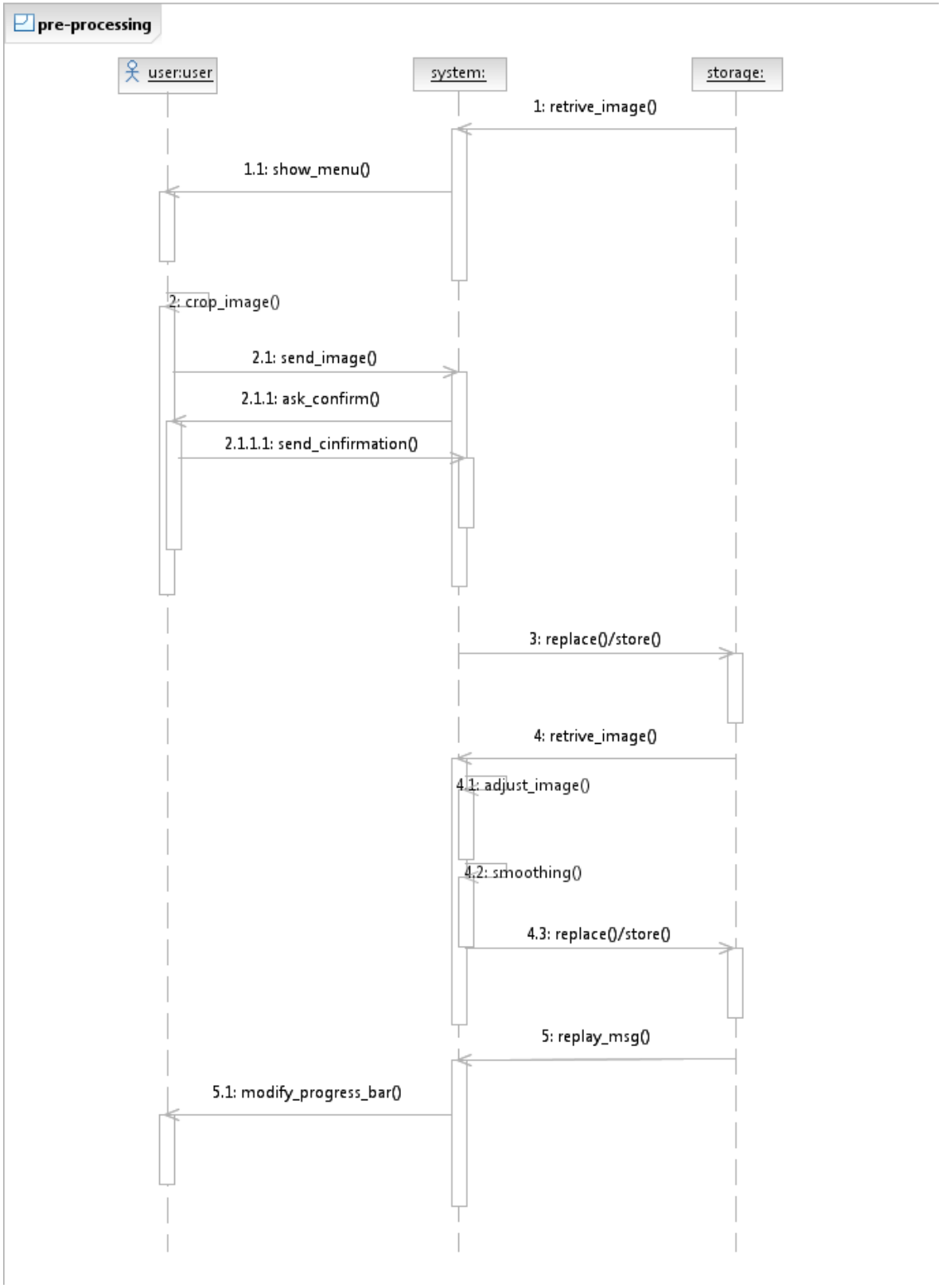

Figure 5.7: Image Pre-processing Sequence Diagram

5.1.3.3 Gray Scale

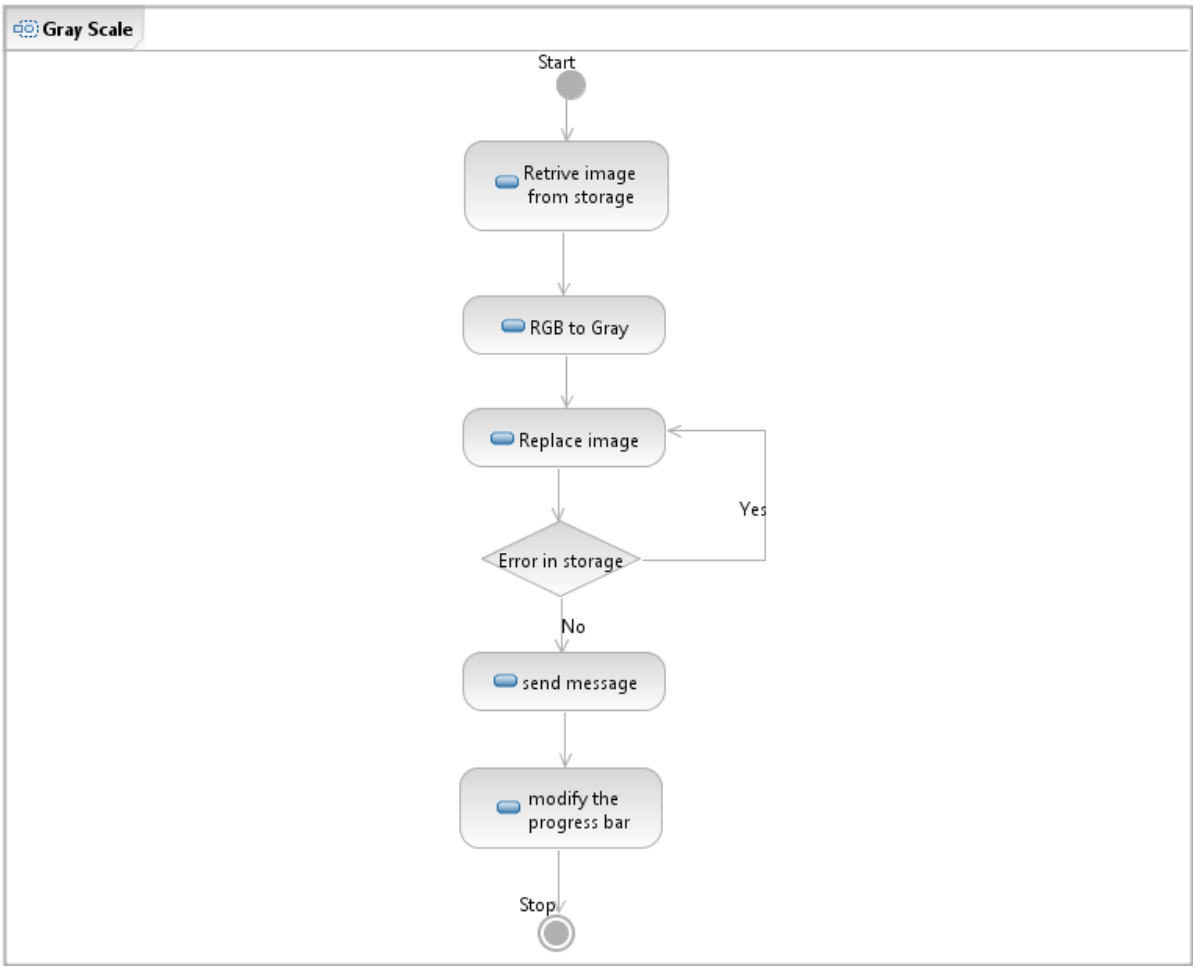

Figure 5.8: Gray Scale Conversion Activity Diagram

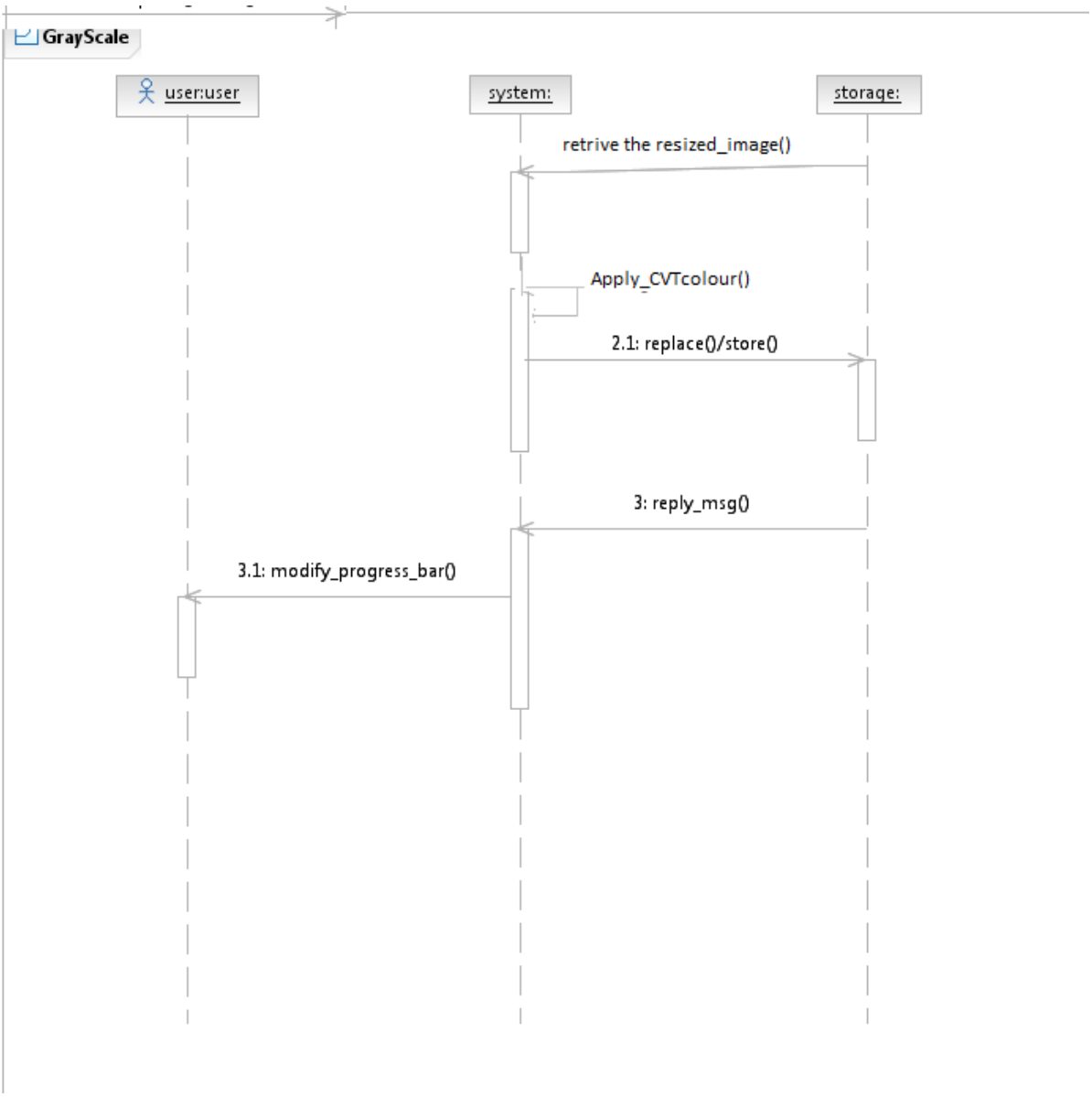

Figure 5.9: Gray Scale Conversion Sequence Diagram

5.1.4 Deployment Diagram

5.1.5 System Architecture

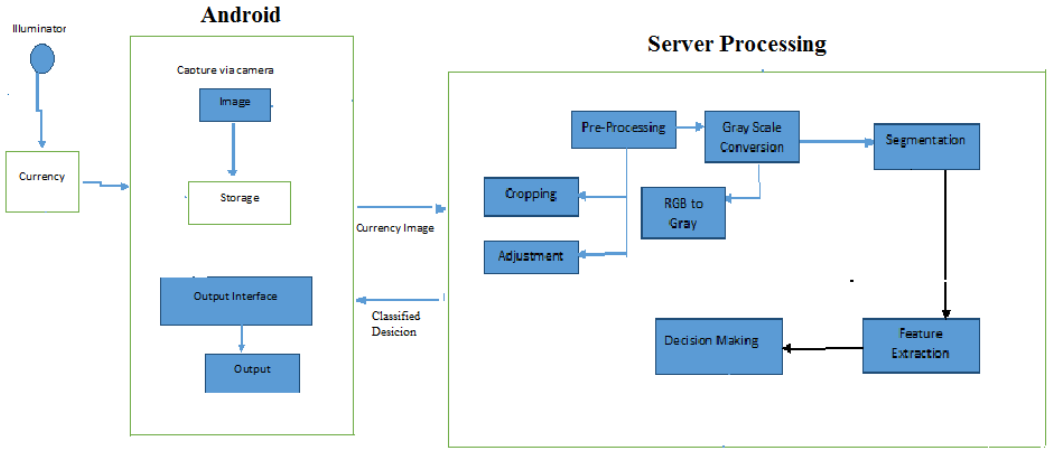

Figure 5.10: Architecture Diagram of Proposed System

5.2 Planning

Activity	Jun 3	Jun 10	Jun 17	Jun 24	Jul 14	Jul 21	Jul 27
Understanding Field Of Research							
Idea Identification							
Research And Processes Related To Idea Implementation							

Activity	Aug 4	Aug 9	Aug 14	Aug 19	Aug 22	Sept 1
Study Of IP Techniques & Background						
Study Of Various Feature Extraction Methods						
Comparison Of Various Feature Extraction Methods						

Activity	Sept 4	Sept 9	Sept 14	Sept 19	Sept 22	Sept 29	Dec 10
Inference & Decision Of Implementation Technique							
Prepare Different UML Diagrams							
Implementation Of input prepration							

Activity	Jan 4	Jan 12	Jan 19	Jan 26	Feb 2	Feb 9
Study Of Classifiers & Its Methods Of Implementation						
Implement Necessary Classifiers						

Activity	Feb 2	Feb 9	Feb 16	Feb 23	Mar 16	Mar 23	Mar 30
Test The Classifier							
Application & Test For Sample Set Test							
Performance For Necessary Set							
Try & Understand Android App							

Activity	Apr 6	Apr 29	May 4	May 11	May 18	May 25	May 30
Study For N/W Issues & Implementation							
Prepare Deployable Model							
Test The Model							
Submit Model							

Chapter 6

Testing

6.1 Introduction

This document is a high-level overview defining our testing strategy for the android mobile application. Its objective is to communicate project-wide quality standards and procedures. It portrays a snapshot of the project as of the end of the planning phase. This document will address the different standards that will apply to the unit, integration and system testing of the specified application. We will utilize testing criteria under the white box, black box, and system-testing paradigm. This paradigm will include, but is not limited to, the testing criteria, methods, and test cases of the overall design.

The definition of testing is ”the process of questioning a product in order to evaluate it”, where the ”questions” are things the tester tries to do with the product, and the product answers with its behavior in reaction to the probing of the tester. Although most of the intellectual processes of testing are nearly identical to that of review or inspection, the word testing is connoted to mean the dynamic analysis of the product putting the product through its paces. The quality of the application can and normally does vary widely from system to system but some of the common quality attributes include reliability, stability, portability, maintainability and usability. Refer to the ISO standard ISO 9126 for a more complete list of attributes and criteria.

Unit Testing

Unit testing, also known as component testing, refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors. These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple

tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to ensure that the building blocks of the software work independently from each other. Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Rather than replace traditional QA focuses, it augments it. Unit testing aims to eliminate construction errors before code is promoted to QA; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development and QA process.

Black Box Testing

Strategies for Black Box Testing Ideally, wed like to test every possible thing that can be done with our program. But, as we said, writing and executing test cases is expensive. We want to make sure that we definitely write test cases for the kinds of things that the customer will do most often or even fairly often. Our objective is to find as many defects as possible in as few test cases as possible. To accomplish this objective, we use some strategies that will be discussed in this subsection. We want to avoid writing redundant test cases that wont tell us anything new (because they have similar conditions to other test cases we already wrote). Each test case should probe a different mode of failure. We also want to design the simplest test cases that could possibly reveal this mode of failure test cases themselves can be error-prone if we don't keep this in mind.

Integration Testing

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed. Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

System Testing

System testing, or end-to-end testing, tests a completely integrated system to verify that it meets its requirements. For example, a system test might involve testing a logon interface, then creating and editing an entry, plus sending or printing results, followed by summary processing

or deletion (or archiving) of entries, then log off. In addition, the software testing should ensure that the program, as well as working as expected, does not also destroy or partially corrupt its operating environment or cause other processes within that environment to become inoperative (this includes not corrupting shared memory, not consuming or locking up excessive resources and leaving any parallel processes unharmed by its presence).

White-box testing

At last the system is delivered to the user for Acceptance testing. White-box testing (also known as clear box testing, glass box testing, transparent box testing and structural testing) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT). While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

6.1.1 Purpose

This document is intended to meet the following objectives:

- The strategy, responsibilities and schedule for the overall testing
- Identify the project and software artifacts that should be tested.
- List the scope of testing.
- List the deliverables of the test phases.
- List the set-up required.
- Identify the tasks and assumptions in the project, if any.
- Define the evolution criteria.

6.1.2 Process Overview

The following represents the overall flow of the testing process:

1. Identify the requirements to be tested. All test cases shall be derived using the current Program Specification.

2. Identify which particular test. will be used to test each module.
3. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.
4. Identify the expected results for each test.
5. Document the test case configuration, test data, and expected results.
6. Perform the test.
7. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the Unit/System Test Report (STR).
8. Successful unit testing is required before the unit is eligible for component integration/system testing.
9. Unsuccessful testing requires a Bug Report Form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.
10. Test documents and reports shall be submitted. Any specifications to be re-viewed, revised, or updated shall be handled immediately.

6.1.3 Testcases

Test case no.	Test case name	Date	Action	Expected Result	Actual Result	Status (P/F)
1	Camera 01	5/1/18	Click on camera button	Camera should start	Camera does not start	Fail
2	Camera 02	8/1/2018	Click on camera button	Camera should start	Camera started	Pass
3	Gallery	19/1/18	Click on gallery	Phone gallery should start	Phone gallery visible	Pass
4	Crop Image	4/2/18	Select/Adjust part of image	Multidirectional cropping	Cropping done	Pass
5	Save Image	14/2/18	Press save button	Image should be displayed on application	Image displayed on application	Pass
6	Send Image 01	23/2/18	Click on send image button	Image sent to server and display sent message	Image not sent to server	Fail
7	Send Image 02	3/3/18	Click on send image button	Image sent to server and display sent message	Image sent to server and displayed sent message	Pass
8	Display Result	14/3/18	-	Result should be displayed according to analysis	Result displayed on applications	Pass

Figure 6.1: Existing Application

Chapter 7

Conclusion and Future Work

This project will help the common citizen of India to check and validate the authenticity of the currency note that he/she would handle. This will also help in reducing the flow of fake currency in the market. Fake currency leads to inflation and economic lose on a large scale. Hence, it is very important to take measures to avoid the flow of fake currency in the market leading to betterment of the society, financially and ethically. This application will help people to not get deceived. Also, it is a small initiative to reduce the flow of fake notes in the market in turn helping the economy along with its people.

This project can be further applied to other currency notes as well. Further research could be done to find the to increase the efficiency by increasing the processing speed of the application.

Appendix A

Code

A.1 Source Code

A.1.1 Accepting Connection

```
public class MultithreadedSocketServer

    public static void main(String[] args) throws Exception
// TODO Auto-generated method stub

    try
ServerSocket server=new ServerSocket(5010);
int counter=0;
System.out.println("Server Started ....");
while(true)
counter++;
Socket serverClient=server.accept(); //server
accept the client connection request
System.out.println(" » " + "Client No:" + counter +
" started!");
ServerClientThread sct = new
ServerClientThread(serverClient,counter);
sct.start();

catch(Exception e)
System.out.println(e);
```

A.1.2 Main Application Thread

```
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Iterator;

    import javax.imageio.ImageIO;
import javax.imageio.ImageReadParam;
import javax.imageio.ImageReader;
import javax.imageio.stream.ImageInputStream;
import javax.xml.ws.handler.MessageContext;

    import org.omg.CORBA_2_3.portable.InputStream;
import org.opencv.imgcodecs.Imgcodecs;

    public class ServerClientThread extends Thread

    Socket serverClient;
int clientNo;

    ServerClientThread(Socket inSocket,int counter)
serverClient = inSocket;
clientNo=counter;


    public void run()
String clientMessage="",servermessage="";
try
DataOutputStream outStream = new
```

```
DataOutputStream(serverClient.getOutputStream());
ByteArrayOutputStream bos = new
ByteArrayOutputStream();
DataInputStream dIn = new
DataInputStream(serverClient.getInputStream());
int length = dIn.readInt(); //
read length of incoming message
byte[] message = new byte[length];
if(length>0)
dIn.readFully(message, 0, message.length); //
read the message


    bos.write(message, 0, message.length);
byte[] bytes = bos.toByteArray();

    ByteArrayInputStream bis = new
ByteArrayInputStream(bytes);
Iterator<?> readers =
ImageIO.getImageReadersByFormatName("jpg");

    ImageReader reader = (ImageReader) readers.next();
Object source = bis;
ImageInputStream iis =
ImageIO.createImageInputStream(source);
reader.setInput(iis, true);
ImageReadParam param =
reader.getDefaultReadParam();

    Image image = reader.read(0, param);
//got an image file

    BufferedImage bufferedImage = new
BufferedImage(image.getWidth(null)
image.getHeight(null)
BufferedImage.TYPE_INT_RGB);
//bufferedImage is the RenderedImage to be written

    Graphics2D g2 = bufferedImage.createGraphics();
```

```
g2.drawImage(image, null, null);

    StringBuilder str=new
StringBuilder("client"+clientNo);
String fileName=str.toString();

    File imageFile = new
File("C:
Users
Ajmal
Downloads
opencv
FakeCurrencyDetection
"+str+".jpg");
ImageIO.write(bufferedImage, "jpg", imageFile);

    FakeCurrencyDetection fcd=new
FakeCurrencyDetection();
servermessage=fcd.takinginput(str);

    outStream.writeUTF(servermessage);

    // inStream.close();
outStream.close();

    serverClient.close();
catch(Exception ex)
System.out.println(ex);

finally
System.out.println("Client -" + clientNo + " exit!! ");
```

A.1.3 Main class

```
import org.opencv.core.Core;
import org.opencv.core.CvType;
```

```
import org.opencv.core.Mat;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;

    public class FakeCurrencyDetection

    public static void main(String[] args)
// TODO Auto-generated method stub
StringBuilder str=new StringBuilder("1111");
/*StringBuilder fl=new StringBuilder("100");
for(int i=0;i<7;i++)
int len=fl.length();

    System.out.println("10"+i);
fl.deleteCharAt(len-1);
fl.append(i);

    takinginput(fl);

    takinginput(str);

    public static String takinginput(StringBuilder str)

    //Object Define
Adjustment adj=new Adjustment();
GrayScale gray=new GrayScale();
Sharp sharp = new Sharp();
AdaptiveThresholding athresh= new
AdaptiveThresholding();
Segmentation seg= new Segmentation();
RandomForest rf=new RandomForest();

    String output="";
System.loadLibrary( Core.NATIVE_LIBRARY_NAME );

    //Read the source image
Mat source =
Imgcodecs.imread(str.toString()+".jpg");
```

```
    //Initialize matrix to store output
Mat adjustout=new Mat();

    //Processing

    //Image adjustment
adjustout=adj.adjustImage(source);
Imgcodecs.imwrite("adjustment.jpg", adjustout);

    //Image GrayScale
gray.grayImage();

    //Sharpening the image in-order to remove noise
sharp.sharpImage();

    //Adaptive thresholding
athresh.adaptiveThresh();

    //Segmentation
seg.segmentation();

    //Random Forest
output=rf.Rforest();

    return output;
```

A.1.4 Image Adjustment

```
import org.opencv.core.Size;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;

    public class Adjustment

    //function to adjust image dimensions
```

```
public Mat adjustImage(Mat source)
//mat to store the result
Mat resizeImage = new Mat();

    try
System.loadLibrary( Core.NATIVE_LIBRARY_NAME );

    //Mat croppedimage = cropImage(image,rect);

    //create filter for resizing
Size sz = new Size(3576,1412);

    //resize the image
Imgproc.resize( source, resizeImage, sz );
    catch (Exception e)
System.out.println("Error: " + e.getMessage());


    //return resized image
return resizeImage;
```

A.1.5 GrayScale

```
import java.awt.image.BufferedImage;
import java.awt.image.DataBufferByte;

    import java.io.File;
import javax.imageio.ImageIO;

    import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;

    public class GrayScale
//function for converting colored image into gray scale
```

```
public void grayImage()
try
System.loadLibrary(
Core.NATIVE_LIBRARY_NAME );

    //load the input file
File input = new File("adjustment.jpg");

    //convert the image file into buffered stream
BufferedImage image = ImageIO.read(input);

    //get data about RGB (three channel data)
byte[] data = ((DataBufferByte)
image.getRaster().getDataBuffer()).getData();

    //convert byte data to mat
Mat mat = new Mat(image.getHeight(),
image.getWidth()  CvType.CV_8UC3);
mat.put(0, 0, data);
mat1  Imgproc.COLOR_RGB2GRAY);

    //converting Mat data to byte stream
byte[] data1 = new byte[mat1.rows() * mat1.cols() * (int)(mat1.elemSize())]; mat1.get(0  0
data1);
BufferedImage image1 = new BufferedImage(mat1.cols(),mat1.rows(), BufferedImage.TYPE_BYTE_GRAY
image1.getRaster().setDataElements(0, 0, mat1.cols()  mat1.rows()  data1);

    //create new empty file
File ouptut = new File("grayscale.jpg");

    //write image mat in file
ImageIO.write(image1, "jpg", ouptut);

    catch (Exception e)
System.out.println("Error: " + e.getMessage());
```

A.1.6 Sharp Gray

```
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.Size;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;

    public class Sharp

    //sharpen the Gray image
public void sharpImage()
try
System.loadLibrary( Core.NATIVE_LIBRARY_NAME );
    //load the input image
Mat source = Imgcodecs.imread("grayscale.jpg", Imgcodecs.CV_LOAD_IMAGE_COLOR);

    //create empty mat to store image
Mat destination = new Mat(source.rows(),source.cols(),source.type());

    //apply filter for sharpening
Imgproc.GaussianBlur(source, destination, new Size(0,0), 60);

    //add weights
Core.addWeighted(source, 1.5, destination, -1, 0, destination);

    //write output
Imgcodecs.imwrite("sharpgray.jpg", destination);
catch (Exception e)
```

A.1.7 Adaptive Thresholding

```
import java.util.ArrayList;
import java.util.List;
import java.util.Vector;

    import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
```

```
import org.opencv.core.MatOfPoint;
import org.opencv.core.MatOfPoint2f;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;

    public class AdaptiveThresholding

    //function for Adaptive Thresholding
public void adaptiveThresh()
try

    System.loadLibrary( Core.NATIVE_LIBRARY_NAME );

    // Reading the Image from the file and storing it in to a Matrix object
String file ="sharpgray.jpg";

    // Reading the image
Mat src = Imgcodecs.imread(file,0);

    // Creating an empty matrix to store the result
Mat dst = new Mat(src.height(), src.width(), CvType.CV_8UC1);
Imgproc.adaptiveThreshold(src, dst, 250, Imgproc.ADAPTIVE_THRESH_MEAN_C, Img-
proc.THRESH_BINARY, 51, 7);

    // Writing the image
Imgcodecs.imwrite("threshbinaryreview.jpg", dst);

    // System.out.println("Image Processed");

    catch (Exception e)
System.out.println("error: " + e.getMessage());
```

A.1.8 Segmentation

```
import java.awt.Rectangle;
import java.util.ArrayList;
import java.util.List;
import java.util.Vector;

    import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.MatOfPoint;
import org.opencv.core.MatOfPoint2f;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;

    public class Segmentation

    //function for segmentation return type is list of matrix
public void segmentation()

    try
// Loading the OpenCV core library
System.loadLibrary( Core.NATIVE_LIBRARY_NAME );

    // Reading the Image from the file and storing it in to a Matrix object
String file ="C:
Users
Ajmal
Downloads
opencv
FakeCurrencyDetection
threshbinaryreview.jpg";
Mat src = Imgcodecs.imread(file,CvType.CV_8UC1);

    /* //drawing rectangle on ROI
```

```
Imgproc.rectangle(src, new Point(573,571), new Point(771,989),new Scalar(0,255,0),8);
Imgproc.rectangle(src, new Point(266,1120), new Point(788,1268),new Scalar(0,255,0),8);
Imgproc.rectangle(src, new Point(145,300), new Point(1085,489),new Scalar(0,255,0),8);
Imgproc.rectangle(src, new Point(2200,1149), new Point(3189,1313),new Scalar(0,255,0),8);
Imgproc.rectangle(src, new Point(3311,695), new Point(3419,787),new Scalar(0,255,0),8);
Imgproc.rectangle(src, new Point(3001,621), new Point(3149,869),new Scalar(0,255,0),8);
Imgproc.rectangle(src, new Point(2025,93), new Point(3201,197),new Scalar(0,255,0),8);
Imgproc.rectangle(src, new Point(1939,117), new Point(2031,1261),new Scalar(0,255,0),8);
Imgproc.rectangle(src, new Point(2521,269), new Point(2997,813),new Scalar(0,255,0),8);
//watermark(change required)
Rect roi1 = new Rect(65,625,(265-65),(761-625));
//100 in floral
Rect roi2 = new Rect(225,801,(377-225),(997-801)); //hidden 100
Rect roi3 = new Rect(65,765,(233-53),(917-765)); //id mark
Rect roi4 = new Rect(305,1161,(1313-305),(1349-1161)); //serial no left
Rect roi5 = new Rect(2553,41,(3517-2553),(245-41)); //serial no right
Rect roi6 = new Rect(1061,157,(2285-1061),(301-157)); //RBI mark
Rect roi7 = new Rect(245,341,(941-245),(1069-341)); //watermark
Rect roi8 = new Rect(2025,1,(2227-2025),(1410-1)); //security thread
*/
    //extracting ROI
Rect roi1 = new Rect(65,625,(353-65),(780-625)); //100 in floral
Rect roi2 = new Rect(260,760,(360-260),(947-760)); //hidden 100
Rect roi3 = new Rect(128,800,(255-128),(905-800)); //id mark
Rect roi4 = new Rect(305,1161,(1313-305),(1349-1161)); //serial no left
Rect roi5 = new Rect(2553,41,(3517-2553),(245-41)); //serial no right
Rect roi6 = new Rect(1061,157,(2285-1061),(301-157)); //RBI mark
Rect roi7 = new Rect(349,407,(1015-349),(1027-407)); //watermark
Rect roi8 = new Rect(2085,1,(2285-2085),(1410-1)); //security thread
// Rect roi9 = new Rect(2605,301,(3101-2605),(873-301)); //watermark(change required)

    List<Mat> feature = new ArrayList();

    feature.add(0, src.submat(roi1));
feature.add(1, src.submat(roi2));
feature.add(2, src.submat(roi3));
feature.add(3, src.submat(roi4));
feature.add(4, src.submat(roi5));
feature.add(5, src.submat(roi6));
```

```
feature.add(6, src.submat(roi7));
feature.add(7, src.submat(roi8));
// feature.add(8, src.submat(roi9));

    StringBuilder str=new StringBuilder("try1");

    // Writing the image
for(int i=0;i<feature.size();i++)
//str.append(i);
str.deleteCharAt(3);
str.append(i);
// System.out.println(str);
Imgcodecs.imwrite("C:
Users
Ajmal
Downloads
opencv
FakeCurrencyDetection
"+str+".jpg",feature.get(i));

    Imgcodecs.imwrite("C:
Users
Ajmal
Downloads
opencv
FakeCurrencyDetection
allcontours.jpg",src);

    // System.out.println("Image Processed ");
catch (Exception e)
System.out.println("Error: " + e.getMessage());
```

A.1.9 Random Forest

```
import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.MatOfDouble;
```

```
import org.opencv.core.Size;
import org.opencv.core.TermCriteria;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.ml.Ml;
import org.opencv.ml.NormalBayesClassifier;
import org.opencv.ml.RTrees;
import java.io.FileReader;
import java.io.BufferedReader;

    public class RandomForest
public String Rforest()
String output = "";
try
System.loadLibrary( Core.NATIVE_LIBRARY_NAME ); Mat trainingTraits = new Mat(100,25,CvType.C
Mat trainingclass = new Mat(100,1,CvType.CV_32S);

    //input dataset
FileReader trainfile = new FileReader("kanchan.csv");
BufferedReader br = new BufferedReader(trainfile);
int counter=0;
String line;
double [] a;
double [] b;

    if(trainfile!= null)
// System.out.println("File opened");

    while ((line = br.readLine()) != null)

    // use comma as separator
String [] feature = line.split(",");

    for(int col=0;col<25;col++)
if(col<24)
trainingTraits.put(counter,col,Double.valueOf(feature[col]));
else
trainingclass.put(counter,0,Integer.valueOf(feature[col]));

    /* for(int col=0;col<22;col++)
```

```
if(col<21)
a=trainingTraits.get(counter, col);
// System.out.println(a[0]);  else
b=trainingclass.get(counter,0);
// System.out.println(b[0]);
*/ counter++;

    //Training Machine
boolean flag;
// int layout = 5;
// RTrees rt = RTrees.create();
// TermCriteria criteria = new TermCriteria(TermCriteria.EPS + TermCriteria.MAX_ITER,200,0.1);
// rt.setTermCriteria(criteria);
// rt.setActiveVarCount(12);
// rt.setCalculateVarImportance(true);
// rt.setMaxCategories(2);
//flag=rt.train(trainingTraits,Ml.ROW_SAMPLE, trainingclass);

    NormalBayesClassifier nb =NormalBayesClassifier.create();

    flag=nb.train(trainingTraits,Ml.ROW_SAMPLE, trainingclass);
System.out.println(flag);

    MatOfDouble stddev =new MatOfDouble();
MatOfDouble mean = new MatOfDouble();

    StringBuilder str=new StringBuilder("try0");
int count=0; double[] values=new double[24];
double meanvalue,stddevvalue,varvalue;
// Writing the image
for(int i=0;i<8;i++)

    str.deleteCharAt(3);
str.append(i);
//System.out.println(str);
Mat src = Imgcodecs.imread(str+".jpg");
Core.meanStdDev(src, mean, stddev);

    meanvalue = mean.get(0,0)[0];
```

```
stddevvalue= stddev.get(0,0)[0];
varvalue=stddevvalue*stddevvalue;

    // System.out.println(meanvalue);
    values[count++]=meanvalue;
values[count++]=stddevvalue;
values[count++]=varvalue;

    Mat testSamples = new Mat(new
Size(24,1),CvType.CV_32FC1);
System.out.println();
for(int i=0;i<24;i++)

    System.out.println(values[i]);

    testSamples.put(0,i,values[i]);

    // System.out.println("height: " + testSamples.height() + " width: " + testSamples.width());

    //Testing

    double pp=nb.predict(testSamples);
System.out.println("printed:"+ pp);
    if(pp==1.0)
output="real";
System.out.println("Its real");
else
output="fake";
System.out.println("its fake");

    System.out.println("Value of counter is :" + counter);
else
System.out.println("File can not be opened");
catch(Exception e)
System.out.println("Error: " + e.getMessage());
return output;
```

A.2 Screen-shots of Application

Figure A.1: Image Capturing Interface

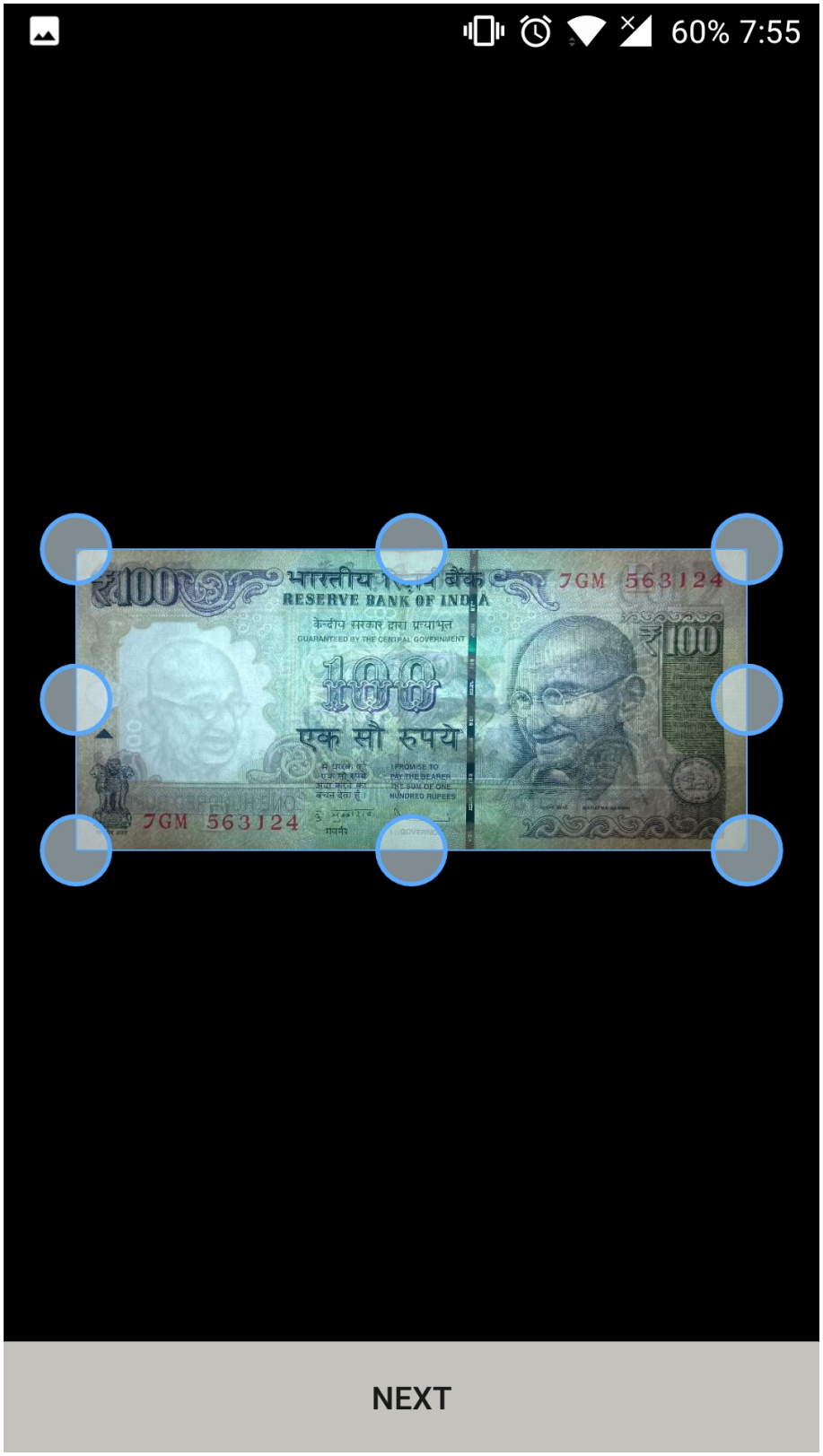

Figure A.2: Image cropping Interface

Figure A.3: Application main Interface

Figure A.4: Application processing

Figure A.5: Application Result

Bibliography

[1] IEEE Std 830-1998 IEEE "*Recommended Practice for Software Requirements Specifications*"; IEEE Computer Society,.

How to extract features like watermark, security thread, identification mark and micro-lettering from the currency note.

[2] Prasanthi, B. S., and Setty, D. R. (2015). " *Indian paper currency authentication system using image processing*"; Int. J. Sci. Res. Eng. Technol, 4, 973-981.

How to extract features like watermark, security thread, identification mark and micro-lettering from the currency note.

[3] Ajinkya babar, Swapnil Jawalekar, Kiran Yadav, Dr. D. B. Salunke, Vol.(2015). "*Counterfeit Currency Detector, International Journal for Technological Research and Application*"; e-ISSN: 2320-8163.

Various features and procedure to detect the authenticity of the currency note.

[4] Alekhya, D., Surya Prabha, G., and Durga Rao, G. (2014)." *Fake Currency Detection Using Image Processing and Other Standard Methods*"; IJRCCT, 3(1), 128-131

Image Processing approach for detecting fake currency. Steps to find out fake currency 1) image acquisition 2) pre-processing 3) Gray scale conversion 4) edge detection 5) segmentation 6) comparing 7) decision making 8) generate output

[5] Yadav, B. P., Patil, C. S., Karhe, R. R., and Patil, P. H. (2014)." *Indian Currency Recognition and Verification System Using Image Processing. International Journal of Engineering Sci-ence and Innovative Technology*" (IJESIT)(ISSN: 2319-5967 ISO 9001: 2008 Certified Vol-ume 3, Issue 4.

Flowchart for Currency Recognition and Verification System

[6] Thakur, M., and Kaur, A. (2014). "*Various fake currency detection techniques. International Journal for Technological Research in Engineering*", 1(11), 1309-1313

Image Processing approach for detecting fake currency. Steps to find out fake currency 1) image acquisition 2) pre-processing 3) Gray scale conversion 4) edge detection 5) segmentation 6) comparing 7) decision making 8) generate output.

[7] Mirza, R., and Nanda, V. (2012). "*Design and implementation of indian paper currency authen-tication system based on feature extraction by edge based segmentation using Sobel opera-tor. International Journal of Engineering Research and Development*", 3(2), 41-46

Various feature extraction approaches of Indian currency like gradient edge detection. Edge detection can be used to find sharp changes in brightness of image and can be used to find boundaries of objects.Serial number feature detection technique and approach

[8] Gopal Krishnan, "*Image Processing based feature extraction for Indian Currency Notes. The-sis report submitted towards the partial fulfillment of requirements for the award of the de-gree of Master of Technology In VLSI Design and CAD*".

Image enhancement, including filtering, filters design, deblurring, and contrast enhancement. Image analysis, including feature detection, morphology, segmentation, and measurement. Spatial transformations and image registration. Image transforms, including FFT, DCT, Radon, and fan-beam projection. Support for multidimensional image processing. Support for ICC version 4 color management system.