TEXT RECOGNITION USING ANDROID BASED MOBILES

By

NC Nitasha Safeer

NC Sehrish Naeem

PC Syed Ahsen Raza

Submitted to the Faculty of Computer Software Engineering

National University of Sciences and Technology, Islamabad in partial fulfillment

for the requirements of a B.E. Degree in Computer Software Engineering

JUNE 2012

ABSTRACT

In optical character recognition (OCR), visible characters appearing as images (i.e. on paper) are recognized as symbolic characters (in ASCII) and stored in a computer's memory or similar device like mobile phone. Modern high-end mobile camera phones are capable of capturing color images whose sizes are at least 1280 × 960 pixels, which is usually quite adequate for performing Optical Character Recognition. Having OCR software available for a camera phone has obvious benefits like it can be used to make quick translations of foreign words, to store notes space-efficiently from lecture slides or notice boards. As technology improves, it can become an invaluable tool for visually challenged people. As demand grows for mobile phone applications, research in optical character recognition, a technology well developed for scanned documents, is shifting focus to the recognition of text embedded in digital photographs.

Through this project titled as "Text recognition on Android based mobile devices" we find the solution to printed text recognition problem accompanied by some feature rich applications which purposefully use the recognized text. Text recognition phase of the project involves a combination of image processing, feature extraction and machine learning (in our case it is MLP (Multi Layer Perceptron) based approaches.

Initially some preprocessing operations are performed on the input image, followed by text segmentation into entities like textual lines, words and characters. Broken characters after this phase are reconstructed using static and dynamic classifiers. Well defined individual characters are then sent for classification to our classification module, which classify and output the input character in ASCII based on its matching to training set data.

All characters converted to ASCII are then grouped together to form complete words and textual lines in post processing phase.

The system is trained with 20 training samples for each character at minimum. Multiple training sessions after iterative testing and evaluation process generated the final training data file. This training file was then used to classify the input character in our project.

Our project purposefully uses the recognized text into six applications named as Text to speech, Translation of text, Transliteration of text, SMS, Email and automatically adding new contact from business cards. Various Testing and evaluation results conducted on the product are extremely promising.

DECLARATION

No portion of the work presented in this dissertation has been submitted in

support of another award or qualification either at this institution or elsewhere.

DEDICATION

In the name of Allah, the Most Merciful, the Most Beneficent

To our parents, without whose unflinching support and unstinting cooperation,

a work of this magnitude would not have been possible

ACKNOWLEDGEMENTS

There is no success without the will of ALLAH. We are grateful to ALLAH, who has given us guidance, strength and enabled us to accomplish this task. Whatever we have achieved, we owe it to Him, in totality. We are also grateful to our parents and family and well-wishers for their admirable support. We would like to thank our supervisors Col. Dr. Fahim Arif and Dr. Imran Siddiqi, for their help and motivation throughout the course of our project. Without their help we would have not been able to accomplish anything. They provided us with the opportunity to polish our technical skills and guided us into this area of learning.

TABLE OF CONTENTS

List of Figures

Figures **Page Number**

List of Tables

Table Number **Page Number**

CHAPTER 1

INTRODUCTION

1. Introduction

1.1 Purpose

This project is aimed at developing a Text Recognition System on Android based mobile devices. The system mainly targets printed text but limited support for non-cursive handwritten text is also provided. Textual content in the image is localized and converted to text using an Android based OCR. The text can then be translated into a large number of target languages. The system also provides text to speech facility to assist the visually impaired.

1.2 Project Background

An OCR (Optical Character Recognition) is a complex algorithm that recognizes characters from a document image captured by a scanner, hand held or cell phone cameras. The output of the OCR is generally in a machine-readable format .The goal of Optical Character Recognition (OCR) is to classify optical patterns (often contained in a digital image) corresponding to alphanumeric or other characters.

The convergence of powerful processors and high resolution cameras on mobile devices has made them an attractive platform for optical character recognition (OCR) software .Most traditional OCR applications have been designed to be highly automated and used on desktop machines .These recognition engines perform well but usually require high quality input images that are reliably obtained. In adapting recognition systems for use on mobile devices, conventional assumptions about the need for automation, available processing power, and input image quality were re-evaluated by the developers in the domain.

The project tilted as “Text Recognition on Android based Mobile Devices” will find the solution to text recognition problem on android based mobile devices. The core tasks involved in the project are image capturing, text localization into textual lines and text recognition accompanied with a useful application of recognized ASCII text translation to another language.

Figure 1 : Text Recognition using Android based mobile phone. (a): Capture image using mobile’s camera. (b): Textual image captured. (c): Recognized text from the captured image.

1.3 Project Scope

Purpose of the system is to develop an Android based application capable of localizing the textual regions in the input image and convert it into text .The work will mainly focus on printed text without skew. Optionally, skew correction and limited recognition of isolated handwritten characters may also be added. Only a limited range of fonts will be handled by the system. Subject to availability of time, a number of applications based on the recognized text can also be developed.

1.4 Objectives

The objectives of our project include:

- To learn mobile application development/interfacing on Android OS.
- To Learn and apply text recognition algorithm on Android OS.
- To develop a useful application on recognized text.
- To learn and understand a basic software (programming environment i.e. java code) and hardware (android based mobile device) interfacing through data cable.

1.5 Work Breakdown Structure

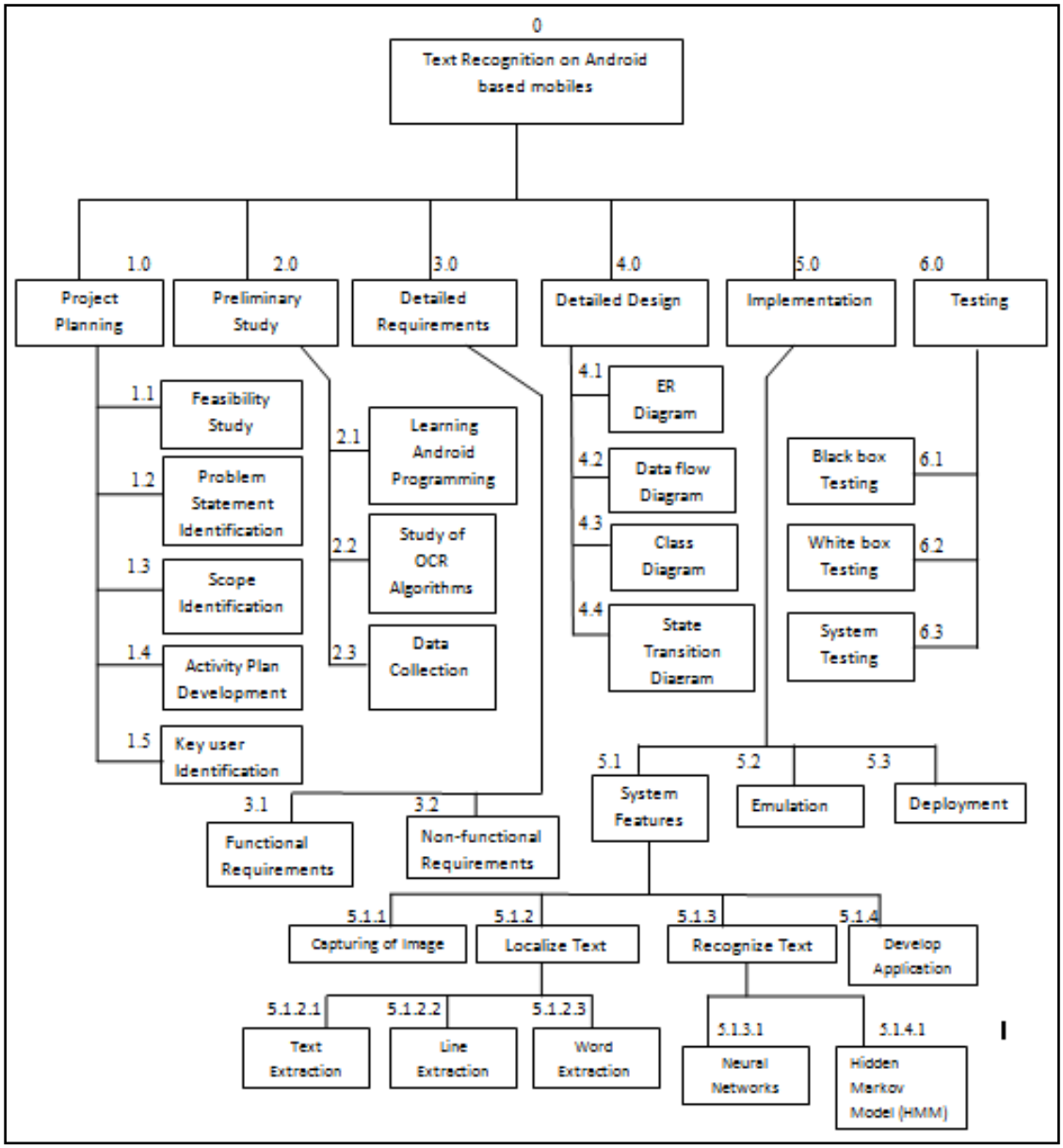


Figure 2: Work Breakdown Structure

1.6 Deliverables

Table 1: Deliverables

Deliverable Name	Deliverable Summary Description
Software Requirements Specification(SRS) Document	Complete Description of WHAT system will do, who will use it. Detailed description of functional and non-functional requirements and system features.
Analysis Document	Detailed requirement analysis and analysis models are included.
Design Document	Complete description of How the system will do. Design models are included.
Code	Complete code with the API.
Testing Document	Whole system is tested corresponding to the specifications. System is tested at all levels of Software Development Life Cycle (SDLC).
Complete System	Complete working system.

CHAPTER 2

LITERATURE REVIEW

2. Literature Review

2.1 Introduction

An OCR (Optical Character Recognition) is an algorithm that recognizes characters from a document image captured by a scanner, hand held or cell phone cameras and converts them into text. As indicated in Figure 3, the goal of an Optical Character Recognition (OCR) system is to classify optical patterns (often contained in a digital image) corresponding to alphanumeric or other characters.

Over the recent years, the Document Analysis and Recognition (DAR) have been attracted by new problem areas within the context of Camera Based Document Analysis and Recognition (CBDAR). CBDAR deals with the textual information in scene and document images taken by low cost hand held devices like digital cameras, cell phone cameras, etc. Numerous applications like text translation for foreigners, reading text for visually impaired, information retrieval from media documents etc., can be developed using the techniques developed in the CBDAR domain.

The evolution of powerful processors and high resolution cameras on mobile devices has made them an attractive and a suitable platform for optical character recognition (OCR) software. Most traditional OCR applications have been designed to be highly automated and used on desktop machines. These recognition engines perform well but usually require high quality input images that are reliably obtained. In adapting recognition systems for use on mobile devices, the development is constrained by parameters like available processing power and input image quality etc.

This domain has emerged in the last ten years and has attracted the attention of document analysis and recognition, computer vision and pattern recognition communities. The

document can be of any type ranging from office documents (constrained environment and an exact format) to natural scene images (where a variety of parameters can be changed, these includes font size of text, style of text, background variations). This particular domain deals with document images captured using emerging devices like digital cameras, PDAs, cell phone cameras. These devices are becoming more and more popular these days due to their low price, weight, portability, flexibility and integration with other networks [1]. The cheaper devices make it possible for everyone to capture images and store them in different formats. In [1], the authors have compared the advantages and disadvantages of traditional scanner technology with the latest digital/hand held and mobile phone cameras.

The document images are generally captured by a scanner or taken with different types of cameras either hand held or cell phone cameras. As a first step the resulting image is converted to a binary image where text pixels are black with white background. Generally the text is printed .The main objective is to localize the textual content in the image and recognize text along with the structure of the document as shown in Figure 4.

The core tasks involved in the project involve identification of text lines and words and recognition of individual characters, which can then be joined into words and lines. Optionally we can include the document images having skew, light background variations. The recognized text can ultimately be used to develop any of the above mentioned applications. The undertaken project's scope is **"To develop an Android based application capable of localizing the textual regions in the input image and convert it into text".**

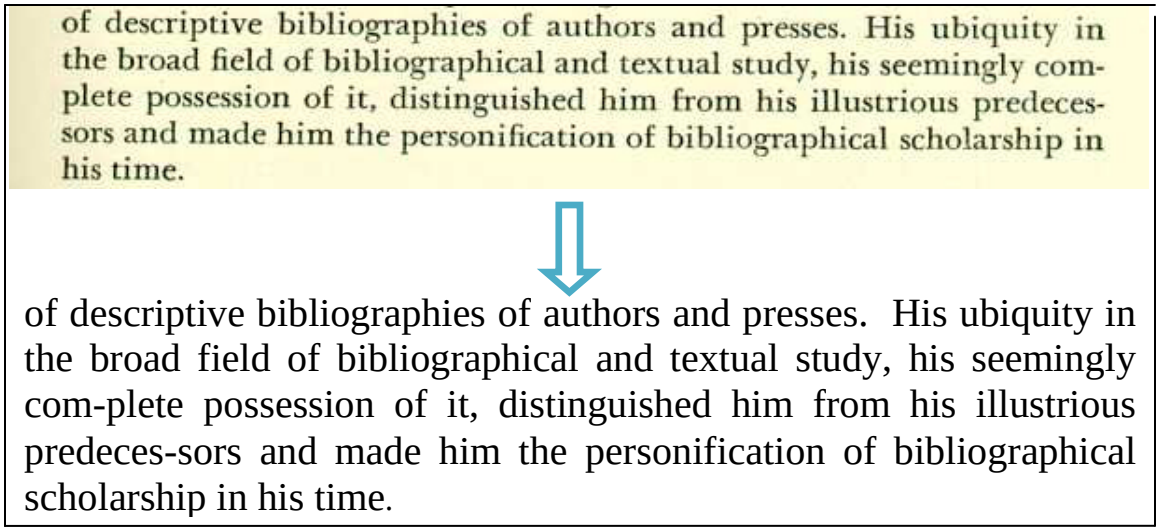


Figure 3: Scanned image of text and its corresponding recognized representation.

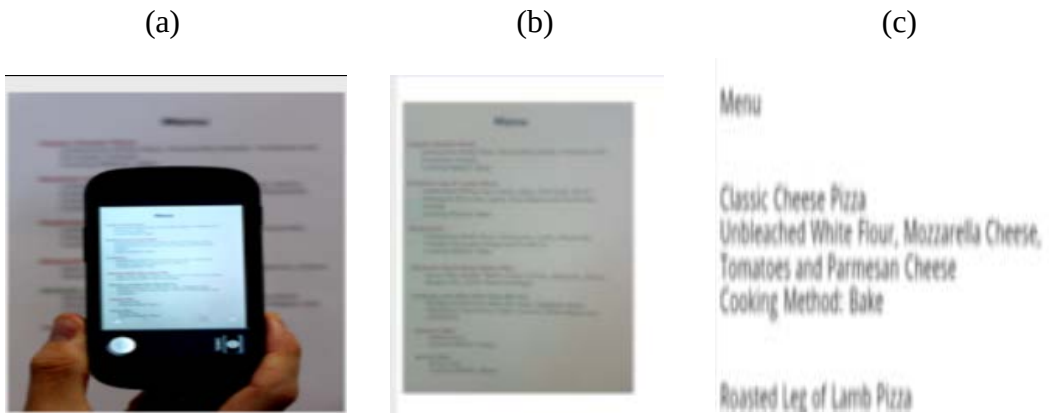


Figure 4 : Text Recognition using Android based mobile phone. (a): Capture image using mobile's camera. (b): Textual image captured. (c): Recognized text from the captured image.

2.2 Problem Domain

The project falls under the broader umbrella of Camera based Document Analysis and Recognition (CBDAR) with focus on mobile devices having "Android" as its operating system. The domain is itself newly emerging in mobile devices and promises a lot.

The overall capabilities of mobile devices have rapidly increased in recent years in terms of processing power, connectivity, and available sensors. These advancements, together

with the growing prevalence of smart phones, have made it feasible and in some cases preferable to run OCR software on mobile platforms. This led the researchers with the co-operation of the developers to take initiative. Naturally, image processing applications that are ported to mobile devices frequently must adapt to very different technical and usage assumptions than those under which they were originally developed. For example, traditional OCR users often use workstations to set up long running, automated jobs to process large amounts of scanned documents. In contrast, mobile users might only need to process short segments of text on demand and with the expectation of low response times.

The text recognition itself is a mature research area and a number of applications have been developed on a variety of platforms. For Google phones the text recognition application was initiated by Google itself in the name of Google Goggles.

2.3 Related Work

Since the popularization of digital cameras, efforts have been made to adapt text recognition algorithms to work on digital/hand held, mobile cameras. In shifting from the pseudo-binary nature of scanned text images to real world pictures, much focus has been on the problem of text localization and extraction. Work in areas such as text line detection, perspective removal, and environmental background removal from natural pictures have yielded accuracy rates ranging from 50 to 70 percent [2-4], depending on the images tested. Some useful surveys on general issues surrounding camera-based text digitization [5, 6] reveal the efforts of mobile application development community.

Researchers have built early examples of standalone OCR systems that run on cell phone hardware [7–9]. Senda et al. [10] demonstrate a “camera-typing” interface that relies on the concatenation of multiple photos taken by a Smartphone camera for text input. The system built by Laine and Nevalainen [9] was found to consume most of its computational time performing tasks that are specific to adapting text recognition for mobile environments (e.g., skew correction, perspective transformations). Experiments from the mentioned systems suggested that performing OCR on commodity mobile hardware was of borderline practicality due to long processing times. This problem is likely to have been lessened to an extent by recent advances in mobile technology. Work in mobile OCR by Joshi et al. [11] takes advantage of new hardware features such as the tilt sensor to aid proper user input.

Prototypes of mobile-phone OCRs for digits and the Roman alphabet appeared in 2001, and commercial products came to market in 2002 [37],[10]. These recognize English words for looking-up in a dictionary, e-mail addresses, URLs, phone numbers, etc. The ability to detect and recognize text using PDAs or cellular phones has promise for both commercial and military applications. Watanabe [39] was the first to describe a sign translation system (Japanese to English). Yang et al. [40], [41] implement an experimental Chinese-to-English sign translation system and give a good analysis on the three components of a general sign translation system

Sign detection, character recognition, and sign translation. Zandifar et al. [42] propose to use camera-based OCR techniques in a head-mounted smart video camera system to help the visually impaired. Their goal is to detect and recognize text in the environment and then convert text to speech.

A number of OCR engines are widely available and in current use. Open-source projects include GOCR [27], OCRAD [28], OCROpus [12], OpenOCR [29], and Tesseract [13,14]. Commercial products supporting mobile platforms include ABBYY Fine Reader [30], Nuance OmniPage [31], and ExperVision Open- RTK [32], among others.

In [34], the authors designed an image preprocessing suite on top of OCR engine to improve the accuracy of text recognition in natural scene images. Hitachi [38] Again, all the processing is implemented at the back end server with no implementation on the mobile device. Nowadays some mobile phones are equipped with business card reader application which facilitates users to store contact information from business cards directly on their mobile phones [35]. Also in [36], authors have discussed about a mobile application to capture barcodes of any item and get detailed information about its ratings, price, and reviews.

Image recognition systems that incorporate human- assistance have been developed. Wilkinsonet al. [15] used an OCR engine augmented with human verification to generate large test sets for other OCR systems. Blostein [16] approached assistance from the user interface perspective, employing specialized input devices and N-best lists to correct errors in diagram recognition. User-driven segmentation and classification were shown to improve recognition of UML diagrams in a system built by Lank et al. [17].

Researchers in the area of mathematics recognition have demonstrated style-preserving morphs for inputting and manipulating hand-written formulas [18, 19]. Mankoff et al. [20] created a user interface toolkit consisting of reusable error correction elements and architectural extensions to provide structural support for handling ambiguity in recognizers.

Augmented reality is a relatively new area where several text recognition and translation applications have emerged. Nokia Point & Find [33] was an experimental application that recognized movie posters in real time and displayed relevant information to the user. World Lens is an application for iPhone that translates text visible in the viewfinder and overlays the results on the screen in real time. The system works without Internet connectivity and currently supports translation from English to Spanish and vice versa. In the academics Fragoso et al. [21] built a similar application that relies on a web-based service for translation.

In the next section, we discuss the contributions to Android based text recognition applications.

2.3.1 Android based Text Recognition Applications

A close look at the literature revealed categories of text recognition application already developed on Android's platform. They are broadly categorized into two categories, which includes Small-scale text recognition application on Android and Feature rich text recognition application on Android

2.3.1.1 Small-scale text recognition application

A variety of small scale applications with limited functionality have been developed over the years. These include recognizing feature specific text recognition. Examples in this category include: Digit recognition [22], Bar code recognition [23], Font (size, style) specific text recognition [22] and Number plate recognition [24].

(a) (b) (c) (d)

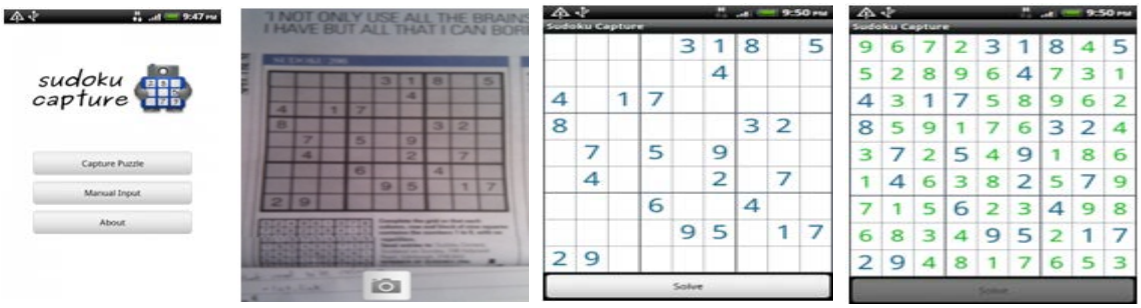
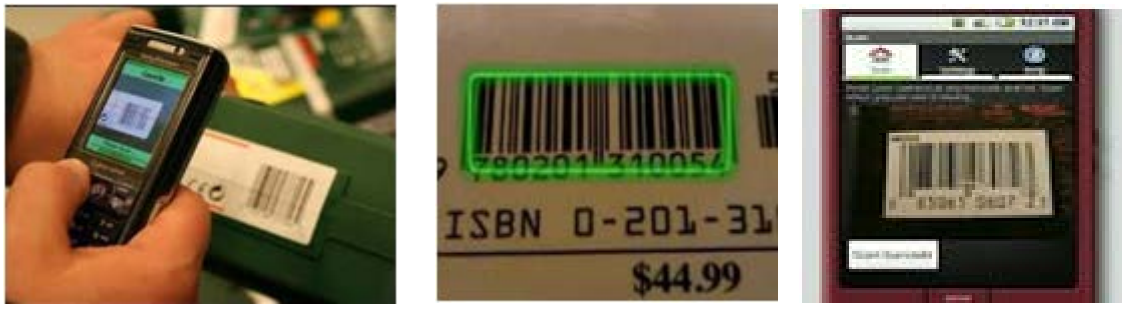


Figure 5 : Android based digit recognizer for Sudoku. (a):Application's main menu.(b):image captured using mobile's camera.(c):Digits recognized.(d):Sudoku game starts.

(a) (b) (c)



(d) (e)

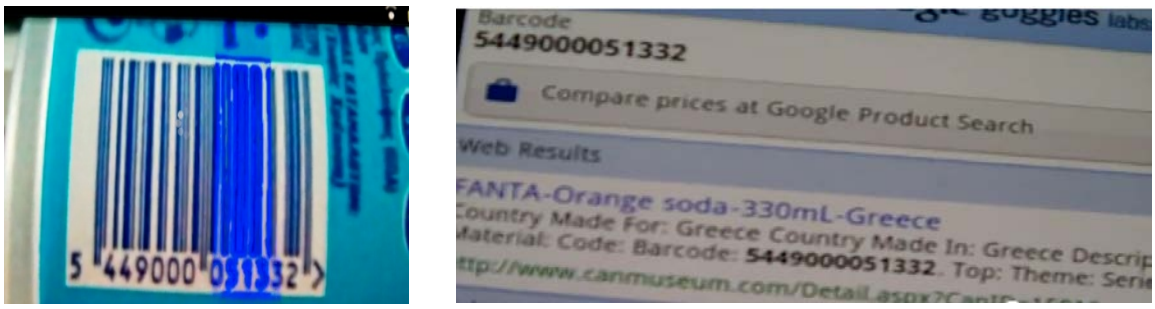


Figure 6: Android based bar code recognizer. (a):Barcode containing image capture.(b):Barcode localized in the image.(c):Barcode scanning start menu(d):Barcode scanner in action.(e):An application: bar code based price comparison

Figure 7: Android based Number Plate recognizer. (a): Number plate image captured (b): Number plate recognized and converted to numeric at the bottom in red color.

2.3.1.2 Feature rich text recognition applications

Significant efforts have been made to make the text recognition a feature rich application. The category of feature rich application includes text recognition by expanding the features of text recognition. This expanded scope includes:

- Recognizing text on non-homogeneous, cluttered, light affected, skew oriented text having background variations.
- Font (style, size) independent text recognition.
- Use the recognized text to make useful applications, like creating new contacts, query text, text translation, text transition (i.e. via SMS, Email) etc.

Few of the Android based text recognition applications have been able to address the mentioned features. Three of widely known applications include:

- Mezzofanti text recognition and application oriented package [25].Google Goggles text recognition application [26].

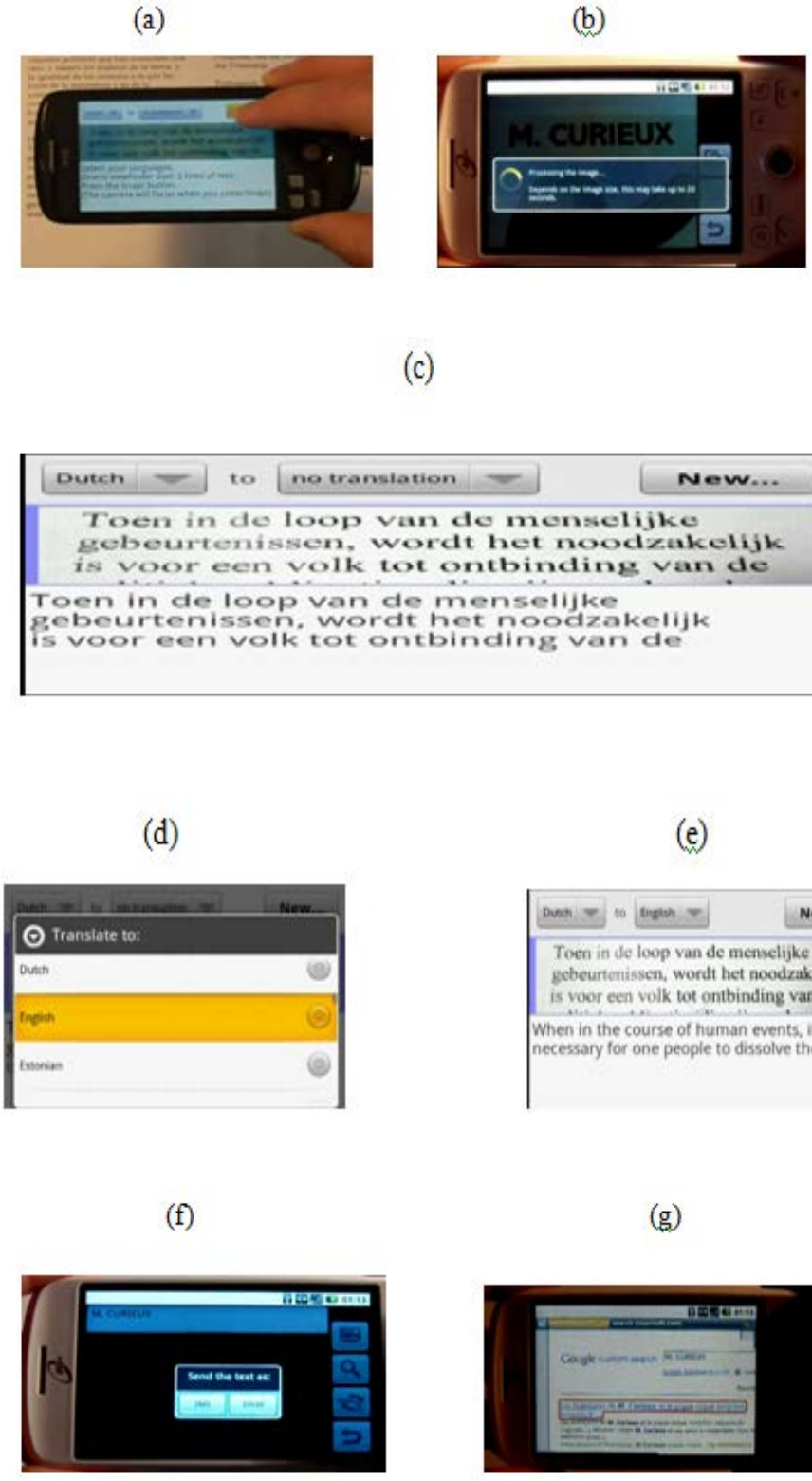


Figure 8: Mezzofanti text recognition application in action

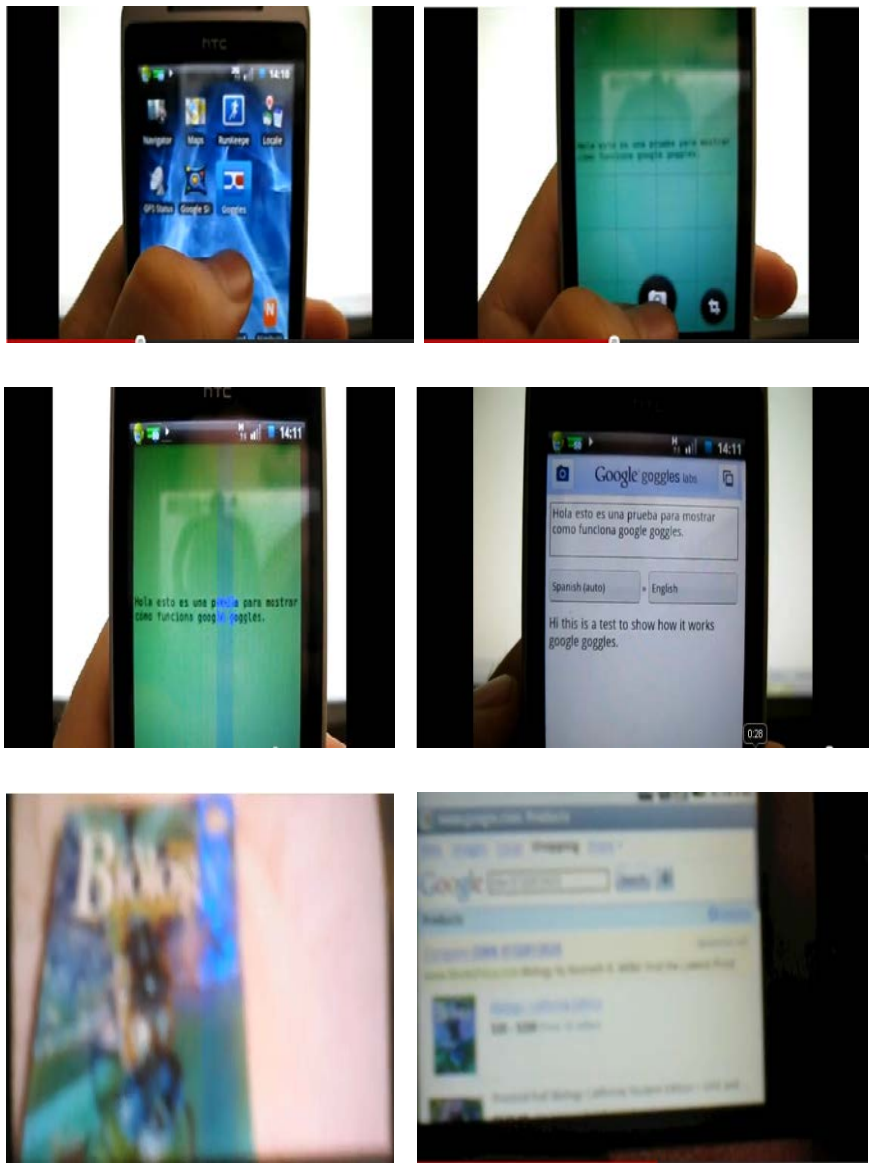


Figure 9: Google Goggles text recognition application in action. (a):Captured image using mobile's camera.(b):Captured text.(c): Text recognition module is processing to recognize the text.(d):An application: to translate the recognized text in the available set of languages.(e):An application: cover page of the book is scanned to search the book on the internet. (f): Book query results displayed

2.4 Shortcomings/issues

Mobile application development has always posed rigorous and difficult requirements to manage. Comparing with the desktop development where we have lesser bounds of memory, cost, size; mobile devices have a less share of these features. Naturally the resulting application could not be as much efficient as the desktop application may be.

Text recognition on Android platform can have following issues:

- One challenge in adapting OCR for mobile devices is in handling low quality input images. Conventional recognition systems assume input data to be high resolution images obtained from flatbed or sheet fed scanners. As a result, many existing engines are optimized for processing clean and sharp images of text. Unfortunately, hardware limitations and volatile operating environments usually cause cell phone cameras to produce lower fidelity images that suffer from defects such as noise or uneven lighting, perspective distortion, complex background etc.
- A second drawback in running OCR on mobile devices is that they have much lower processing capabilities than workstations. This presents an obstacle in particular for the recognition of complex scripts and characters such as those of certain East Asian languages.
- Memory bounds in mobile devices also make OCR a challenging task on these devices.

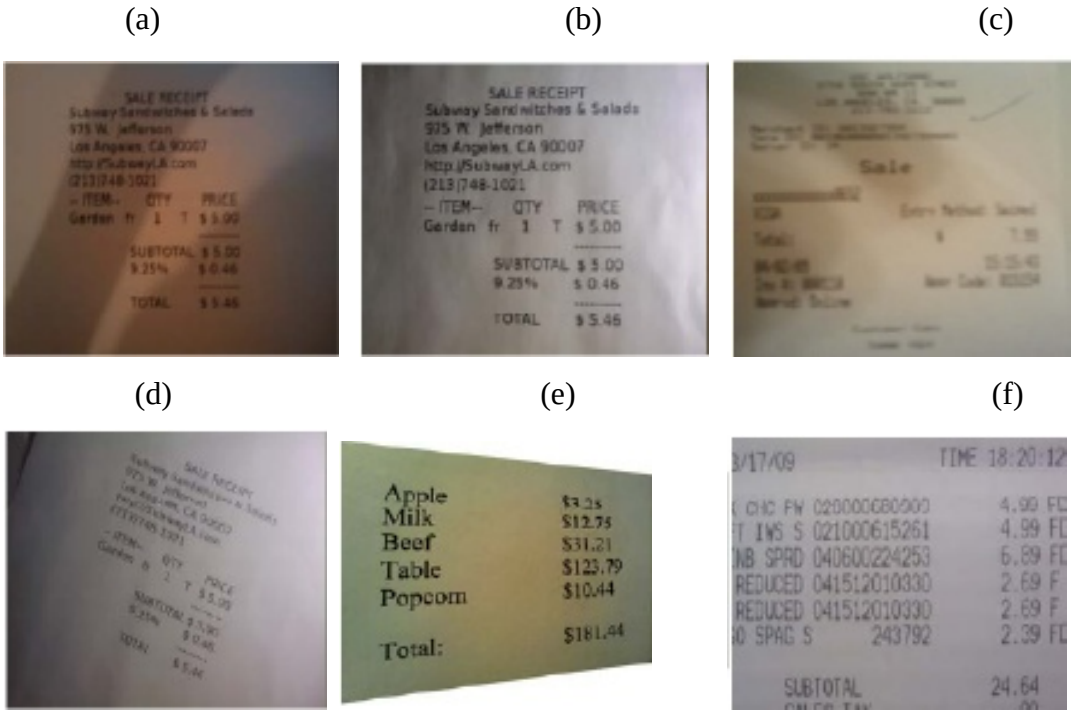


Figure 10: Different issues arising in camera-captured documents: (a) shading (b) flooding (c) blur (d) skew (e) tilted (f) misalignment.

2.5 Proposed Project

The proposed project finds the solution to text recognition problem on Google Phone having Android as OS. The core features of our project include:

- Capture picture/image through cell phones camera.
- Localizing the textual occurrences in the image.
- Text recognition of isolated printed text (i.e. text on business cards, word documents etc). Printed text is assumed to be
    - Non-skewed(tilted at some angle)
    - Homogenous text style
    - Non-cluttered (text characters are isolated from one another).

- Optional features include, using the recognized text to make some useful applications, like:
    - Query the recognized text on a search engine (i.e. Google).
    - Create a new contact list in the directory of mobile (in case of business card text recognition).
    - Translate the recognized text into another language.

2.6 Technological requirements

Our project requires following software and hardware requirements.

2.7 Software Requirements

The software(s) required for the implementation of our project includes:

- Java Development Kit 1.6 ,a pre-requisite for android platform
- Eclipse IDE for Java
- Android-SDK version 2.2 of Level 8 API

2.8 Hardware Requirements

The Hardware required for the implementation of our project includes:

- Mobile Phone having Android as its OS and camera of preferably 3.0 Megapixels.
- Data Cable for hardware and software interfacing.

CHAPTER 3

SYSTEM REQUIREMENT SPECIFICATION

3. Introduction

3.1 Purpose

The purpose of the project is to develop a text recognition system for Android based mobile devices. Printed text without skew is mainly the focus of the system but subject to time constraints; hand written text and skew can also be incorporated in the system. The recognized text may also be used for a wide variety of applications.

3.2 Document Conventions

Android based mobiles: Mobile Phones having Android as operating system

OCR: Optical Character Recognition

3.3 Intended Audience and Reading Suggestions

Intended Audience of SRS includes:

- Project Supervisor
- Project Coordinator
- Project Panel
- External Evaluators

This document completely describes the product scope, product perspective, product features, user classes and characteristics, operating environment, design and implementation constraints, user documentation, assumptions and dependencies, external interface requirements, system features, functional requirements, non functional requirements and other requirements. This document should be read in a sequence in which it is presented.

3.4 Product Scope

Purpose of the system is to develop an Android based application capable of localizing the textual regions in the input image and convert it into text .The work will mainly focus on printed text without skew. Optionally, skew correction and limited recognition of isolated handwritten characters may also be added**.** Only a limited range of fonts will be handled by the system. Subject to availability of time, a number of applications based on the recognized text can also be developed.

3.5 References

- Project Synopsis- An approved document with complete information containing extended title, brief description of the project, scope of work, academic objectives, applications, previous work, material resources required, no of students and special skill required has already been submitted to the department.

3.6 Overall Description

3.6.1 Product Perspective

The product will be an application of the developed text recognition system. The textual content in the image captured through an Android based mobile device will be converted to text and fed to an application like translation, card reading or text to speech conversion.

3.6.2 Product Functions

- Abstraction level of the system should be high.
- The product must be an application of text recognition.

3.6.3 User Classes and Characteristics

- **General Public** can use it for text/business card reading. Contact information on business cards can be directly saved in the address book of the mobile device.
- **Visually impaired individuals** can benefit from the developed system if the recognized text is passed to a speech converter.
- **Students** can use the system for automatic reading of books and articles. They can also edit the text, search for keywords on the internet and automatically convert the lecture slide images to notes. The system can also be extended to handwriting recognition allowing student to take images of board notes and convert them to text.
- **Non-native speakers** of languages based on the Latin alphabet (English, French and German etc.) can also benefit from the system by translating the text written on sign boards and other places into their desired language.

3.6.4 Operating Environment

Software system will operate on Android based mobile devices.

3.6.5 Design and Implementation Constraints

The application will be developed for mobile devices with limited memory and processing power. For an effective text recognition system, extensive training on large data sets is required naturally requiring large memory and high processing power. However, since the training phase is offline, it can be carried out on a desktop machine and the learned parameters can be used on the mobile device for recognition of the input textual content.

In addition, due to limited time of the project, some of the features like skew correction and recognition of isolated handwritten characters have been kept optional.

3.6.6 User Documentation

User manual and complete project report will be delivered along with the system. User will also be provided with a website containing the product and manuals.

3.6.7 Assumptions and Dependencies

- The project is based on the following assumptions.
- An Android based mobile will be available for deployment of application.
- The application would be developed using Android's OS architectural model.
- Application would run on required mobile device without integration of any extra hardware.
- The project progress would be delivered in specified deliverables after a gap of 3 or 4 weeks.
- We may use some third party components, COTS, to specifically achieve functional or non functional requirements, as is required.
- We may require some initial training (i.e. workshop to familiarize us with the technology).
- Optional components of the project would only be implemented only if the time and required resources are available.
- As now the project scope/core functionalities to achieve have been finalized and approved by the concerned committees, scope may not be further enhanced at any stage during the project construction.

3.7 External Interface Requirements

3.7.1 User Interfaces

User will have to interact with an Android based mobile phone's interface while using the software system. Logically API will cover from unlocking the mobile to the application's core interface. After unlocking the mobile the user will navigate to the text recognition utility from the home screen of the mobile.

The interface of the application includes three parts:

- Capture the picture/image
- Recognize the text from the image
- Use the recognized text in an application

User will press the button to take the picture from the mobile's camera. Camera will remain focused properly on the textual region of the picture ignoring other details of the scene. Text recognition will be done by clicking the recognize text button. Recognized text from the image will be displayed in the text bar.

All the three components of the application will be placed on a single container holding the three controls for the mentioned functionalities. User will have to move sequentially while using the software like, "**Take picture**" then "**Recognize text**" then "**Use application**". While navigating through the mentioned functionalities if a user goes against the intended behavior which is required from the user, application would generate an error message informing the user about the cause of the error.

3.7.2 Hardware Interfaces

The only hardware required is some Data Cables for the interaction of our programming environment (Java) with the physical device (Android based mobile phone) and data sending or retrieving. This would be a wired interaction where one end of the cable would be connected to the mobile device and the other end with the machine where our programming platform resides**.**

3.7.3 Software Interfaces

As the application will be built on Android operating system, so it will definitely use the routines and procedures of the underlying OS. As the implementation of the system will be supported from certain libraries/APIs, for certain tasks to be done, the application will certainly interact with those libraries/APIs.

3.7.4 Communication Interfaces

While implementing the core functionalities the application will purely act as mobile in-house application, which requires no intra or inter-network connection. One of the optional features "**Query recognized text** "require an internet connection with standard communication protocol HTTP, with a compatible browser for browsing and default internet connection settings.

3.8 System Features

3.8.1 Capturing of Image

3.8.1.1 Description and Priority

The foremost step in the development of the system is capturing of the image using an Android based mobile with standard camera. The image will contain textual regions from which the text will be recognized. Priority of this system feature is 3(note: 1-Highest)

3.8.1.2 Stimulus/Response Sequences

User will capture the image using the camera functionality of Android based mobile

3.8.1.3 Functional Requirements

It shall use a camera of standard resolution i.e. within the range of 2-4 Mega pixels as decided according to the specifications and the image must contain textual regions.

Requirement-1: Standard Resolution

Requirement-2: Text Image

3.8.2 Localize text

3.8.2.1 Description and Priority

After capturing of the image in standard resolution, the textual regions within the image are localized. The system is only concerned with the textual regions and complex backgrounds are not within the scope of the project. Priority of this system feature is 4 (note: 1-Highest).

3.8.2.2 Stimulus/Response Sequences

The captured image is processed and the first step in the processing is localization of the textual regions in the image.

3.8.2.3 Functional Requirements

The image must be sharp and it must contain textual regions and the background of the image must be homogeneous.

Requirement-1: Textual regions in the image

Requirement-2: Text on homogeneous background

3.8.3 Recognize text

3.8.3.1 Description and Priority

The core functionality of the system is to recognize the text from image. Priority of this system feature is highest 1.

3.8.3.2 Stimulus/Response Sequence

The localized textual regions are used as input for this system feature and the text (Alphanumeric characters: A-Z, a-z and 0-9) in these regions is recognized.

3.8.3.3 Functional Requirements

The textual regions must be localized and the background must be homogeneous. Text must be printed but isolated handwritten text can optionally be handled. The characters within the localized regions must be alpha numeric. A limited range of fonts must be allowed. The image should not be blurred.

Requirement-1: Localized image

Requirement-2: Text on Homogeneous background

Requirement-3: Printed text

Requirement-4: Alpha numeric characters

Requirement-5: Limited font styles

3.8.4 Application development

3.8.4.1 Description and Priority

At least one application of the system must be developed. The applications which can be developed are:

- Business Card Reader
- Assistance to Visually Impaired Individuals
- Automatic Reading Of Books/Articles
- Translation of Text

The preferred application to be developed is “Translation of text”. Priority of this system feature is 2 (note: 1-Highest)

3.8.4.2 Stimulus/Response Sequences

The image is captured; textual regions are localized and then converted into text. The recognized text is then fed to the application.

3.8.4.3 Functional Requirements

Text must be recognized from the image to develop the applications.

Requirement-1: Text recognition

3.9 Other Nonfunctional Requirements

3.9.1 Performance Requirements

For the best performance of the software user must follow the sequence of the activities to achieve the required results i.e. do not proceed to recognize text before the picture is captured. While using the software, user's action must be consistent and unique. Input to the software must be in the required format.

3.9.2 Safety Requirements

While the software is executing a particular task, user must wait for the undergoing task to complete and must not interrupt otherwise the application may crash or output may be affected. Incompatible input or format may also result the crash of complete application or a part of it.

3.9.3 Security Requirements

Mobile applications developed in Android's platform require proper application security signature and certificate when application is deployed to the market or end user. Usually '.apk' file contains the signature and certified version of the application. Our application also requires the mentioned security and certification requirements.

3.9.4 Software Quality Attributes

Some of the quality attributes identified includes:

- **Portability**-in API, portability can be defined as "compatibility of application with platform (Android's version) upgraded or downgraded versions. In Android's platform when an up gradation is done, application requires some

changes for compatibility with new version. As android's OS is backward compatible so no changes are required in down gradation

- **Maintainability**-Whenever there is a change in requirement or bug found, the application will be easily maintainable.
- **Adaptability**-as already mentioned that android OS is backward compatible, for forward compatibility some changes in the implementation are required.
- **Availability**-the application will be available 24/7, provided mobile is in working state and application is configured properly.
- **Flexibility**-the layout/architecture of the application will be flexible enough for some later requirements change or application enhancement.
- **Usability**-the presentational features of the application will be designed user friendly with minimum training required to use the application.
- **Reliability**-the system will be designed to provide maximum reliability.

3.10 Business Rules

Roles of individuals in application development include:

- **Developers**- individuals who are developing the application
- **Supervisor-**an individual supervising our project.
- **Project coordinator-**an individual who coordinates all the matters regarding projects with the concerned authorities
- **Testing team-**a team of individuals which tests implemented projected against the agreed requirements

3.11 Other Requirements

All the requirements have been already stated in SRS.

CHAPTER 4

SYSTEM DESIGN SPECIFICATION

4. Architectural Design

4.1 System Block Diagram

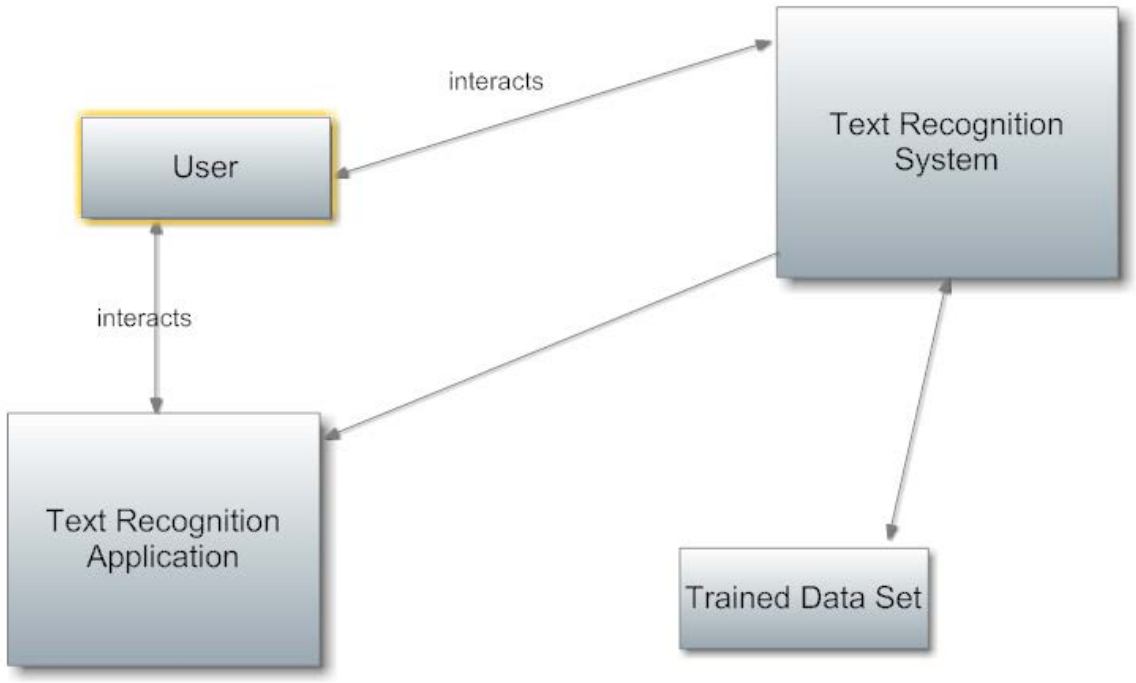


Figure 11: System Block Diagram

Before going into the detail of architectural design we here present logical division of project into modules/parts/subparts and different useful views.

4.3 Software Components

Our project contains following software components:

- **Operating Systems**
    - Linux(Ubuntu11.10)
    - Windows XP

- Android OS

➢ **Software Packages**

- Android SDK(Software Development Kit)
- Eclipse IDE(Integrated Development Environment)
- Java SDK 1.6

4.4 Hardware Components

➢ Personal Computer(s)
➢ Mobile Phone Having Android as OS and a 2-4Mpx Camera
➢ Connecting Cables

4.5 High Level Design Diagram (Modules Identification)

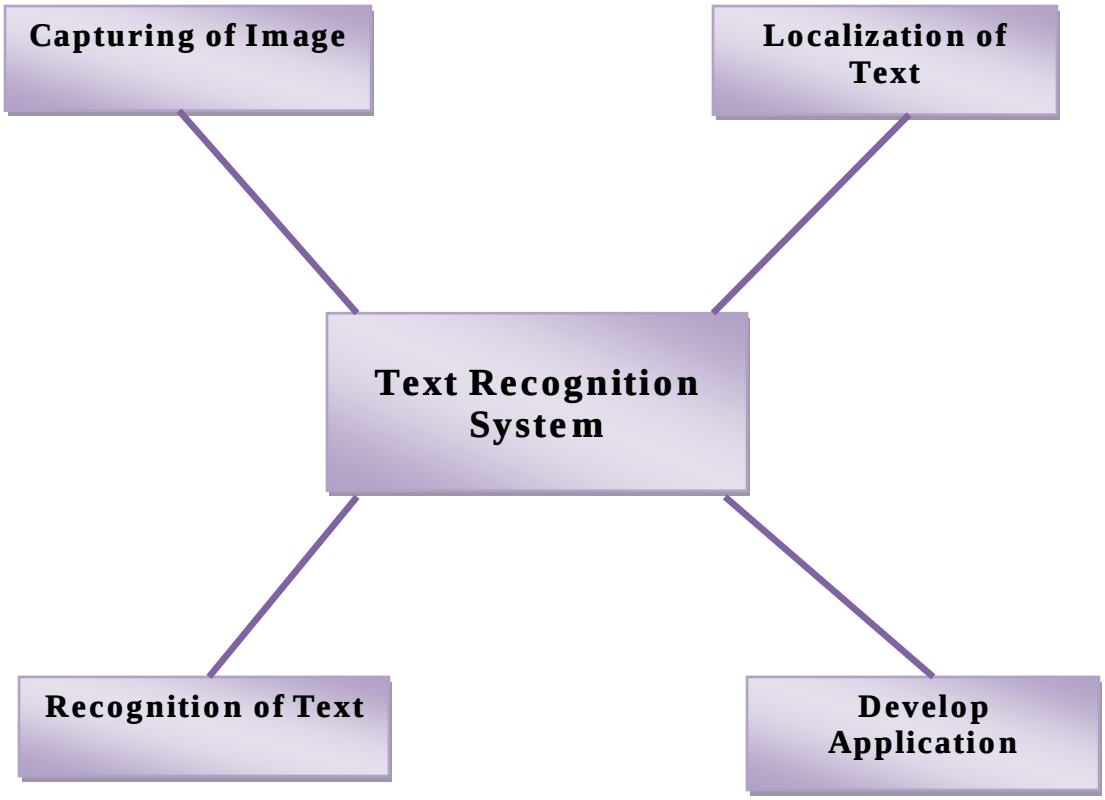


Figure 12: High Level Design

4.6 Interaction among Modules (Abstract View)

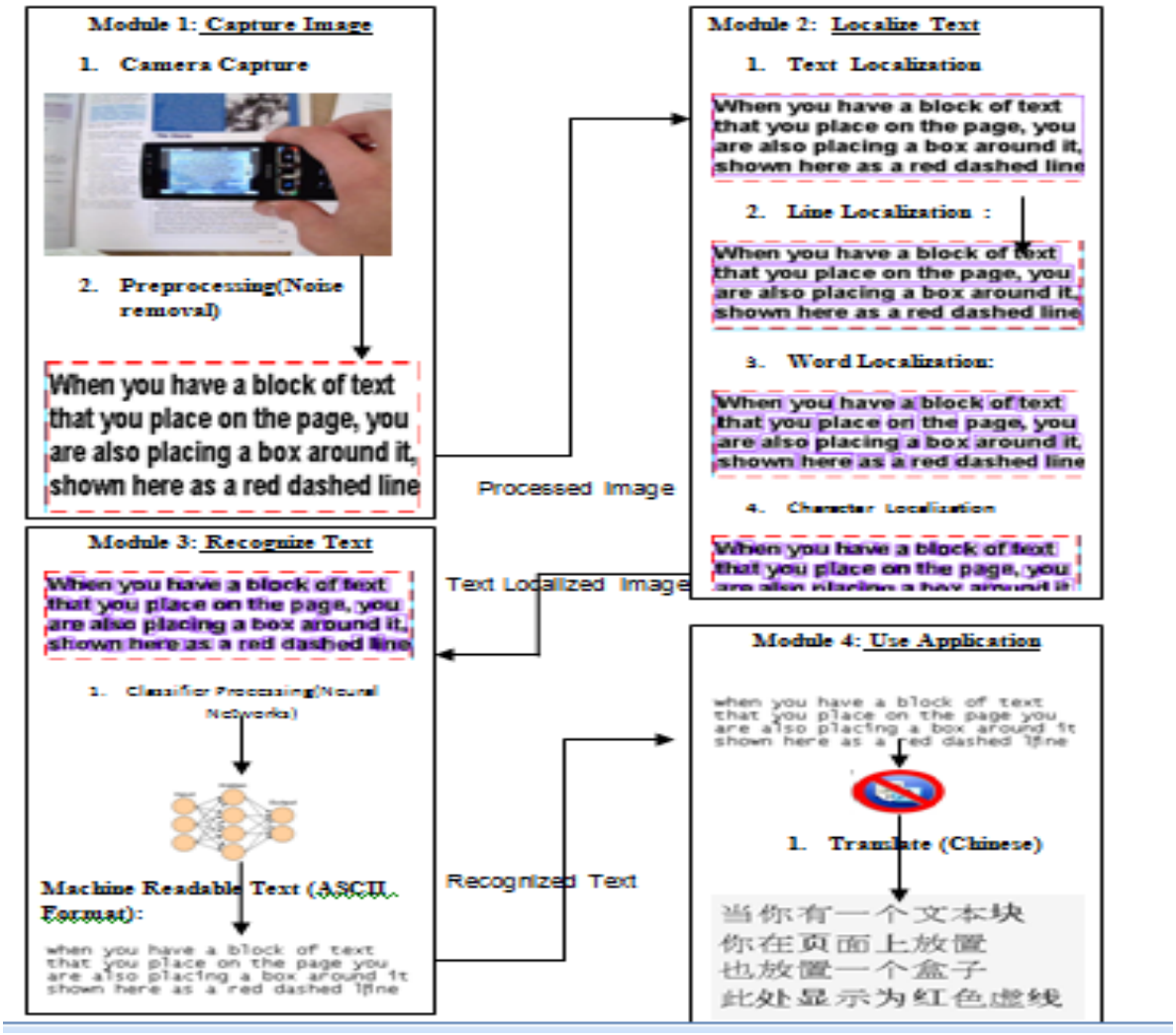


Figure 13: Interaction among Modules- An Abstract View

4.7 Architectural Style

Architecture of text recognition system can be modeled using **Model-View-Controller** Interface of the system is distinct from the application logic. Interface forms the View of the architecture. Model has the application data .Controller controls the coordination between the view and the model. There is no direct communication between model and the view all the communication is directed using the controller. The system is divided

into modules as per the main functionality of the system. These functionalities can be used separately as off-the shelf components .Interface of this system has no application logic embedded in it so the system architecture can easily be made using Model View Controller approach. User class which has the functions related to the GUI forms the **View**. Pressing_CaptureImage_Button(),Pressing_RecognizeText_Button() and Pressing _UseApplication_Button are the functions of User class .These function handle the processing of interface but no application logic is involved in it.

Controller in this system is usecase handlers as the system functionality is clearly divided in modules. But for reusability of the system a façade controller will be a bad choice so usecase handler controllers are used. CaptureImageHandler is used for the CaptureImage module, RecognizeTextHandler is used for the RecognizeText module and UseApplicationHandler is used for the UseApplication module. These UsecaseHandlers only invoke the functions of their respect Modules. Controller only invokes the functions but do no processing.

Model in this system is the CaptureImage , RecognizeText, UseApplication class which is encapsulating the application logic. These classes are composing the other classes which have the distributed functionality. These classes process the requests made by the view through the controller. After processing Model gives the retrieved results back to

controller.

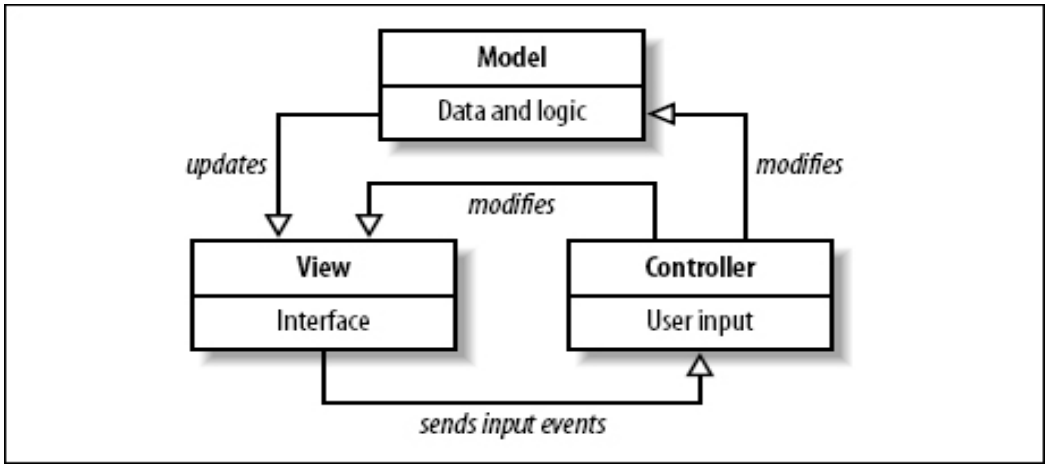


Figure 14: Model View Controller – Generic Interaction

4.8 Detailed Design

4.8.1 Database Diagram

4.8.1.2 Entity Relationship Diagram

Text recognition System requires trained data set which contains collection of textual images on which training is done so that each character can be labeled. This training is done on a limited number of fonts so recognition can also be done only on these fonts. Extracted characters from the captured image are labeled according to this data set.

ERD of the system is given below

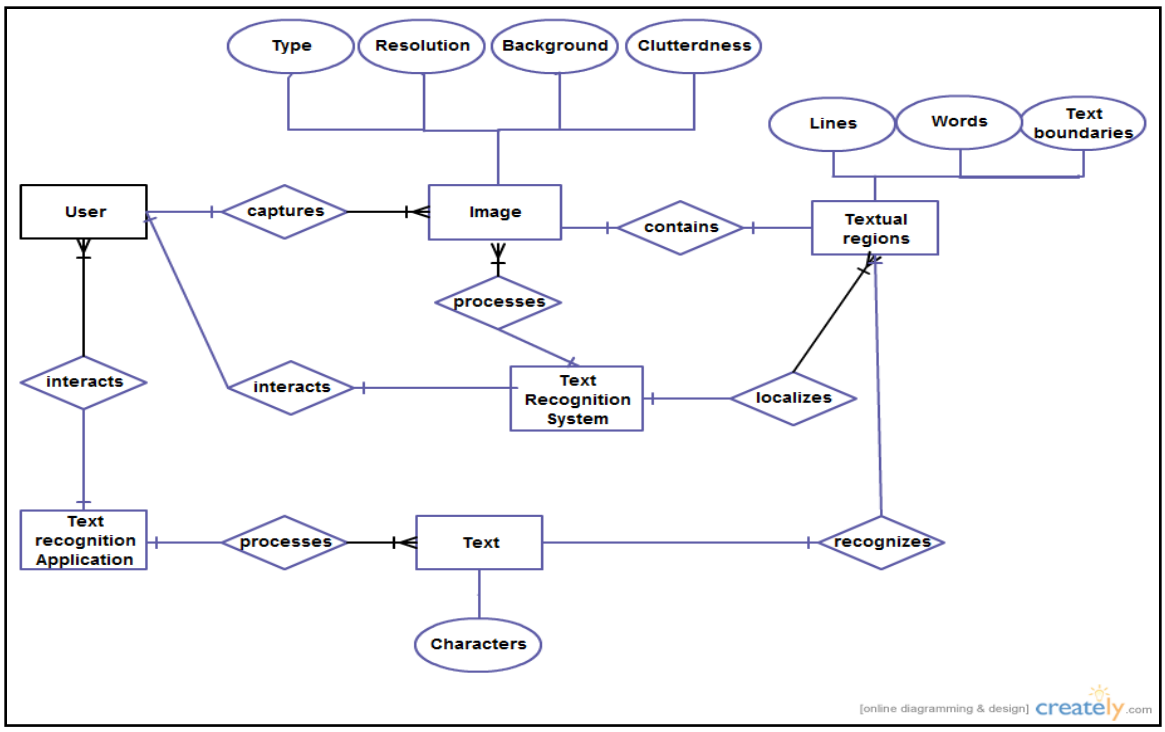


Figure 15: Entity Relationship Diagram

Entities & Attributes

- **User**
- **Image**
  - Resolution (must be within the range of 2-4 Mpixels)
  - Clutterdness (must be less cluttered)
  - Background (must have homogenous background)
  - Type(gif, jpeg etc)
- **Textual regions**
  - Text Boundary

- Lines
- Words

➢ **Text**

- Characters

➢ **Text Recognition System**

➢ **Text Recognition Application**

4.9 UML Diagrams

4.9.1 Usecase Diagram

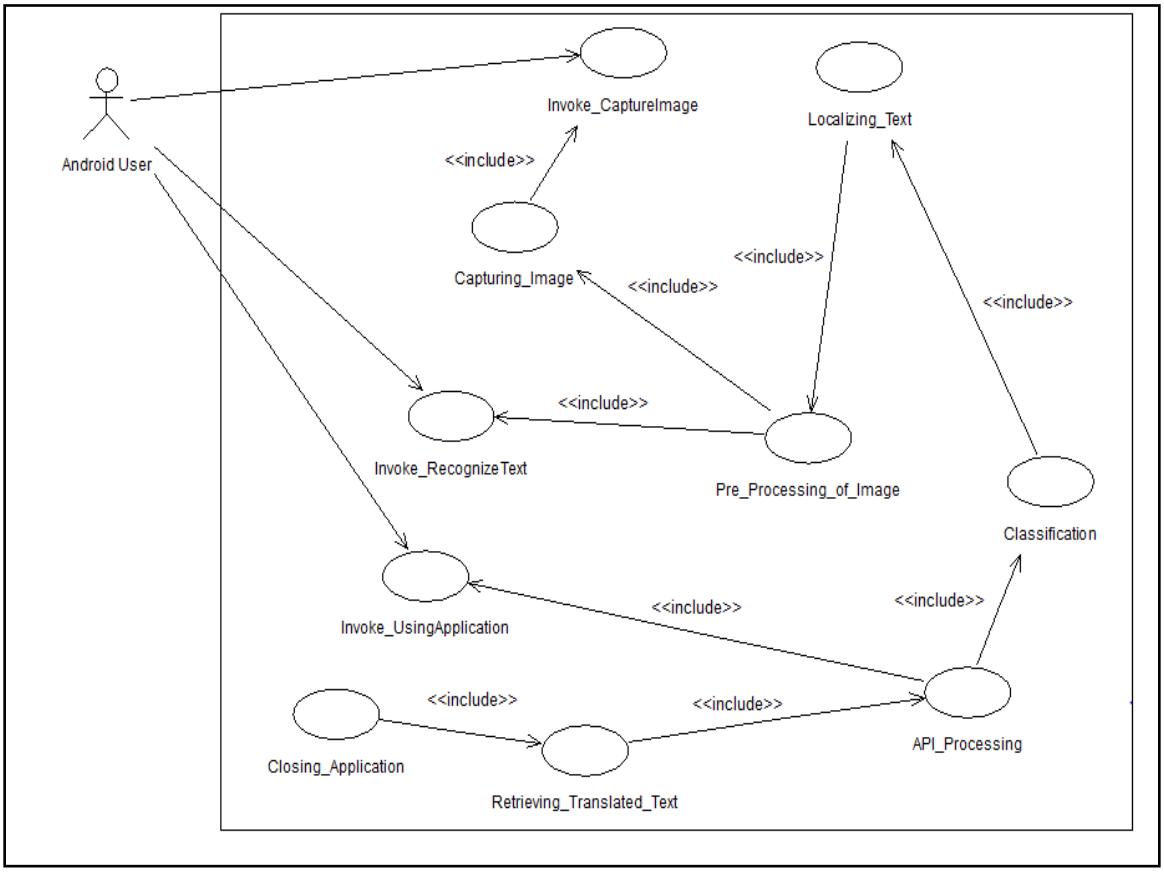


Figure 16: Usecase Diagram

4.9.1.1 Usecase Specifications

4.9.1.1.1 Actors

- Android mobile users

4.9.1.1.2 Use Cases

- Invoke_CaptureImage
- Invoke_RecognizeText
- Invoke_Use Application
- Capture_Image
- Processing_Image
- Localizing_Image
- Classification
- API_Processing
- Retrieving_Translated_Text
- Closing_ Application

a) Invoke_CaptureImage

1. Brief Description

This use case describes the whole processing after the event when user click/taps Capture Image button.

2. Actors

Android mobile users

3. Pre-conditions

- Mobile must be in the working state.
- Camera functionality of mobile phone is available and working.

4. Basic Flow of Events

This use case starts when the user clicks/taps the capture image button.

1. Camera of mobile phone will automatically open up.
2. Camera will show the image which is being focused by the camera eye.
3. Adjust the image if required by clicking/taping auto-adjust functionality of camera.
4. User clicks/taps the camera icon.
5. Image will be captured.

5. Alternative Flows

5.1 Missing Functionality

- If in step 1 of the basic flow i.e. pressing the capture image button does not open the camera the use case ends with a failure condition
- If clicking camera icon doesn't capture image then use case ends with a failure condition

5.2 Wrong Functionality

If the picture is not captured successfully then use case ends with the failure condition.

6. Key Scenarios

N/A

7. Post-conditions

7.1 Successful Condition

If use case was successful, Image will be captured successfully.

7.2 Failure Condition

If not, image will not be captured.

8. Special Requirements

- Image must be text based
- Resolution of camera must be within the range of 2-4 MPixels.

b) Invoke_RecognizeText

1. Brief Description

This use case describes what happens when user clicks Recognize Text button and how system processes captured image.

2. Actors

- Android mobile users

3. Pre-conditions

- Mobile must be in working state.
- Image has been captured.

4. Basic Flow of Events

This use case starts when the user clicks/taps the Recognize Text button.

Following is the sequence of action:

1. Processing of captured image will automatically start.
2. Noise will be removed from the image if required.
3. Brightness will be adjusted if required.
4. Blurriness will be reduced if required.
5. Textual regions will be localized.
6. Lines will be extracted from the textual regions.
7. Words will be extracted from the line within the textual regions.
8. Characters will be extracted from words.
9. Characters are labeled according to trained data set.
10. Recognized text will be available.

5. Alternative Flows

5.1 Missing Functionality

- If in step 1 of the basic flow i.e. pressing the recognize text button does not start the processing of image automatically then use case ends with a failure condition.
- If the system is not providing the functionality for noise removal, brightness adjustment and blurriness reduction then use case will end with a failure condition.
- If the system does not provide functionality for localizing the textual regions, use case will end with the failure condition.

- If the system does not provide functionality for classification, use case will end with the failure condition.

5.2 Wrong Functionality

- If picture is not processed successfully then use case ends with the failure condition.
- If textual regions are not localized successfully then use case ends with the failure condition.
- If text is not recognized successfully then use case ends with the failure condition.

6. Key Scenarios

N/A

7. Post-conditions

a. Successful Condition

If use case was successful, text will be recognized successfully.

b. Failure Condition

If not, Text will not be recognized.

8. Special Requirements

- Image must be text based.
- Background of image must be homogenous.

c) Invoke_UseApplication

1. Brief Description

This use case describes what happens when user clicks Use Application button and how application uses recognize text.

2. Actors

- Android mobile users

3. Pre-conditions

- Mobile must be in working state.
- Internet connection must be available.
- Text must be recognized from image.

4. Basic Flow of Events

This use case starts when the user clicks/taps the Use Application button.

Following is the sequence of action:

1. Application will open up.
2. API processing of recognized text will start automatically.
3. Recognized text will be passed to the internet for browsing the translated text in a desired language.
4. Translated text will be available.

5. Alternative Flows

5.1 Missing Functionality

- If in step 1 of the basic flow i.e. pressing the Use Application button does not open the application then use case ends with a failure condition.

- If API processing does not start automatically then use case ends with a failure condition.
- If the system does not provide functionality for using the application, use case will end with the failure condition.

5.2 Wrong Functionality

If application does not perform in accordance with the desired functionality then use case ends with a failure condition

6. Key Scenarios

N/A

7. Post-conditions

a. Successful Condition

If the use case was successful, translated text will be available and application will be closed.

b. Failure Condition

If not, text will not be translated.

8. Special Requirements

N/A

4.9.2 Sequence Diagrams of key use cases

a) Use case: Invoke _CaptureImage

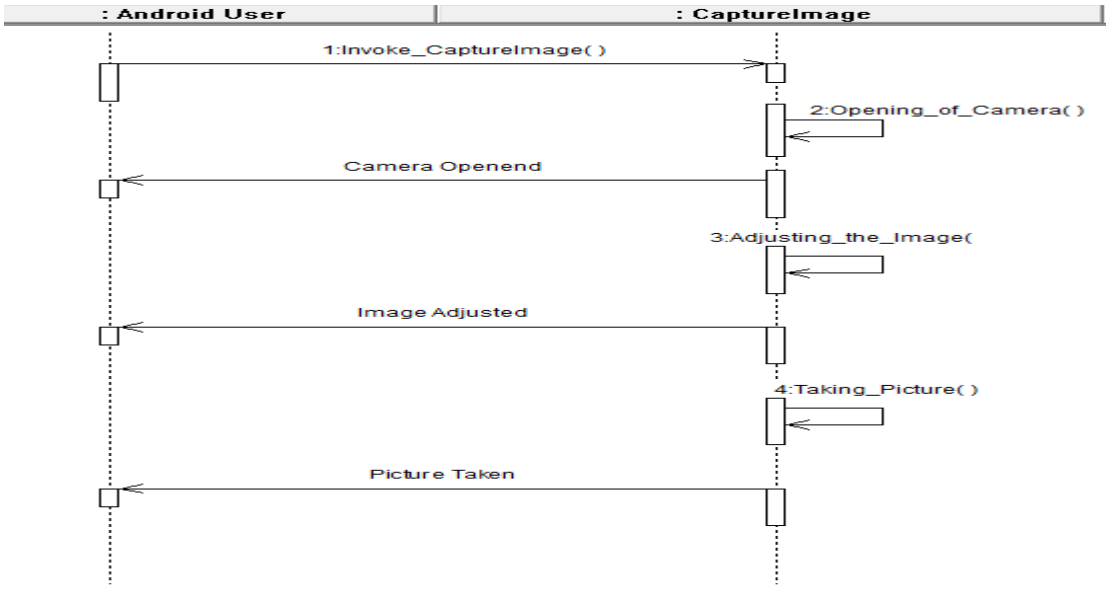


Figure 17: Sequence Diagram of Invoke_CaptureImage

b) Use case: Invoke_RecognizeText

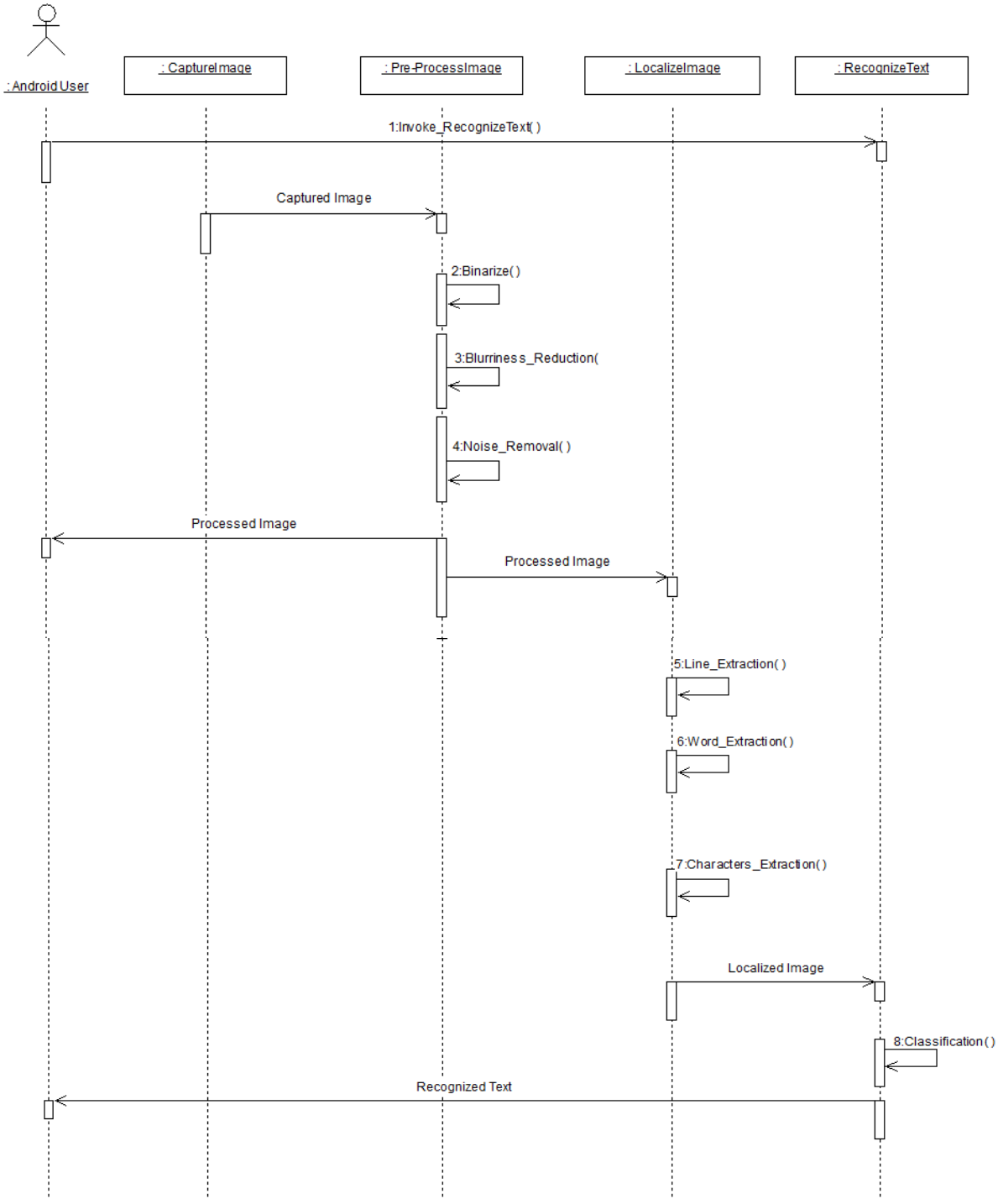


Figure 18: Sequence Diagram of Invoke_RecognizeText

c) Use case: Invoke_UseApplication

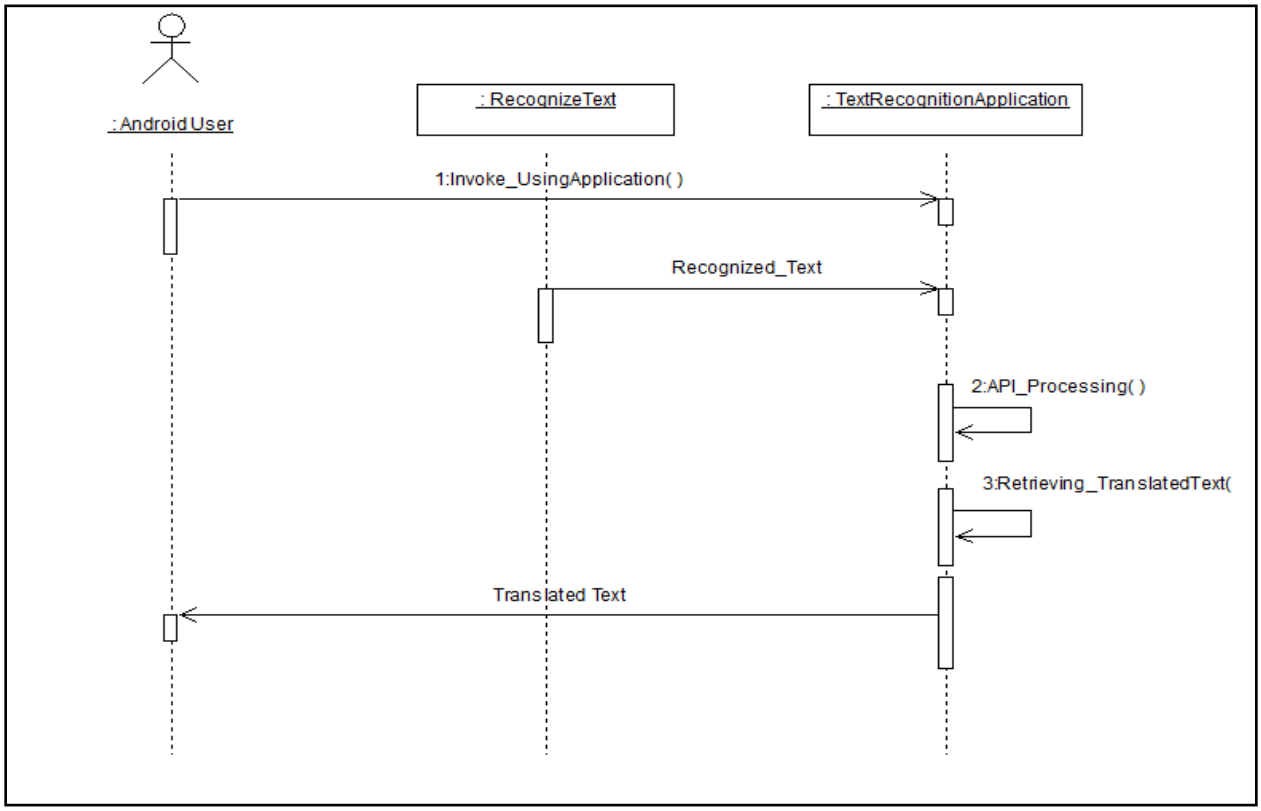


Figure 19: Sequence Diagram of Invoke_UseApplication

4.9.3 Collaboration Diagrams of key use cases

a) Use case: Invoke_CaptureImage

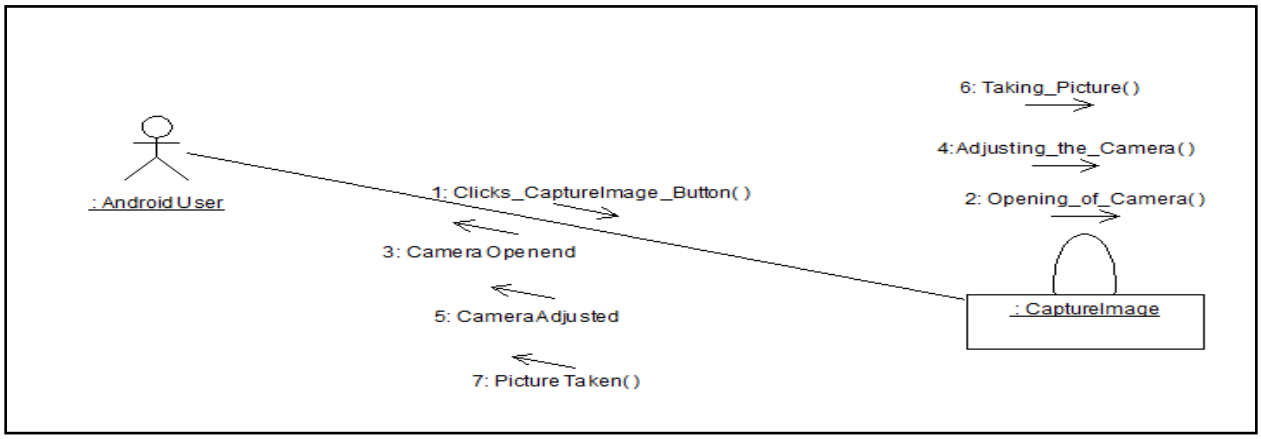


Figure 20: Collaboration Diagram of Invoke_CaptureImage

b) Use case: Invoke_RecognizeText

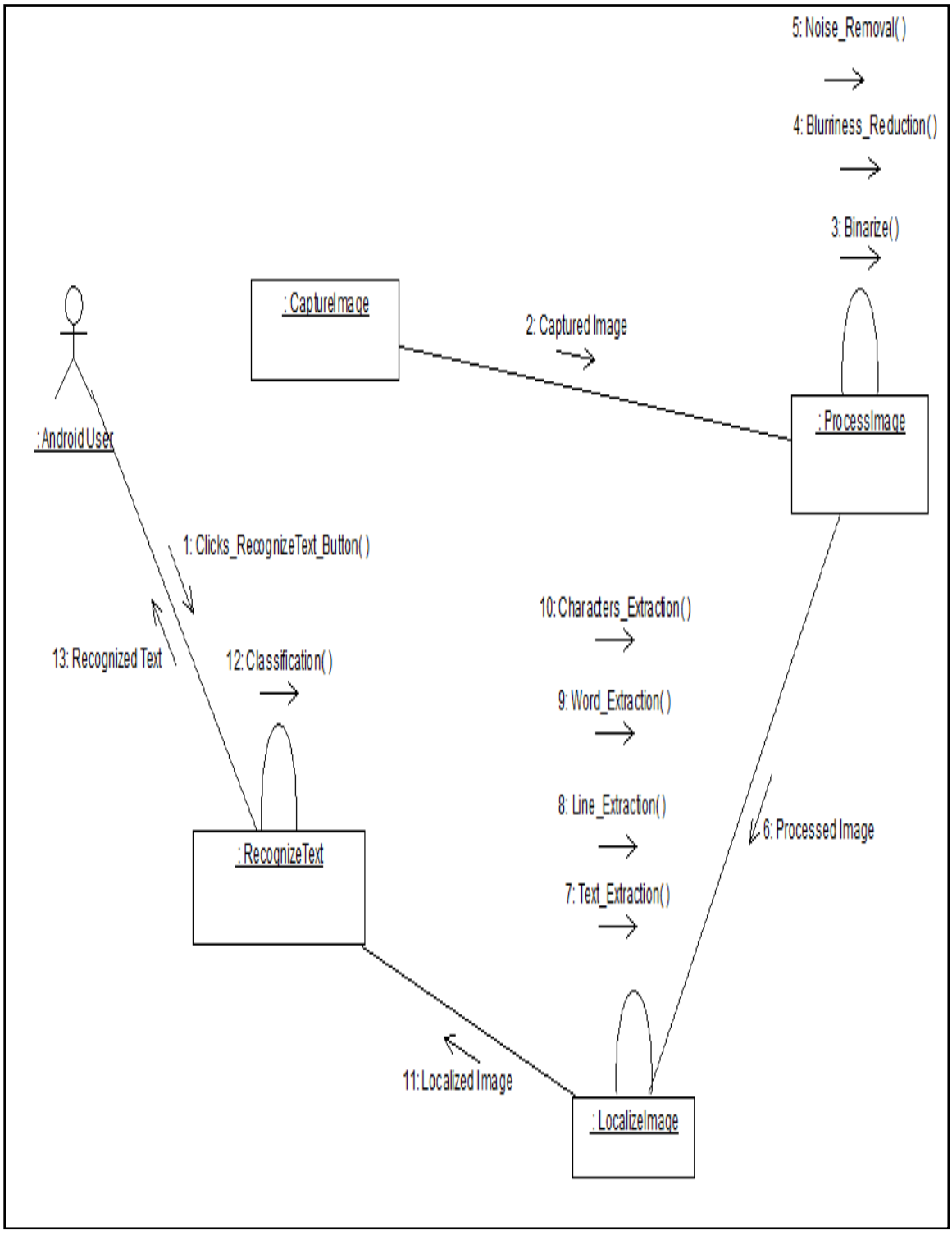


Figure 21: Collaboration Diagram of Invoke_RecognizeText

c) Usecase: Invoke_UseApplication

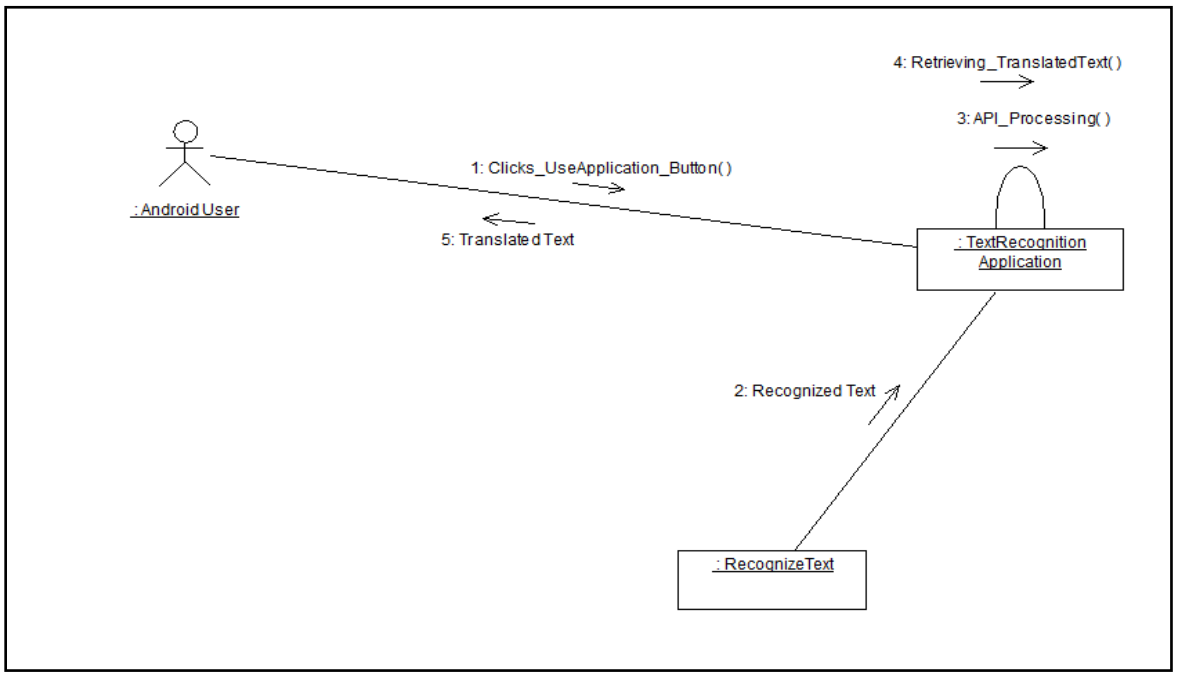


Figure 22: Collaboration Diagram of Invoke_UseApplication

4.10 Logical View

4.10.1 State Transition Diagrams

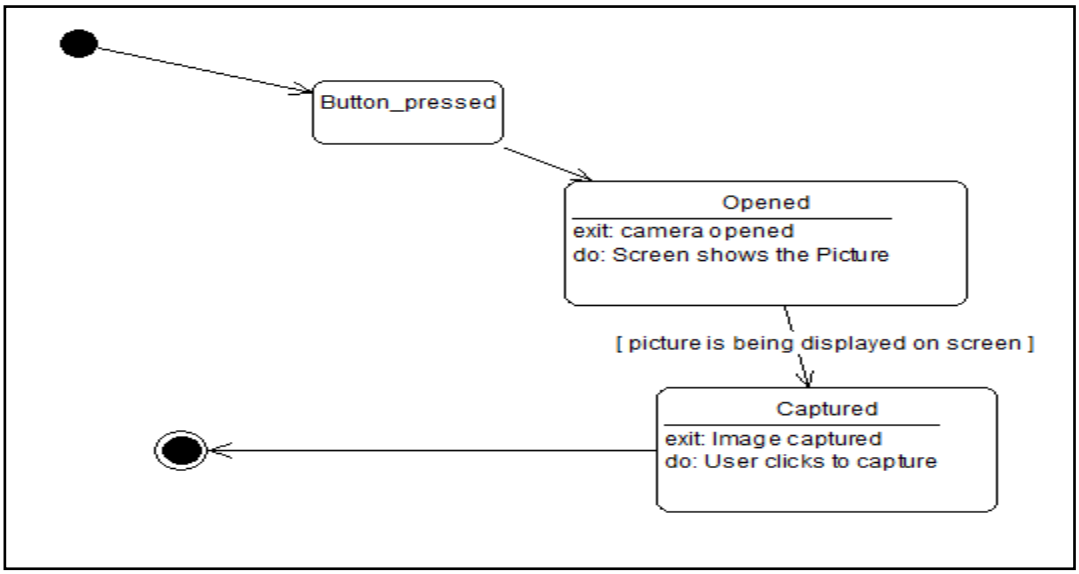


Figure 23: State Transition Diagram of CaptureImage

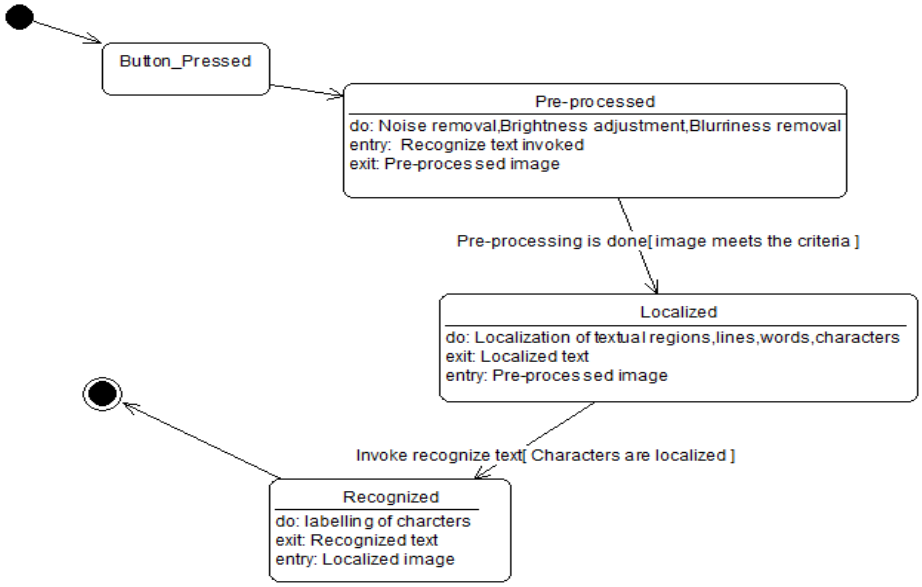


Figure 24: State Transition Diagram of RecognizeText

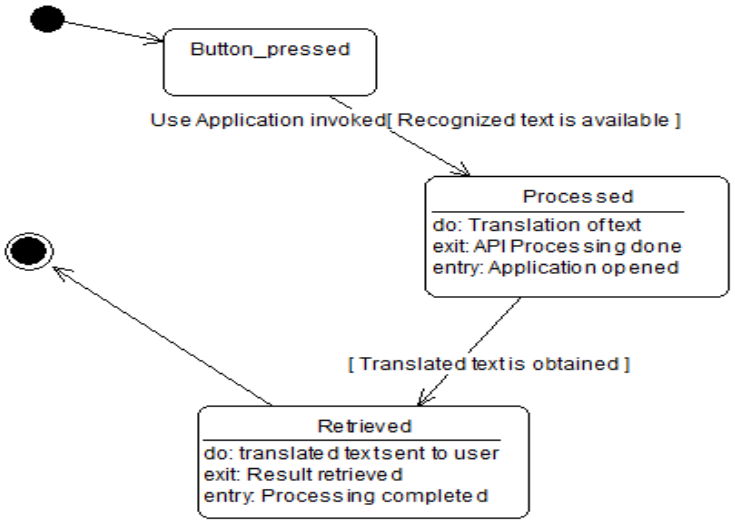


Figure 25: State Transition Diagram of UseApplication

4.11 Dynamic View

4.11.1 Activity diagram

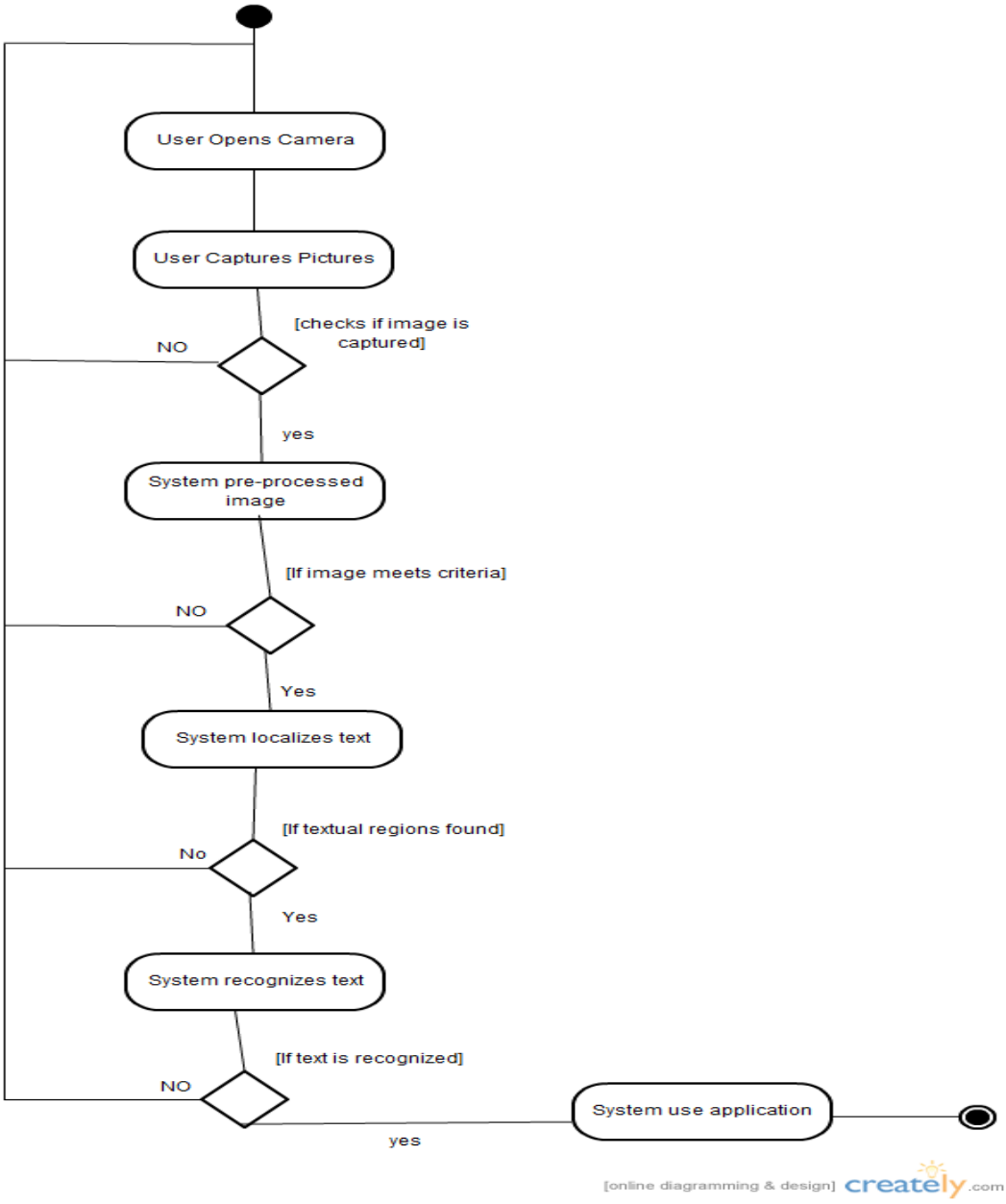


Figure 26: Activity Diagram

4.11.2 Data Flow Diagram

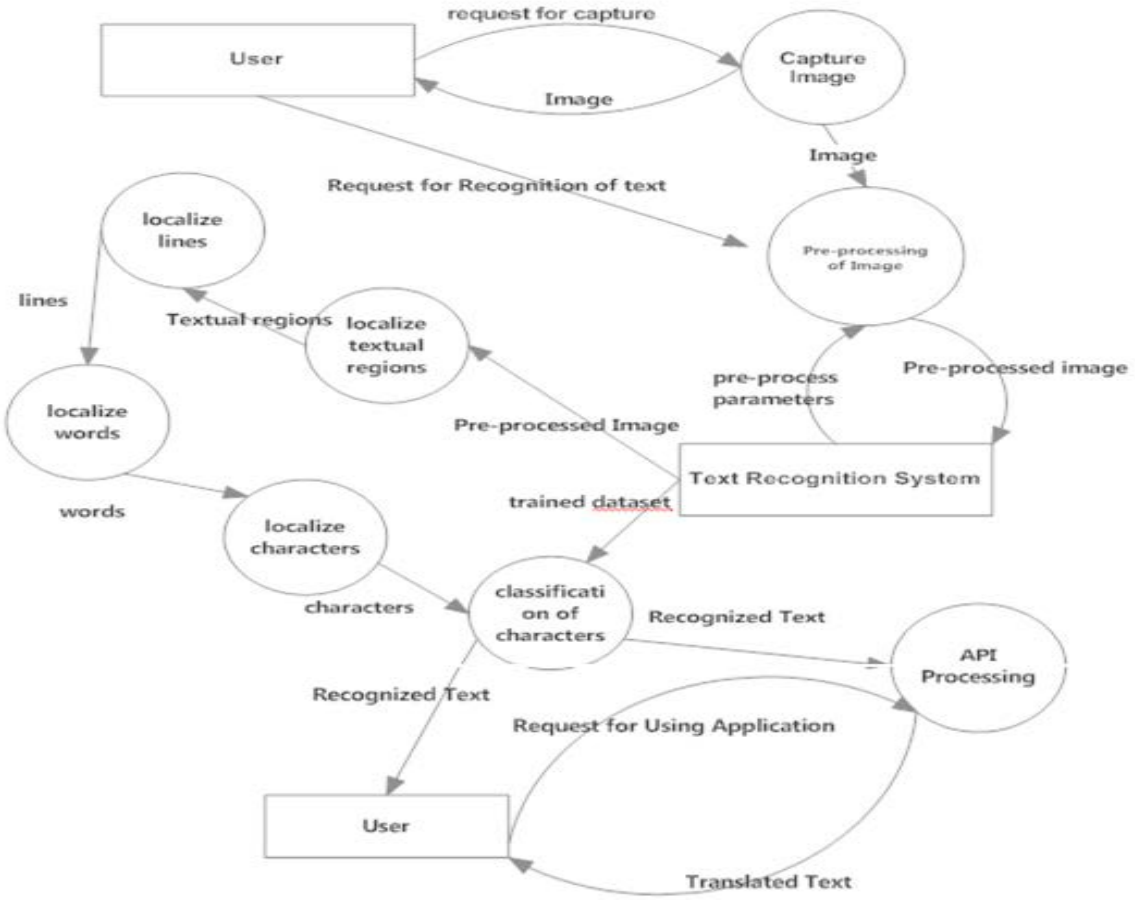


Figure 27: Data Flow Diagram

4.12 Implementation View

4.12.1 System Class Diagram

Text recognition system comprises of following classes:

- User
- Capture Image
- Process Image
- Localize Text

- Recognize Text
- Text Recognition Application
- Train Data Set

Class Diagram is given below

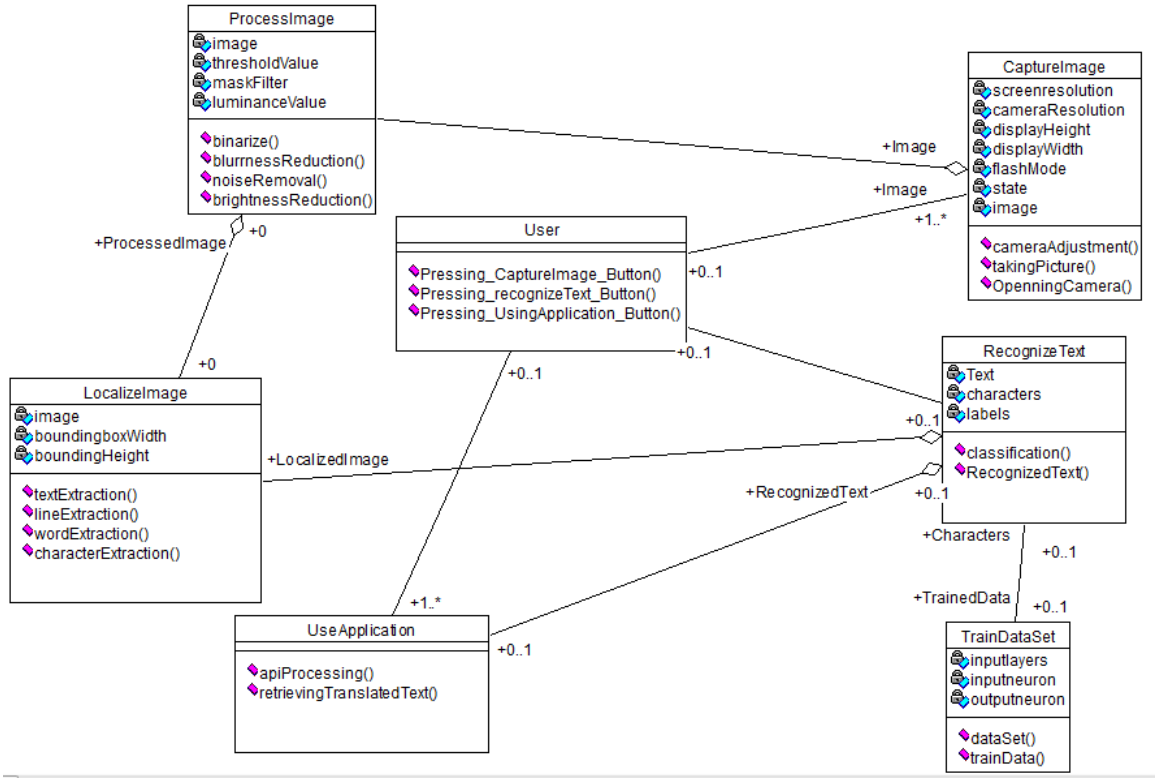


Figure 28: Class Diagram

4.12.1.1 System Classes Description

Name	Description
User	This class contains the functions which are directly related to the UI.Pressing_CaptureImage_Button() is called when an external user presses the Capture Image button().This function then invokes the functions in the CaptureImage class which processes the request. Pressing_RecognizeText_Button() is called when an external user presses the Recognize Text button.This function then invokes the functions in the RecognizeText class which processes the request. Pressing_UsingApplication_Button is called when an external user presses the Using Application button.This function then invokes the functions in the UseApplication class which processes the request
CaptureImage	This class contains OpenningCamera( ),takingPicture( ) and adjustingCamera( ).This class is solely responsible for the capturing of image.
ProcessImage	The main role of this class is to pre-process the image.Pre-processing is done by binarize( ),blurrinessReduction() ,noiseRemoval( ),brightnessAdjustment( ).This pre-processing is done so that the image meets the criteria for further processing.It takes the image from the CaptureImage class.
LocalizeImage	This class localizes the textual regions,lines,words,characters from the image using its functions textExtraction( ), lineExtraction( ), wordExtraction( ), characterExtraction( ) respectively.These functions take the output of the previous function as input like lines are extracted from the textual regions.This class uses the pre-

	processed image from the ProcessImage class for processing.
RecognizeText	This class matches the characters which are taken from the LocalizeImage class with the trained dataset from the TrainDataSet class and then these characters are labelled accordingly.This is done in Classification( ) function.And then the final recognized text is sent to the user by the function RecognizeText( ).This recognized text is then used by the UseApplication class.
UseApplication	This class contains all the functionality of processing related to the application of text recognition system.In this class apiProcessing( ) function handles all the processing on the recognized text retrieved from the RecognizeText classs .Then the translated text is then passed to the user in the function retrievingTranslatedText( ).
TrainDataSet	This class contains the training dataset and processing on it so that the extracted characters can be labelled according to the trained dataset.

Table 2: Class Description

4.12.2 Package Diagram

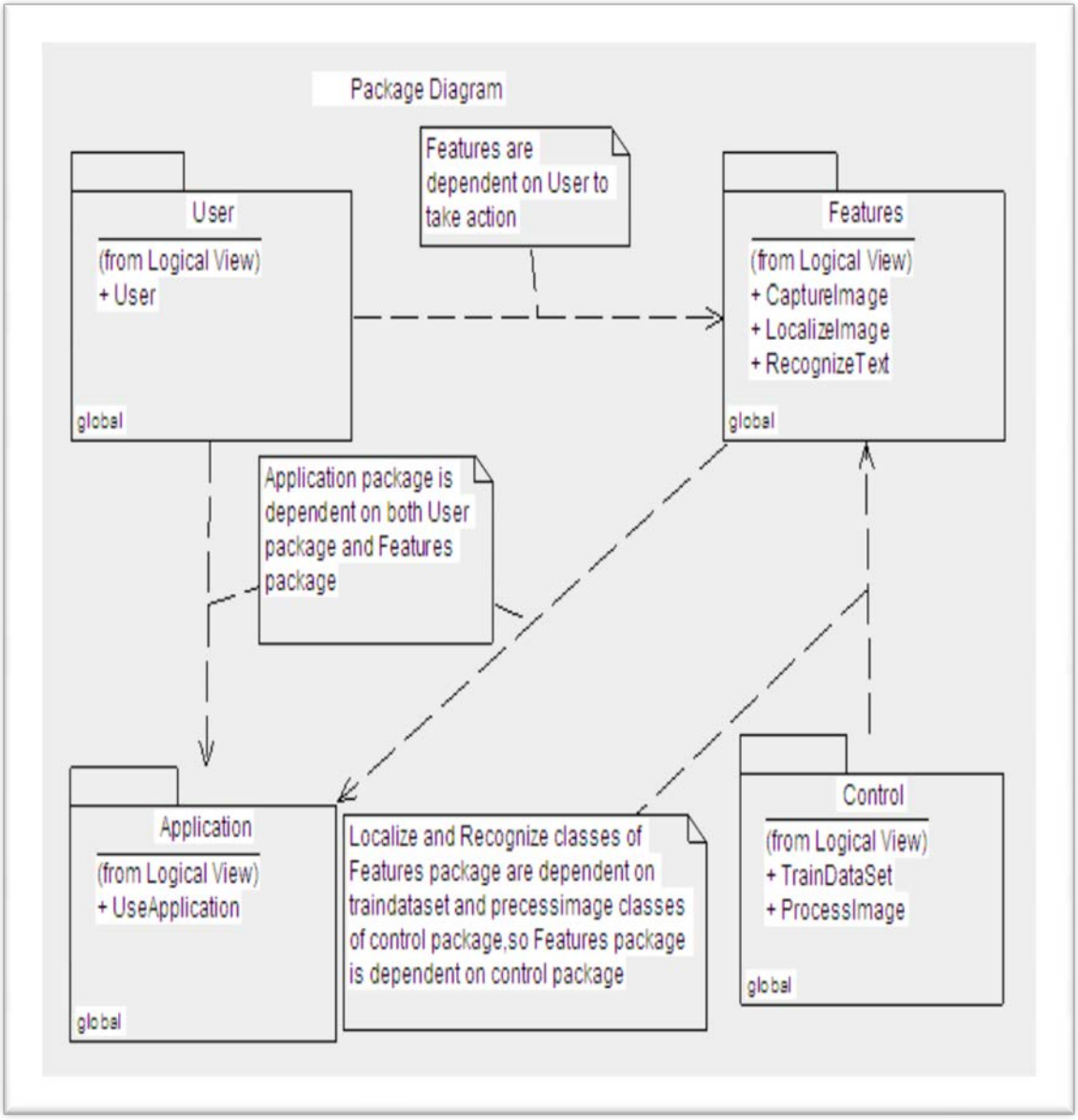


Figure 29: Package Diagram

4.13 Design Patterns

Name: Controller Pattern

Problem: What first object beyond the UI layer receives and coordinates ("controls") a system operation (Capture Image)?

Solution:

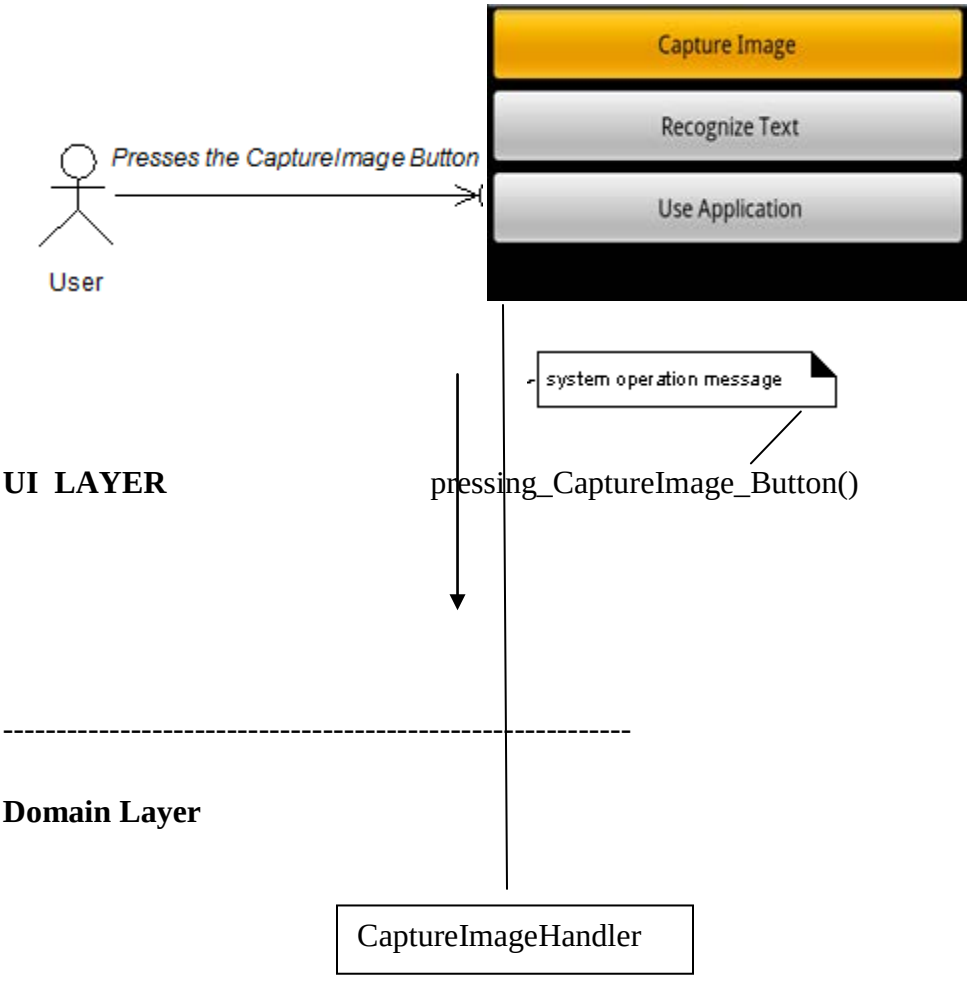
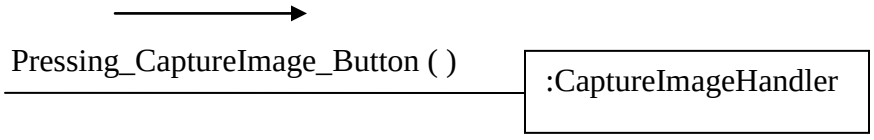


Figure 30: Design Pattern Application-Capture Image

Whenever an input system event is generated by pressing the CaptureImage Button which is in the **View** of the system, request for the processing will be made to the **Controller** which in this scenario is CaptureImageHandler - a usecase controller class. CaptureImageHandler will then forward this request to the CaptureImage class which is **Model** in this case, will then process the request using its functions and data. After processing Model will return the result to View which in this case will be an image.

Processing in the Model includes opening of the camera feature of mobile, adjusting the image if required and then capturing it at last.

By using this Controller pattern interface logic will be separated from the business logic .It will increase the potential for reuse and pluggable interfaces. It also provides an opportunity to reason about the state of the use case which is done by the controller which allows the state information to be saved in the usecase handler. It provides low coupling, high cohesion and high reusability.



Name: Controller Pattern

Problem: What first object beyond the UI layer receives and coordinates ("controls") a system operation (Recognize Text)?

Solution:

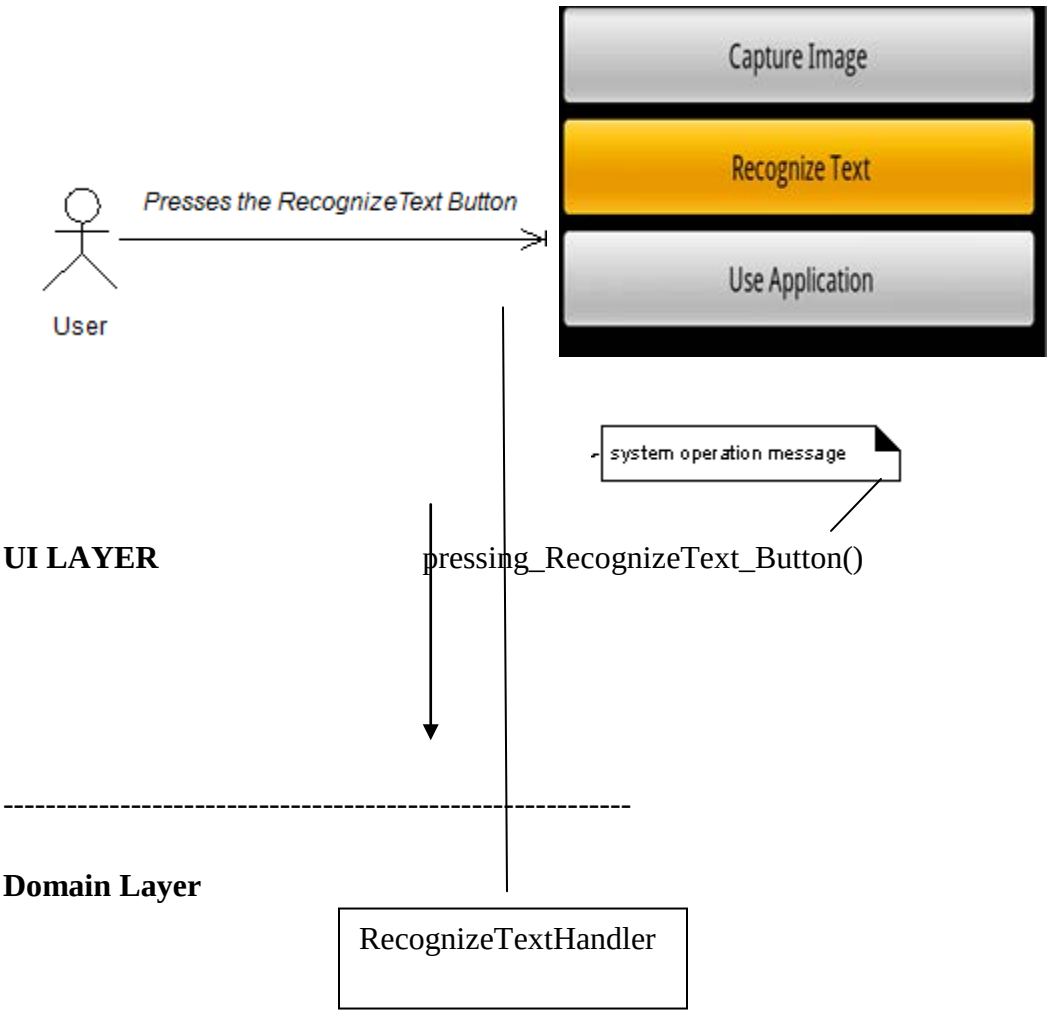
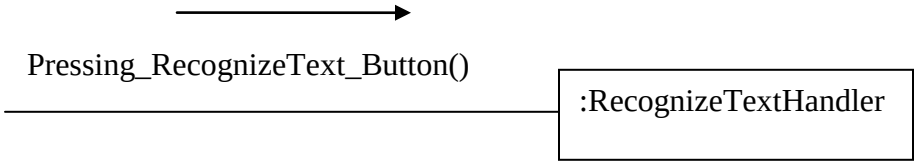


Figure 31: Design Pattern Application – Recognize Text

Whenever an input system event is generated by pressing the RecognizeText Button which is in the **View** of the system, request for the processing will be made to the **Controller** which in this scenario is RecognizeTextHandler - a usecase controller class. RecognizeTextHandler will then forward this request to the RecognizeText class which is **Model** in this case, will then process the request using its functions and data.

RecognizeText class first pre-processes the image and then localizes the textual regions within that image. Lines are extracted from the localized regions and later these lines are used to extract words, from which the characters are extracted. The characters are then labeled using the trained dataset. After processing by the Model it will send the result back to the controller which will pass it to the UI layer.

By using this Controller pattern interface logic will be separated from the business logic .It will increase the potential for reuse and pluggable interfaces. It also provides an opportunity to reason about the state of the use case which is done by the controller which allows the state information to be saved in the usecase handler. It provides low coupling, high cohesion and high reusability.



Name: Controller Pattern

Problem: What first object beyond the UI layer receives and coordinates ("controls") a system operation (Use Application)?

Solution:

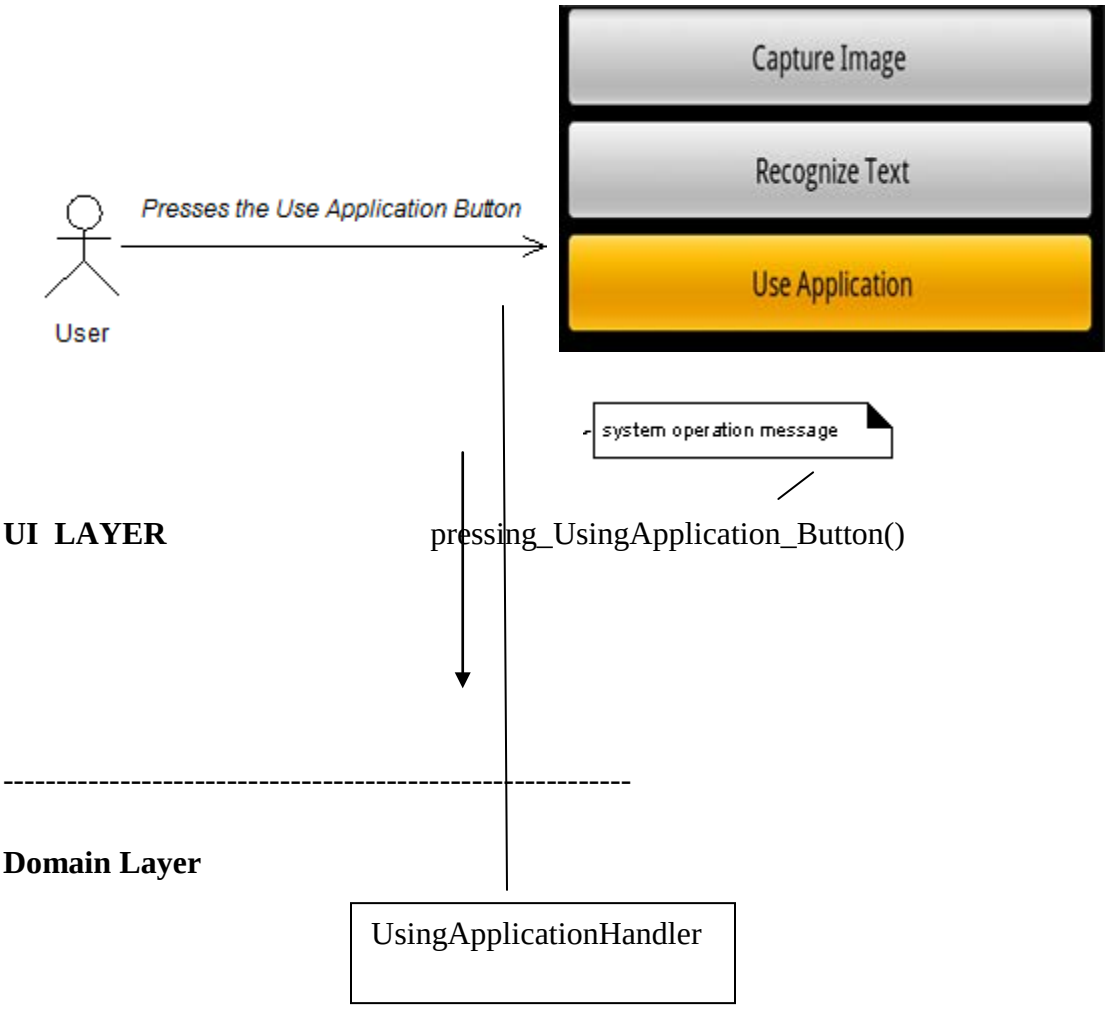


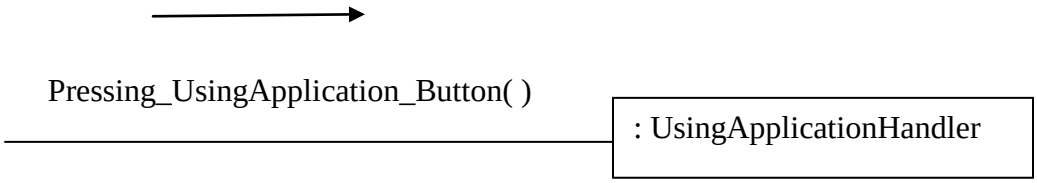
Figure 32: Design Pattern Application – Using Application

Whenever an input system event is generated by pressing the UsingApplication Button which is in the **View** of the system, request for the processing will be made to the **Controller** which in this scenario is UsingApplicationHandler - a usecase controller. UsingApplicationHandler will then forward this request to the UsingApplication class which is **Model** in this case, will then process the request using its functions and data.

After processing by the Model it will send the result back to the controller which will then forward the retrieved result to the UI layer.

UseApplication uses the recognized text from the RecognizedText class and browse it on internet for the translation and send the result to the controller which will pass it to the UI layer

By using this Controller pattern interface logic will be separated from the business logic .It will increase the potential for reuse and pluggable interfaces. It also provides an opportunity to reason about the state of the use case which is done by the controller which allows the state information to be saved in the usecase handler. It provides low coupling, high cohesion and high reusability.



CHAPTER 5

SYSTEM IMPLEMENTATION

5. System Implementation

5.1 Text Recognition Process: An Overview

5.1.1 Methods of OCR

The main principle in automatic recognition of patterns is first to teach the machine which classes of patterns that may occur and what they look like. In OCR the patterns are letters, numbers and some special symbols like commas, question marks etc., while the different classes correspond to the different characters. The teaching of the machine is performed by showing the machine examples of characters of all the different classes. Based on these examples the machine builds a prototype or a description of each class of characters. Then, during recognition, the unknown characters are compared to the previously obtained descriptions, and assigned the class that gives the best match. In most commercial systems for character recognition, the training process has been performed in advance. Some systems do however include facilities for training in the case of inclusion of new classes of characters.

5.1.2 Components of an OCR system

A typical OCR system consists of several components. In figure 33 a common setup is illustrated. The first step in the process is to digitize the analog document using an optical scanner. When the regions containing text are located, each symbol is extracted through a segmentation process. The extracted symbols may then be preprocessed, eliminating noise, to facilitate the extraction of features in the next step.

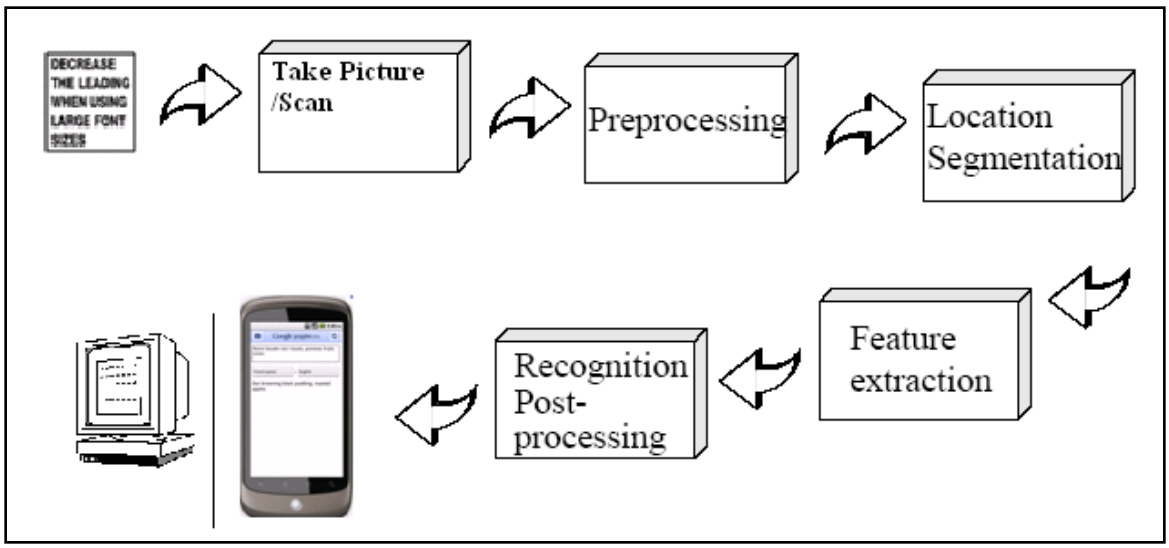


Figure 33: A common setup of OCR process

The identity of each symbol is found by comparing the extracted features with descriptions of the symbol classes obtained through a previous learning phase. Finally contextual information is used to reconstruct the words and numbers of the original text.

5.1.3 Our Methodology

The methodology used for text recognition is illustrated in Figure 35. It basically consists of a combination of various image processing, feature extraction and machine learning based steps. Each step of the methodology is explained in detail subsequently in this chapter

5.2 Procedure

5.2.1 Take Picture/Scanning

Through mobile camera digital image of the original document is captured. As the quality of the Image and hence resolution are widely dependent on camera. Normally camera

based hand held devices have low resolution hence quality of the image is low as well as have more probability of the introduction of noise in the image.

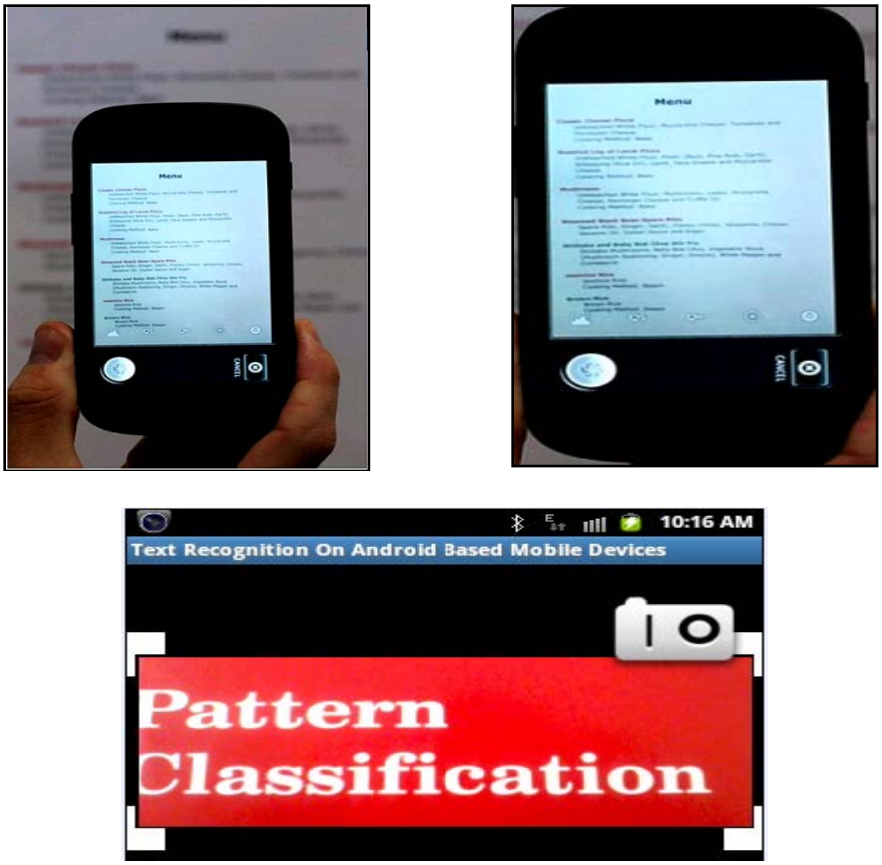


Figure 34: Various instances of taking picture using Mobile's camera

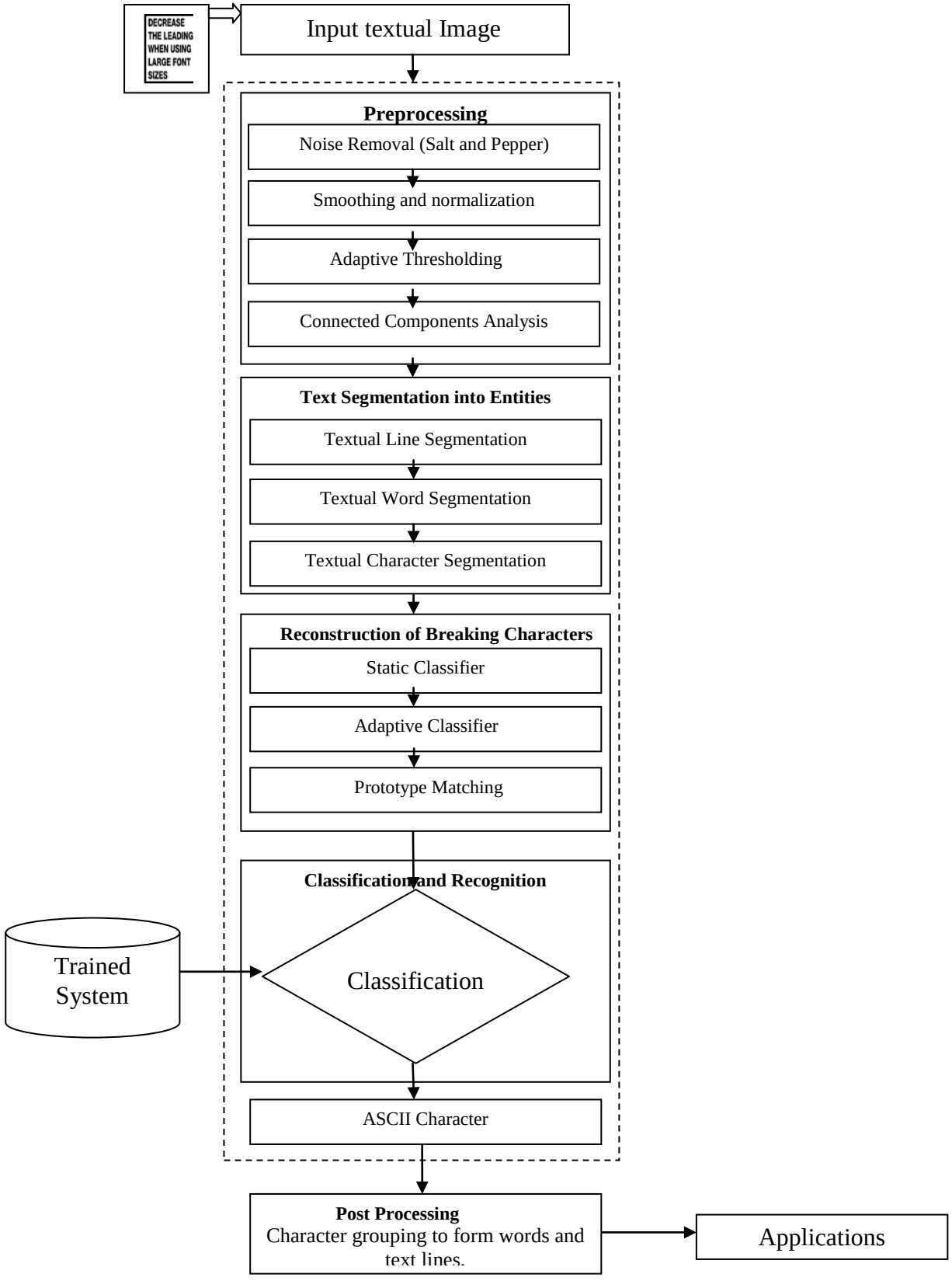


Figure 35: Complete flow of text recognition system.

5.2.2 Noise Removal

As the captured image is taken through mobile camera, there is a probability of introduction of noise especially salt and pepper noise in the captured image. Depending on the resolution on the scanner/mobile camera and the success of the applied technique for thresholding, the characters may be smeared or broken. Some of these defects, which may later cause poor recognition rates, can be eliminated by using a preprocessor to smooth the digitized characters as well as make the image noise free. Image smoothing implies both filling and thinning. Filling is done on to the input image to eliminate small breaks, gaps and holes in the digitized characters, while thinning is used to reduce the width of the line. To remove the noise we used Median filter. In addition to smoothing, normalization is also applied to obtain characters of uniform size, slant and rotation.

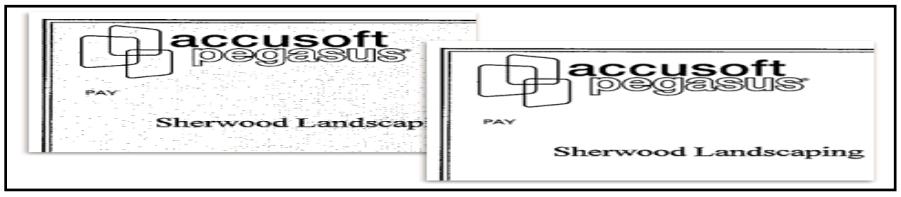


Figure 36: Making an image noise free containing salt and pepper noise

5.2.3 Adaptive Thresholding

Printed documents usually consist of black print on a white background. Hence, when performing OCR, it is common practice to convert the multilevel image into a bi-level image of black and white. The thresholding process is important as the results of the following recognition are totally dependent of the quality of the bi-level image. Here we decided to perform an adaptive thresholding. Adaptive thresholding is necessary as the image may be affected from uneven lightning conditions as shown in figure. In this case

the different regions of the image may exhibit different shadings, for which a global threshold may not feasible to binarize the image. Hence there is a need of different thresholds for different regions of image.

Figure 37: Some instance of original documents vs. Adaptive Threshold document

5.3 Text Segmentation into Entities

Segmentation is a process that determines the constituents of an image. It is necessary to locate the regions of the document where data have been printed and distinguish them from figures and graphics. For instance, when performing automatic mail-sorting, the address must be located and separated from other print on the envelope like stamps and company logos, prior to recognition. Applied to text, segmentation this is the separation of textual contents into textual lines, textual words and textual characters. The isolation of characters or words from textual regions. The majority of optical character recognition algorithms segment the words into isolated characters which are recognized individually. Usually this segmentation is performed by isolating each connected component that is each connected black area. This technique is easy to implement.

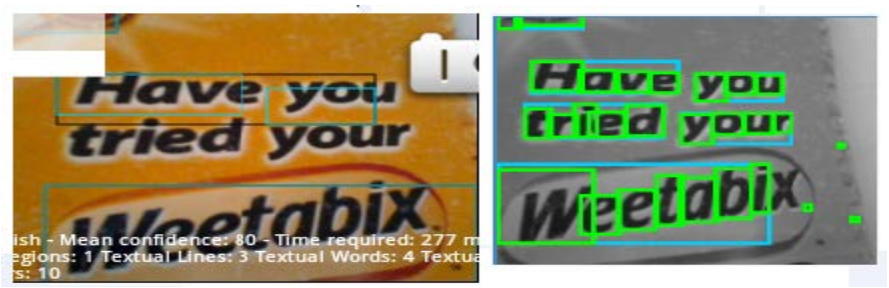


Figure 38: Text line segmentation of a document image

5.3.1 Text line segmentation

For text line segmentation we have used horizontal projection profile and some base line estimations. As the text is printed on white background mostly with no skew, so this methodology works fine for textual line detection.

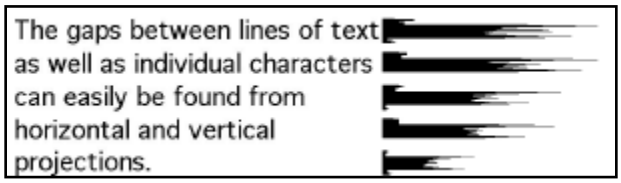


Figure 39: Horizontal projection profiles for text line segmentation

5.3.2 Text word segmentation

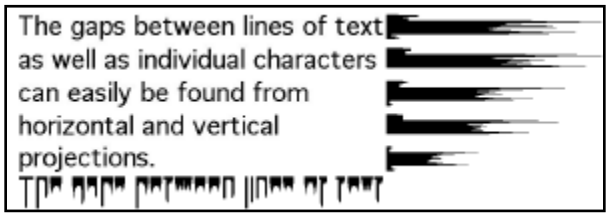
After the textual lines segmentations have been performed, a technique based on inter-word distances is used to segment the individual textual lines into textual words. For a printed document the inter-word distances are mostly same. A threshold in this case t=0.5 is maintained for inter-word distances. Two textual occurrences having a distance measure threshold greater then t are categorized as two different words.

5.3.3 Text character segmentation

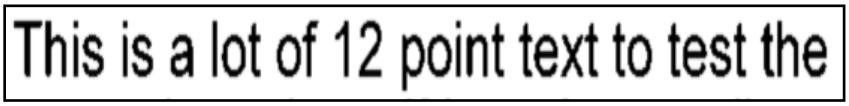
After successful segmentation of textual lines into words, next task is to segment the individual words into characters so that character by character input is sent to

classification module to output the corresponding character into ASCII. We have segmented the individual words into characters using vertical projection profile as in isolated printed documents the textual characters of words contains a minute white area as shown in figure ,which can be detected by vertical projection profiles.

(a)



(b)



(c)

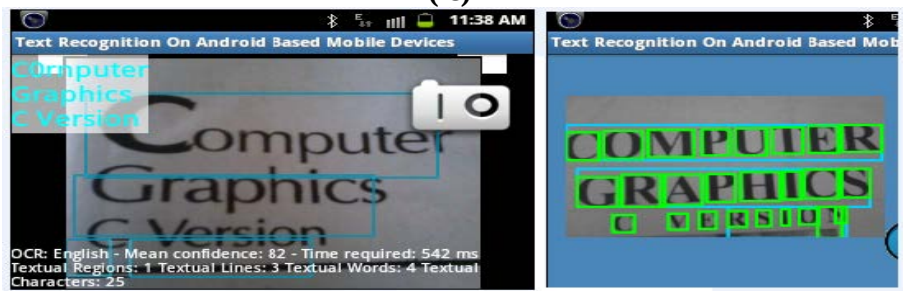


Figure 40: (a): Vertical projections for character segmentation. (b): Inter-character distances.(c):Text character segmentation with original image.

5.4 Reconstruction of Breaking Characters

At this point we have generally well segmented individual characters available. But during preprocessing phase especially during thresholding there is a possibility that individual characters break down into parts as shown in figure 41(a). This has a very adverse effect in the classification phase as the broken characters could not be classified correctly and hence a false ASCII output is obtained. To overcome this problem we have

used feature based approach based on static and dynamic classifiers. Static classifiers basically uses an outline approximation of the character prototype and hence match the outline features with the prototype (features of character stored during training phase) and approximate the input characters. Dynamic classifier has also the same procedure accompanied by normalization of the input character.

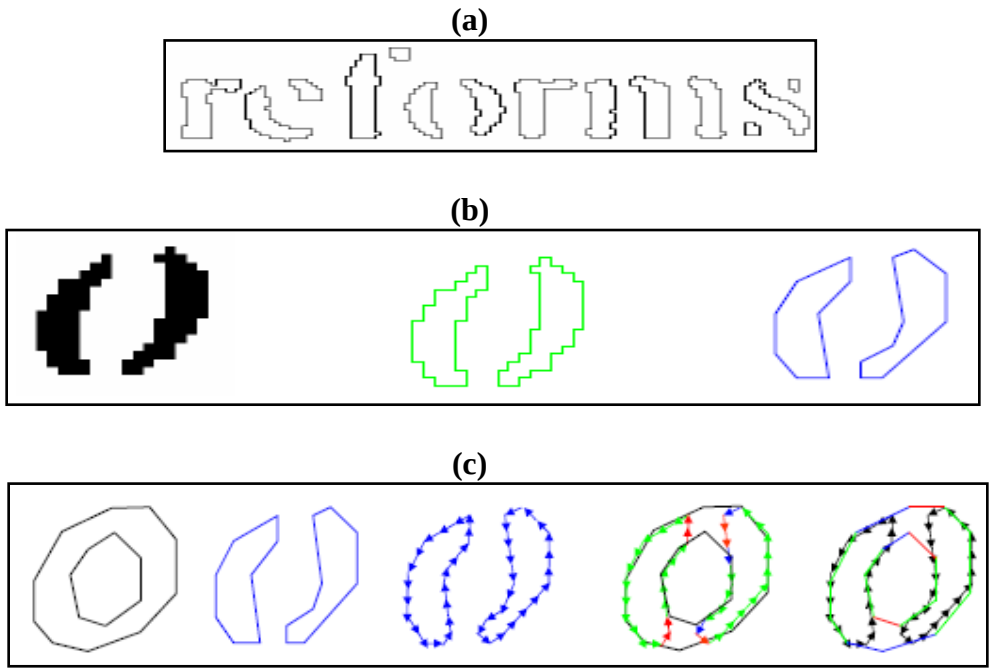


Figure 41: (a): A broken character word. (b): Polygonized approximation of the broken word. (c): outline features matching with character prototype.

5.5 Training the Text Recognition System

5.5.1 An Overview

After successfully segmenting the textual content from the image into its basic entities i.e. characters, the next stage is to classify the input character and subsequently output correct ASCII character. This stage of our text recognition system is the most critical and challenging stage as for as the accuracy is concerned. The classification stage includes a classifier to correctly classify the input based on training the intended output. Hence the

role of classifier is extremely focal in any recognition paradigm. There are various classifiers existing like bi-level or multi-level classifiers. For our text recognition system we have used Multi-Layer neural networks to train the system. An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of the ANN paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in union to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well. The most common neural network model is known as a supervised network because it requires a desired output in order to learn. The goal of this network type is to create a model that maps the input to the output using historical data so that the model can then be used to produce the output when the desired output is unknown. A graphical representation of a Multi-Layer Perceptron (MLP) is shown in Figure 42

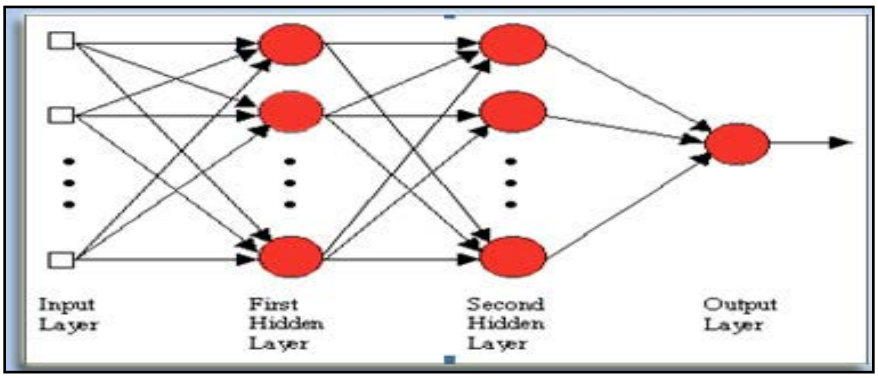


Figure 42: A graphical representation of Multi layer neural network.

Figure 43 demonstrates a neural network used within an optical character recognition (OCR) application the original document is scanned into the computer and saved as an image. The OCR software breaks the image into sub-images, each containing a single character. The sub-images are then translated from an image format into a binary format, where each 0 and 1 represents an individual pixel of the sub-image. The binary data is then fed into a neural network that has been trained to make the association between the character image data and a numeric value that corresponds to the character. The output from the neural network is then translated into ASCII text and saved as a file.

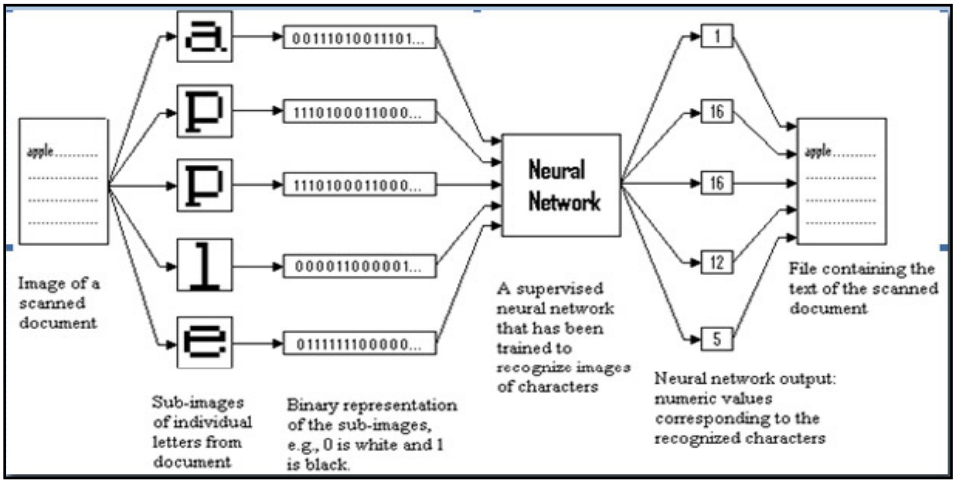


Figure 43: An internal (working) representation of Multi layer neural network.

5.5.2 Our Training Procedure

A back-propagation neural network with one hidden layer was used to create an adaptive character recognition system. The system was trained and evaluated with printed text, as well as printed like several different forms of handwriting provided by both male and female participants.

5.5.2.1 Generation of Training Images

5.5.2.1.1 Training Data specifications

The first step is to determine the full character set to be used, and prepare a text or word processor file containing a set of examples. This text recognition system is basically developed for bigger fonts (16+pts).Most likely suitable for signboard recognition. The training data used have following specifications:

- Times New Roman inspired fonts are used to train
- Bigger fonts(16+) size is used for training data
- Different variations of font style like bold and italic styled characters were also used in training set.
- Printed like handwritten text was also used to train the system

The most important points we kept in mind when creating a training file are as follows:

- We made sure that there are a minimum number of samples of each character. We used 20 such samples for each character.
- There were more samples of the more frequent characters like a.b.c, m, n etc - at least 40.
- We did not make a mistake of grouping all the non-letters together. We made the training text more realistic. For example, **The quick brown fox jumps over the lazy dog. 0123456789!@#$ %^&(),.{}<>/?** is terrible to be produced for training. Much better is **The (quick) brown {fox} jumps! over the $3,456.78** `<lazy>` **#90 dog & duck/goose, as 12.5% of E-mail from aspammer@website.com is spam?**

This gives the text line finding code a much better chance of getting sensible baseline metrics for the special characters.

- We paid special attention that it is ABSOLUTELY **VITAL** to space out the text a bit when printing, so up the inter-character and inter-line spacing in any word processor. Not spacing text out sufficiently causes "FAILURE! box overlaps no blobs or blobs in multiple rows" errors during training file generation usually occurs.
- The training data was grouped by font. Ideally, all samples of a single font were grouped in a single tiff file,
- Some instances of training data used to train the system are shown in figure.

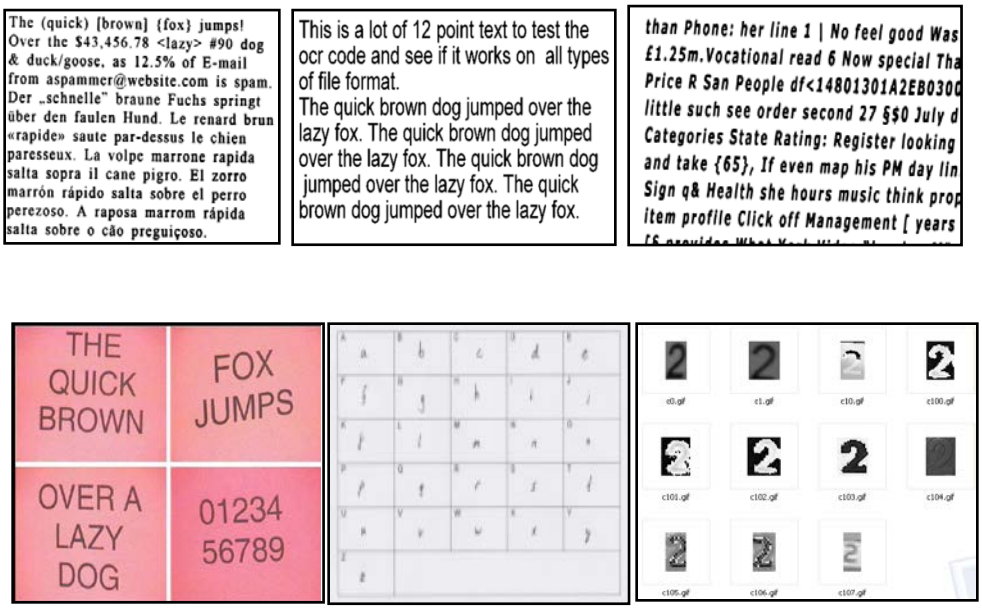


Figure 44: Some instances of images used for training

5.5.2.1.2 Neural Network Design

The neural network had three layers: an input layer consisting of 100 nodes (for the 10 by 10 character input), a hidden layer consisting of 50 nodes, and an output layer with 26 nodes (one for each letter). The network uses back-propagation in addition to bias weights and momentum.

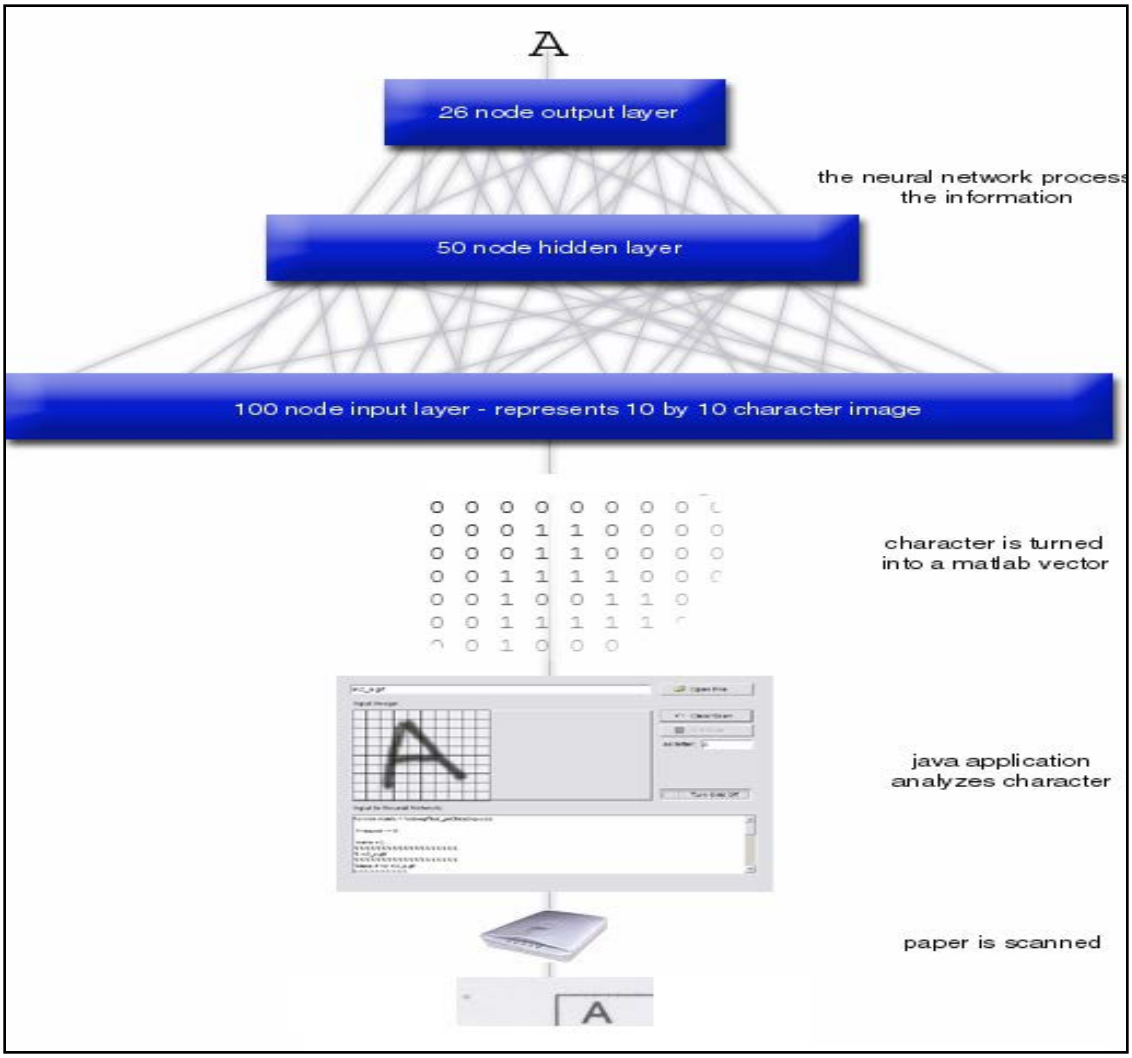


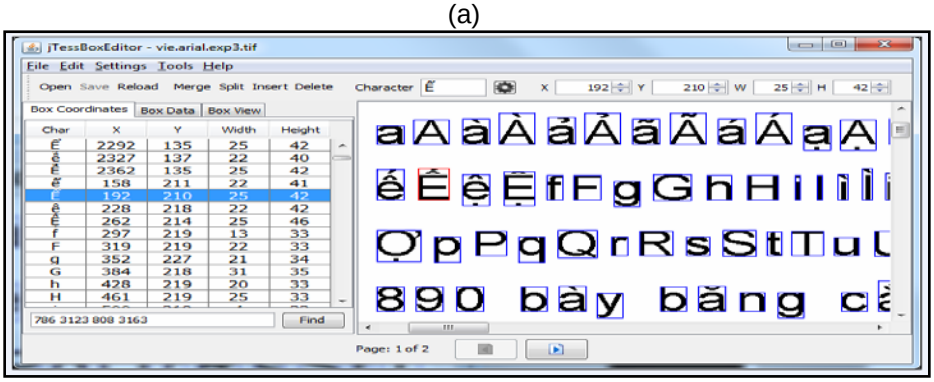
Figure 45: Neural network design and working representation

Next we printed and scanned the training set to create an image (.tiff format) of our training page.

5.5.2.2 Generation of Box Files

The next step is to generate the box files. The training software needs a 'box' file to go with each training image. The boxes were generated around each training character using automatic tool as shown in figure. The box file is a text file that lists the characters in the training image, in order, one per line, with the coordinates of the bounding box around the image. Each boxed and labeled character was then given as an input to the training software for system training. This is a text file. Some samples of a part of box text files containing character labels and dimensions are shown in figure 46.

(a)



(b)

```
s 734 494 751 519
p 753 486 776 518
r 779 494 796 518
i 799 494 810 527
n 814 494 837 518
g 839 485 862 518
t 865 492 878 521
u 101 453 122 484
b 126 453 146 486
e 149 452 168 477
r 172 453 187 476
d 211 451 232 484
e 236 451 255 475
n 259 452 281 475
```

```
D 101 504 131 535
e 135 502 154 528
r 158 503 173 526
, 197 498 206 510
, 206 497 214 509
s 220 501 236 526
c 239 501 258 525
h 262 502 284 534
n 288 501 310 525
e 313 500 332 524
l 336 501 347 534
l 352 500 363 532
e 367 499 386 524
" 389 520 407 532
```

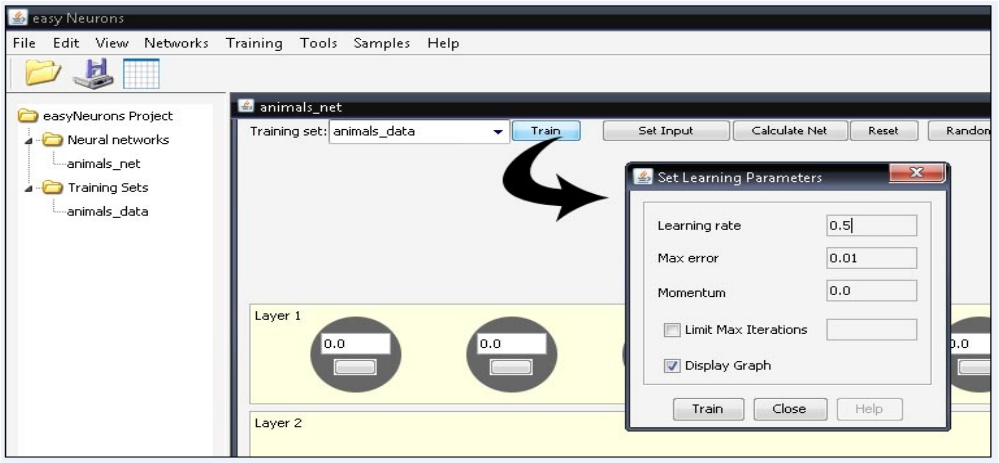
```
D 101 504 131 535
e 135 502 154 528
r 158 503 173 526
" 197 497 214 510
s 220 501 236 526
c 239 501 258 525
h 262 502 284 534
n 288 501 310 525
e 313 500 332 524
l 336 501 347 534
l 352 500 363 532
e 367 499 386 524
" 389 520 407 532
```

Figure 46: Box files (a) and corresponding character label files (b).

5.5.2.3 Training

The box files along with boxed training characters are given as input to the training software in our case it is java based software called “easy neurons”. First the network design parameters are specified like input/output nodes and number of hidden layers which in our case is only one. The training software then make the system to learn based on input character image. The learning process is extremely time consuming. In our case it took five hours to generate an initial first version of trained data file. We had multiple training sessions of approximately same periods to generate a final training file after testing. Some instances of training software while training are shown in figure 47.

(a)



(b)

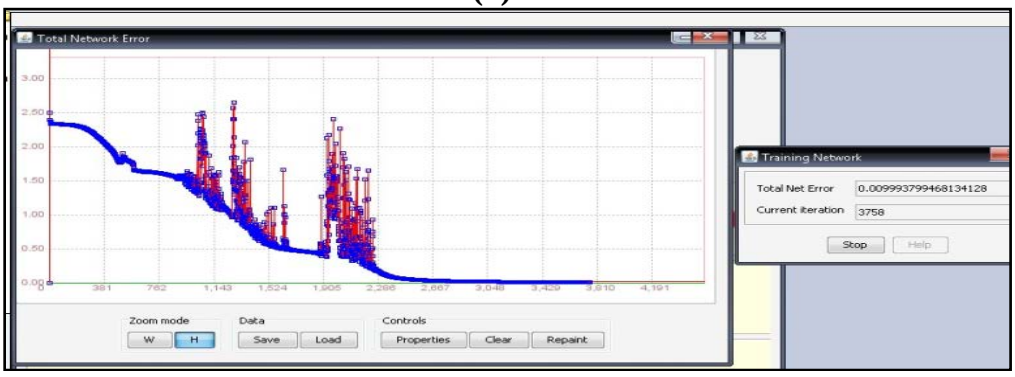


Figure 47: some training instances of text recognition system.

5.6 Recognition/ASCII Output

The trained file produced in previous stage of training was used in our Android project. The complete android based system work as follows.

- Image is captured using mobile's camera
- Preprocessing and character segmentation is done.
- Segmented characters are matched with training file
- Likely character corresponding to the input character is output in an ASCII format.

5.7 Post Processing

Character grouping to form words

As the characters combine to form words .The individual ASCII characters are combined to form words and the combination of words produces textual lines and textual regions.

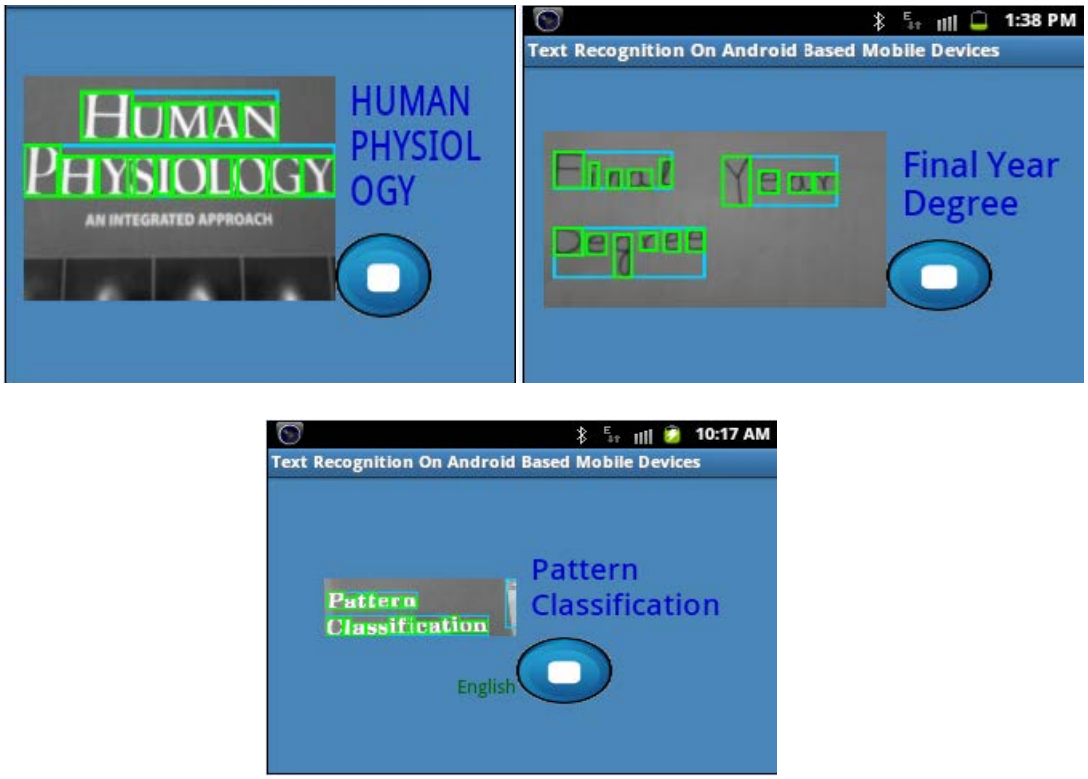


Figure 48: some instances of images showing character groupings to form word.

5.8 Applications

The ASCII text obtained from text recognition module remains useless until and unless we use the recognized text purposefully into some applications we therefore have developed six applications which uses recognize text in six different ways. Each of the application is described as follows.

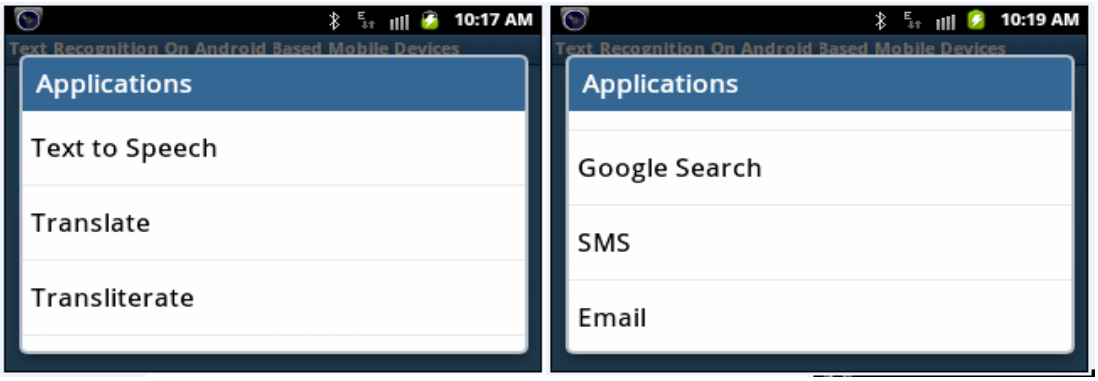


Figure 49: Applications Interface

Text to Speech

Our text to speech application converts the recognized text to speech. This is a useful application for visually impaired people. For text to speech conversion we used “tts” android text to speech service, which is extremely efficient. An illustration of this utility is shown in figure.

Figure 50: Text to Speech Interface

Translate

Automatic translation of recognized text is another feature of our project. Text can be translated into more than 50 languages now. We used Google translation API for translation which is the best and efficient for real time translation. An illustration of this application is shown in figure.

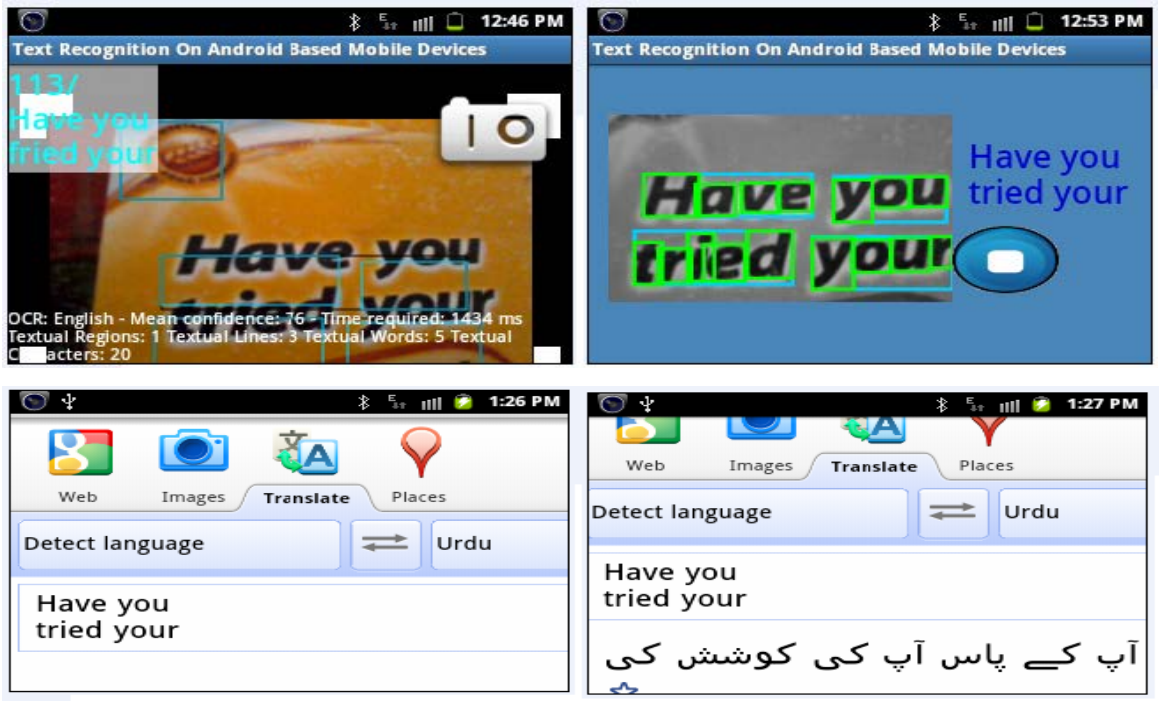


Figure 51: Translation Application Interface

Transliterate

Yet another very novel application provided by our system is to transliterate the recognized text. Transliteration of text is to convert the text syntactically to some other language e.g. to write English in Urdu syntactically. For this utility we used Google transliterate service. An illustration of this utility is shown in figure**.**

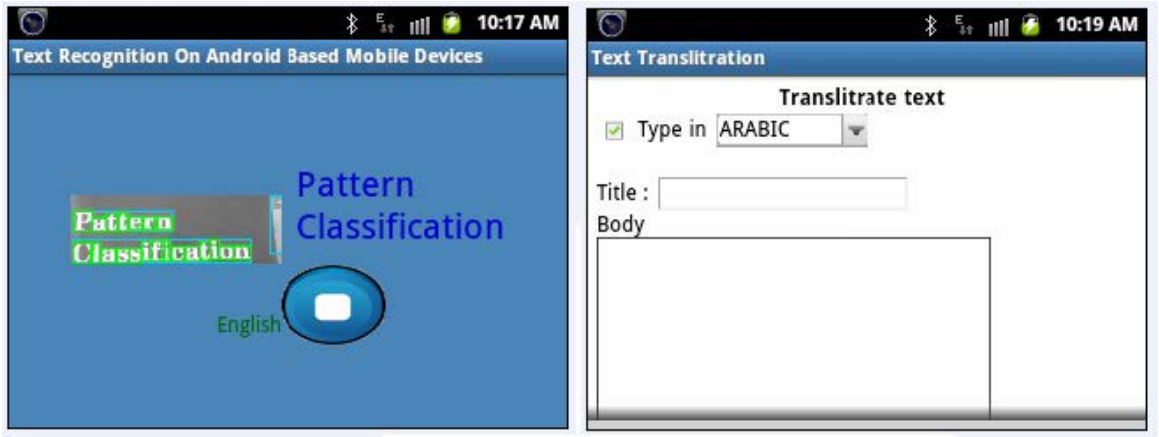


Figure 52: Transliterate Application Interface

SMS

Another service provided by our project is to SMS (share the information) the recognized text. For this we used android SMS service. An illustration of the utility is shown in figure.

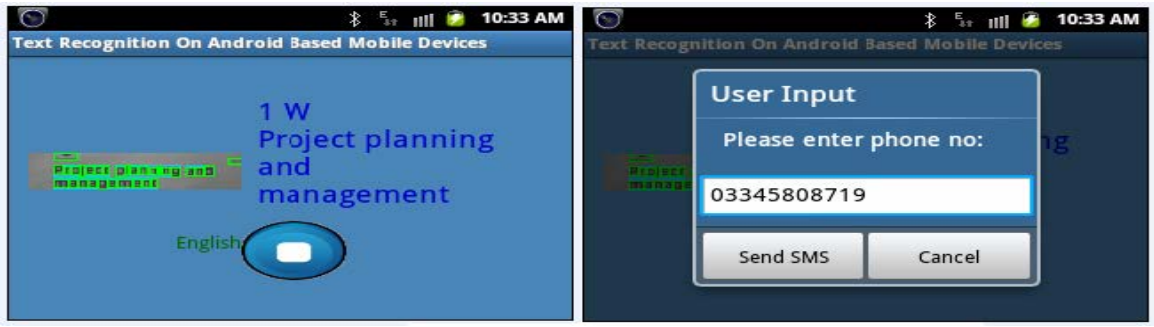


Figure 53: SMS Application Interface

Email

To email the recognized text we used Android's telephony service. This is yet another way of sharing information. An illustration of the service is shown in figure.

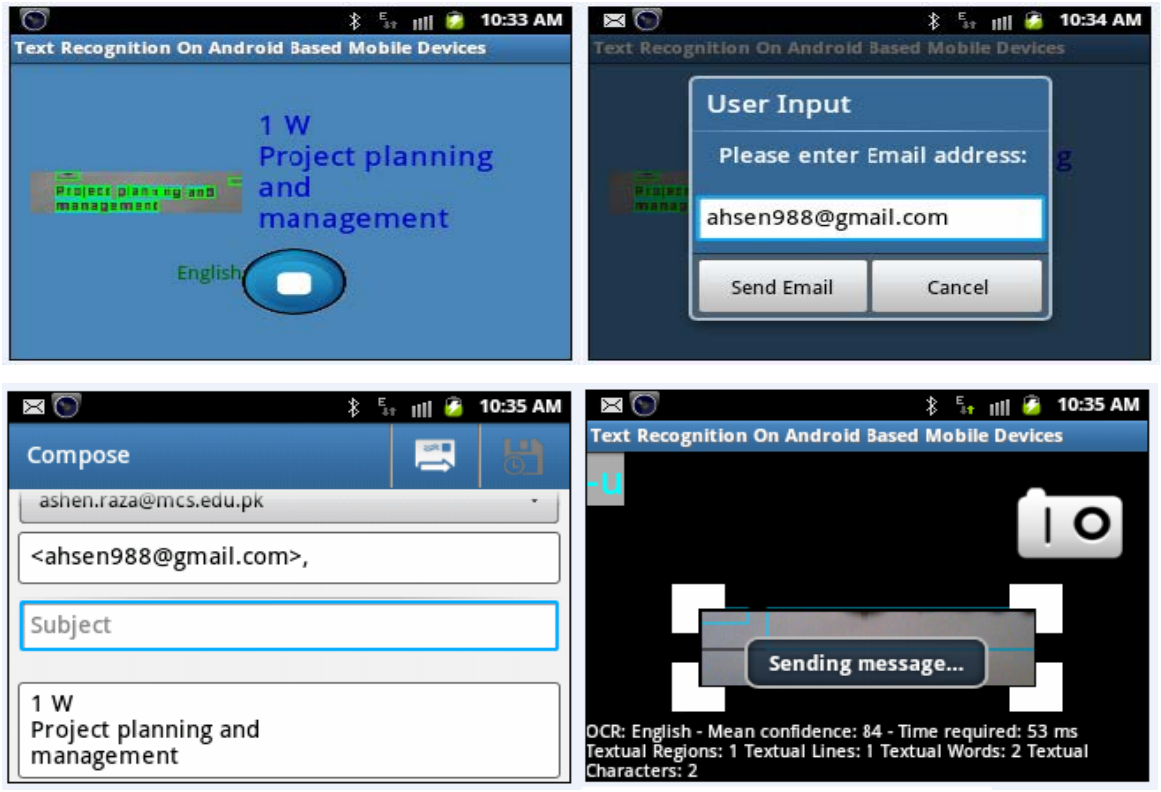


Figure 54: Email Application Interface

Add new contact

Another application of our text recognition project is to get snap shot of just important fields of business cards (name and phone number) and just make a new contact in contact's directory. An illustration of the service is shown in figure.

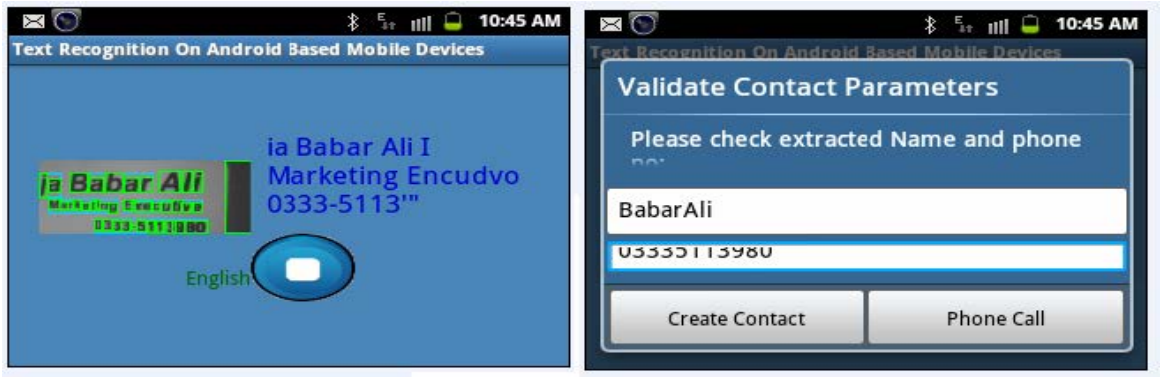


Figure 55: Auto Contact Application Interface

CHAPTER 6

TESTING AND EVALUATION

6. Testing

Testing any software project/product/program is essential to check and ensure the provision of intended functionality and quality of software product. We have tested our software product on two levels:

- Interface Testing
- Functional Testing
  - Text Recognition Testing
  - Application Testing

6.1 Interface Testing

Test Case 1

Test Case ID	01
Test Case name	Text Recognition Application button testing
Input(s)	Press Text Recognition Application button
Output	Opens the Text Recognition application
Sequence of Action(s)	Press Text Recognition Application button from menu of Android Mobile
Result	Success

Table 3: Test Case 1

Execution of test case

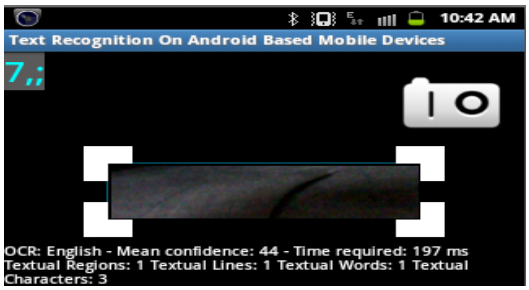


Figure 56: Test case # 01 Execution

Test Case 2

Test Case ID	02
Test Case name	Continuous Preview testing
Input(s)	An Image containing textual occurrences
Output	Continuously showing ASCII text
Sequence of Action(s)	Focus on textual regions of image using rectangular box
Result	Success

Table 4: Test Case 2

Execution of Test Case

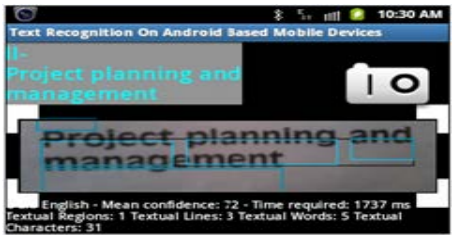


Figure 57: Test case # 02 Execution

Test case 3

Test Case ID	03
Test Case name	Rectangular box testing
Input(s)	An Image containing textual occurrences
Output	Displaying textual regions of image in continuous preview
Sequence of Action(s)	Focus on textual regions in the image using rectangular box
Result	Success

Table 5: Test Case 3

Execution of test case

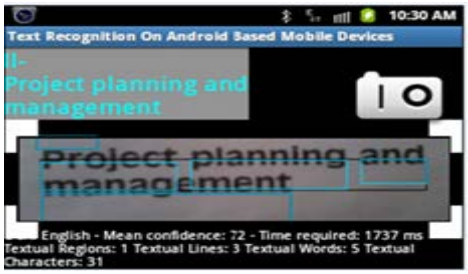


Figure 58: Test case # 03 Execution

Test Case 4

Test Case ID	04
Test Case name	Capture Image button testing
Input(s)	An Image containing textual occurrences, continuous preview displaying correct results
Output	ASCII text
Sequence of Action(s)	Focus on textual region in the image-> Continuous Preview displaying correct results -> Press Capture Image button
Result	Success

Table 6: Test Case 4

Execution of test case

Figure 59: Test case # 04 Execution

Test Case 5

Test Case ID	05
Test Case name	Bounding boxes testing
Input(s)	Correctly recognized ASCII text
Output	Bounding boxes on each character of recognized text
Sequence of Action(s)	Focus on textual region in the image-> Continuous Preview displaying correct results -> Press Capture Image button -> ASCII text -> Bounding boxes on each character of ASCII text
Result	Success

Table 7: Test Case 5

Execution of test case

Figure 60: Test case # 05 Execution

Test Case 6

Test Case ID	06
Test Case name	Applications button testing
Input(s)	Press Applications button
Output	List of applications
Sequence of Action(s)	ASCII text -> Press Applications button -> Opens list of applications
Result	Success

Table 8: Test Case 6

Execution of test case

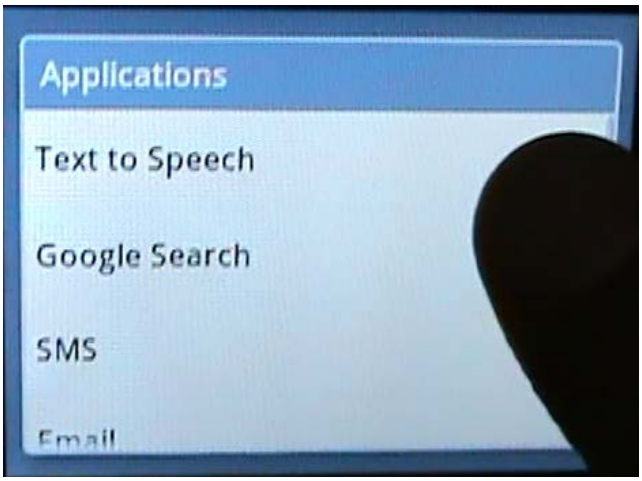


Figure 61: Test case # 06 Execution

Test Case 7

Test Case ID	07
Test Case name	Text to Speech Option testing
Input(s)	ASCII text, Select Text to Speech option
Output	Spoken ASCII text
Sequence of Action(s)	ASCII text -> Press Applications button -> Opens list of applications -> Select Text to Speech option from Applications list -> Speak ASCII text -> Displays toast message
Result	Success

Table 9: Test Case 7

Execution of test case

Figure 62: Test case # 07 Execution

Test Case 8

Test Case ID	08
Test Case name	Google Search Option testing
Input(s)	ASCII text, Select Google Search option, Internet Connection available
Output	Google results regarding ASCII text
Sequence of Action(s)	ASCII text -> Press Applications button -> Opens list of applications -> Select Google Search option from Applications list -> Pastes ASCII text on Google bar -> Displays Google results regarding ASCII text
Result	Success

Table 10: Test Case 8

Execution of test case

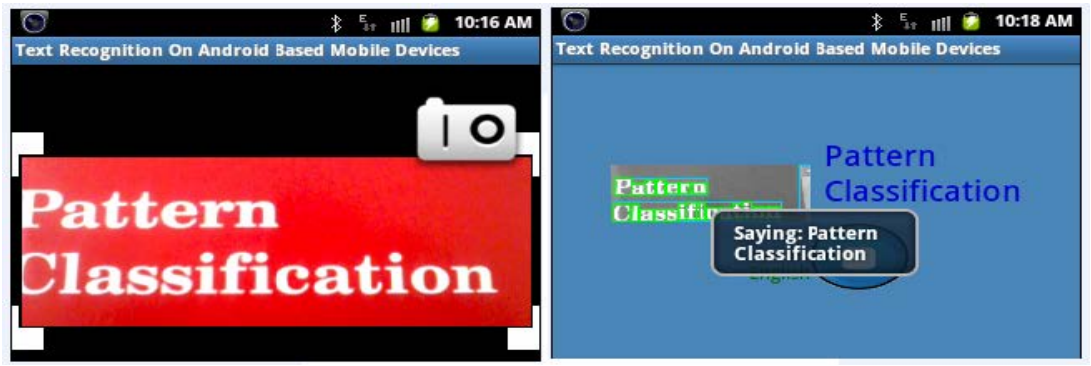


Figure 63: Test case # 08 Execution

Test Case 9

Test Case ID	09
Test Case name	SMS Option testing
Input(s)	ASCII text, Select SMS option
Output	SMS ASCII text
Sequence of Action(s)	ASCII text -> Press Applications button -> Opens list of applications -> Select SMS option from Applications list -> Enter phone number of recipient -> Send SMS
Result	Success

Table 11: Test Case 9

Execution of test case

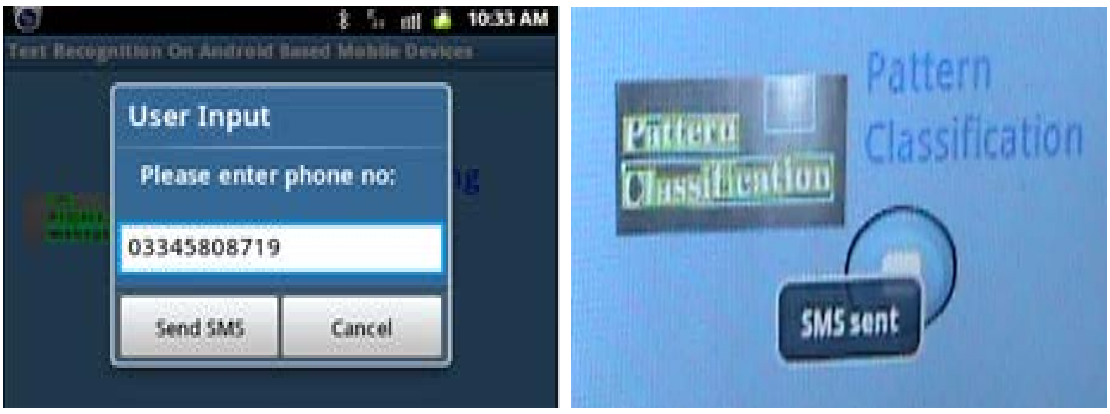


Figure 64: Test case # 09 Execution

Test Case 10

Test Case ID	10
Test Case name	Email Option testing
Input(s)	ASCII text, Internet Connection available, Select Email option
Output	Email ASCII text
Sequence of Action(s)	ASCII text -> Press Applications button -> Opens list of applications -> Select Email option from Applications list -> Enter email id of recipient -> Send Email
Result	Success

Table 12: Test Case 10

Execution of test case

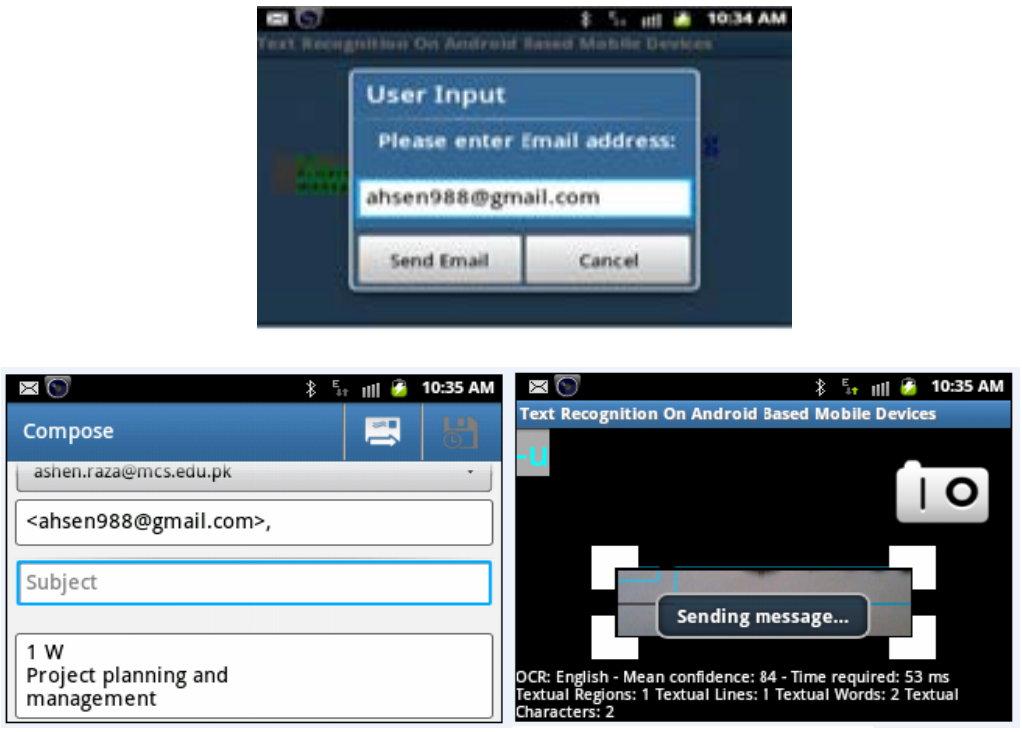


Figure 65: Test case # 10 Execution

Test case 11

Test Case ID	011
Test Case name	Email Option testing
Input(s)	ASCII text, Select Auto Contact option
Output	Add new contact to Contact's directory
Sequence of Action(s)	ASCII text -> Press Applications button -> Opens list of applications -> Select Auto Contact option from Applications list -> Add new contact in contact list
Result	Success

Table 13: Test Case 11

Execution of test case

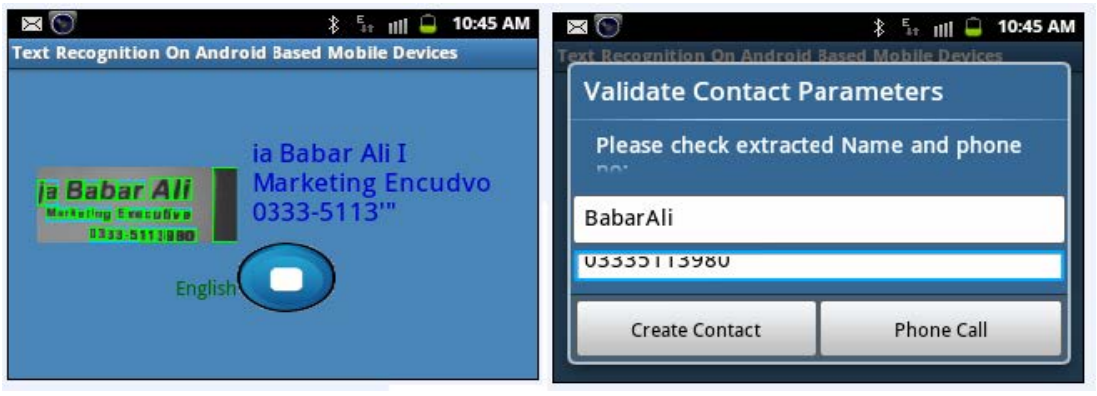


Figure 66: Test case # 11 Execution

Test Case 12

Test Case ID	012
Test Case name	Translate Option testing
Input(s)	ASCII text, Internet Connection available, Select Translate option
Output	Translated ASCII text in to selected language
Sequence of Action(s)	ASCII text -> Press Applications button -> Opens list of applications -> Select Translate option from Applications list
Result	Success

Table 14: Test Case 12

Execution of test case

Figure 67: Test case # 12 Execution

Test Case 13

Test Case ID	013
Test Case name	Transliterate Option testing
Input(s)	ASCII text, Internet Connection available, Select Transliterate option
Output	Transliterated ASCII text in to selected language
Sequence of Action(s)	ASCII text -> Press Applications button -> Opens list of applications -> Select Transliterate option from Applications list
Result	Success

Table 15: Test Case 13

Execution of test case

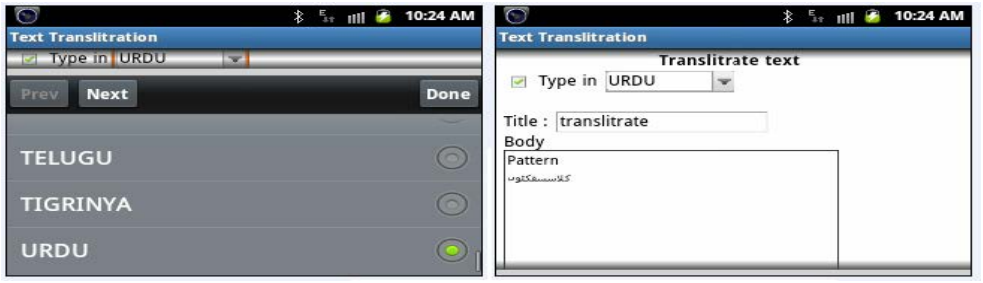


Figure 68: Test case # 13 Execution

6.2 Functional Testing

We first present the representative test cases of text recognition testing.

6.2.1 Text Recognition Testing

Test Case: 1

Test Case ID	01
Test Case name	Normal flow testing
Input(s)	An Image containing textual occurrences
Output	ASCII text
Sequence of Action(s)	Capture image->Recognize text->ASCII text
Result	Success

Table 16: Test Case (Recognition) 1

Execution of Test Case

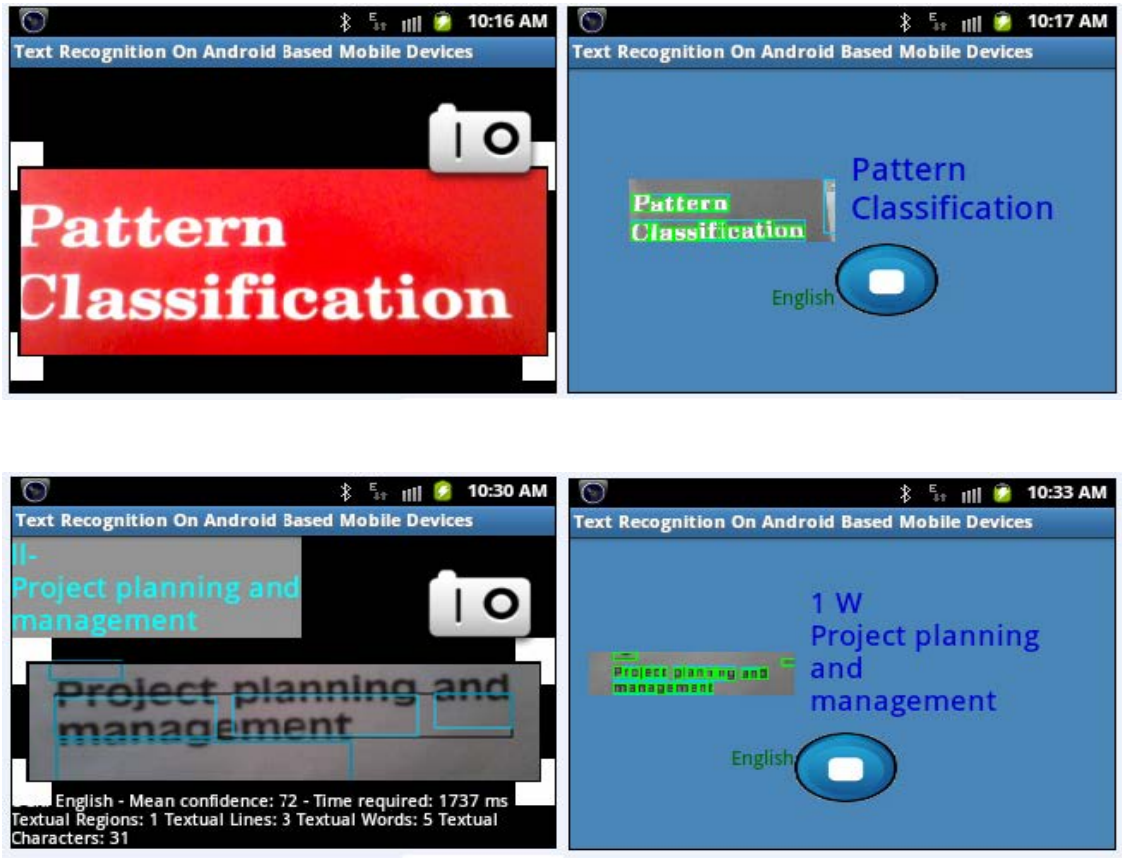


Figure 69: Test case # 1 (Text Recognition) Execution

Test Case: 2

Test Case ID	02
Test Case name	Lightning condition based testing
Input(s)	An Image containing textual occurrences
Output	ASCII text
Sequence of Action(s)	Capture image->Recognize text->ASCII text
Result	Success

Table 17: Test Case (Recognition 2)

Execution of Test Case

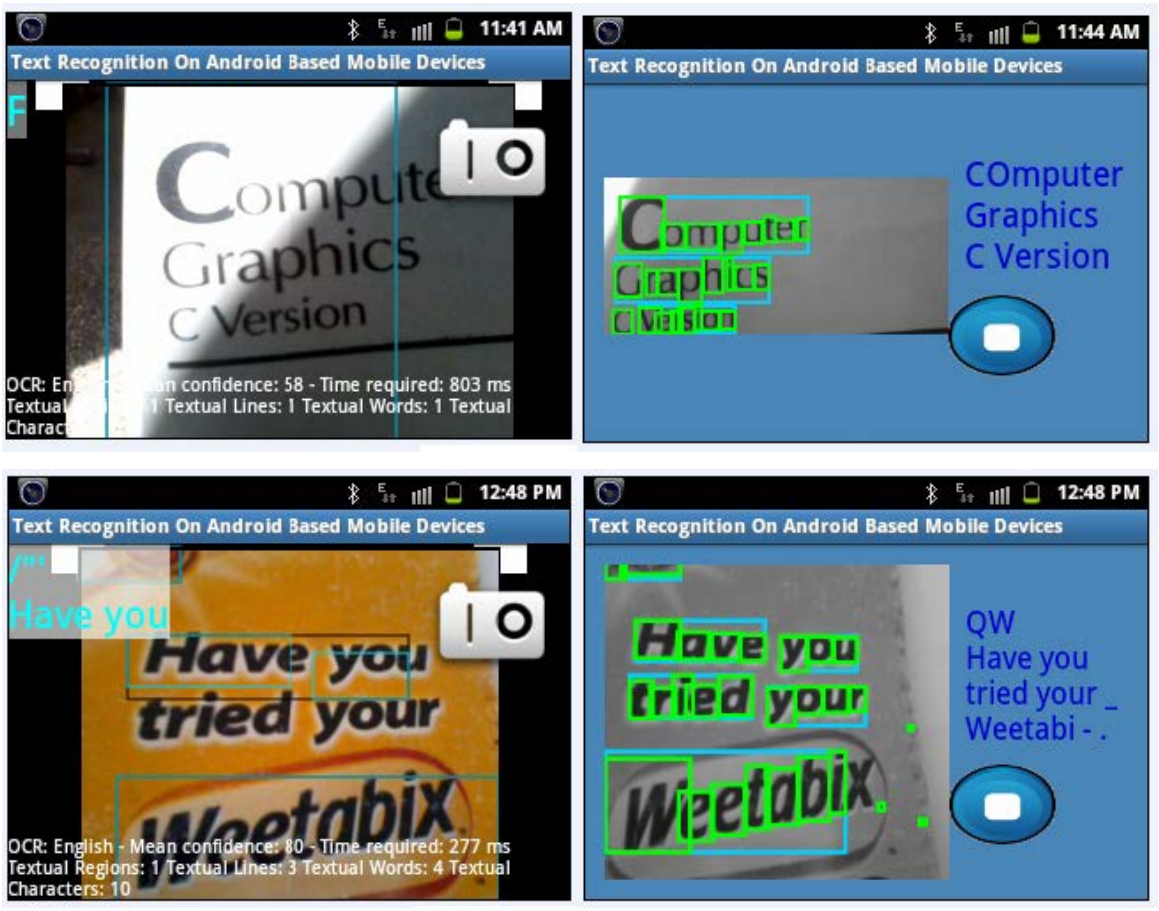


Figure 70: Test case # 2 (Text Recognition) Execution

Test Case: 3

Test Case ID	03
Test Case name	Text background testing
Input(s)	An Image containing textual occurrences
Output	ASCII text
Sequence of Action(s)	Capture image->Recognize text->ASCII text
Result	Success

Table 18: Test Case (Recognition 3)

Execution of Test Case

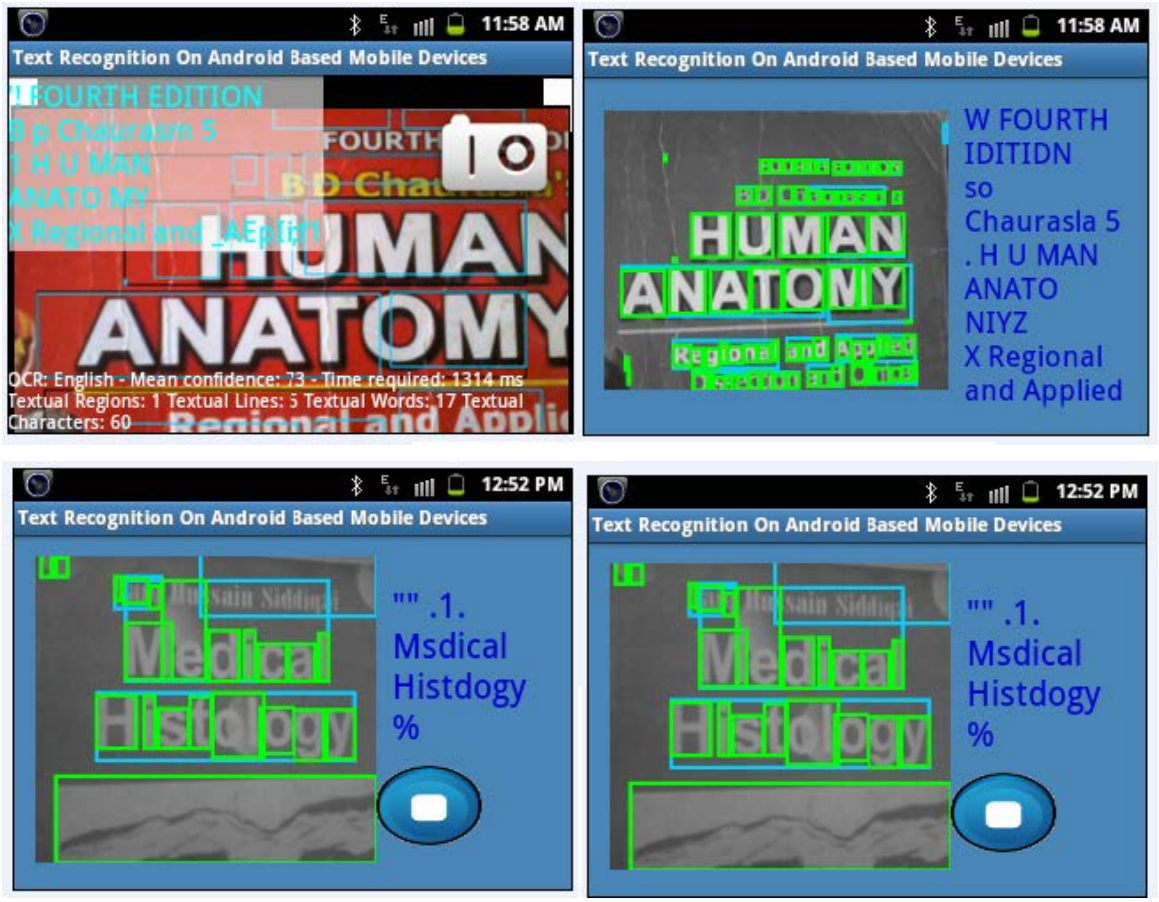


Figure 71: Test case # 3 (Text Recognition) Execution

Test Case: 4

Test Case ID	04
Test Case name	Text Font/size/style testing
Input(s)	An Image containing textual occurrences
Output	ASCII text
Sequence of Action(s)	Capture image->Recognize text->ASCII text
Result	Success

Table 19: Test Case (Recognition 4)

Execution of Test Case

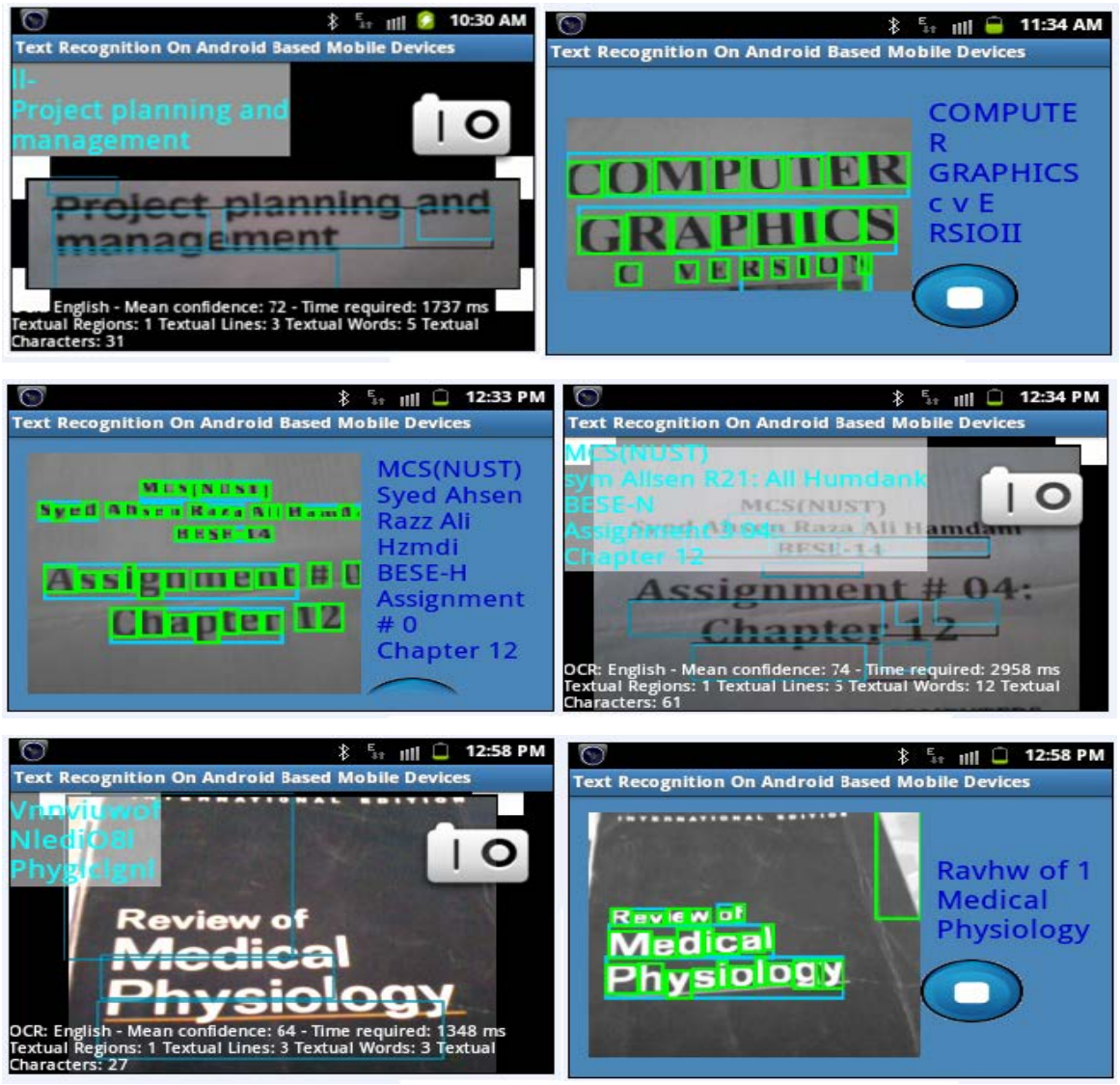


Figure 72: Test case # 4 (Text Recognition) Execution

Test Case: 5

Test Case ID	05
Test Case name	Document skew testing
Input(s)	An Image containing textual occurrences
Output	ASCII text
Sequence of Action(s)	Capture image->Recognize text->ASCII text
Result	Success

Table 20: Test Case (Recognition 5)

Execution of Test Case

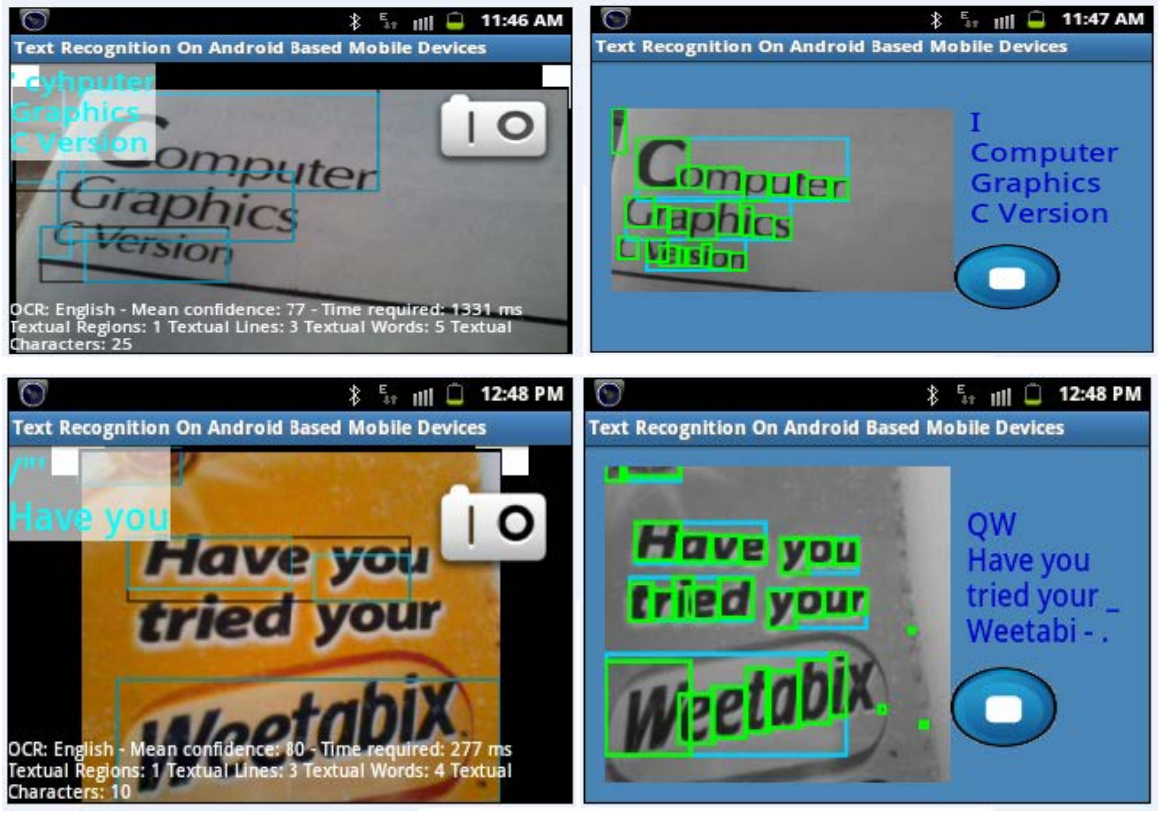


Figure 73: Test case # 5 (Text Recognition) Execution

Test Case: 6

Test Case ID	06
Test Case name	Graphics containing textual image testing
Input(s)	An Image containing textual occurrences with graphics
Output	ASCII text
Sequence of Action(s)	Capture image->Recognize text->ASCII text
Result	Success

Table 21: Test Case (Recognition 6)

Execution of Test Case

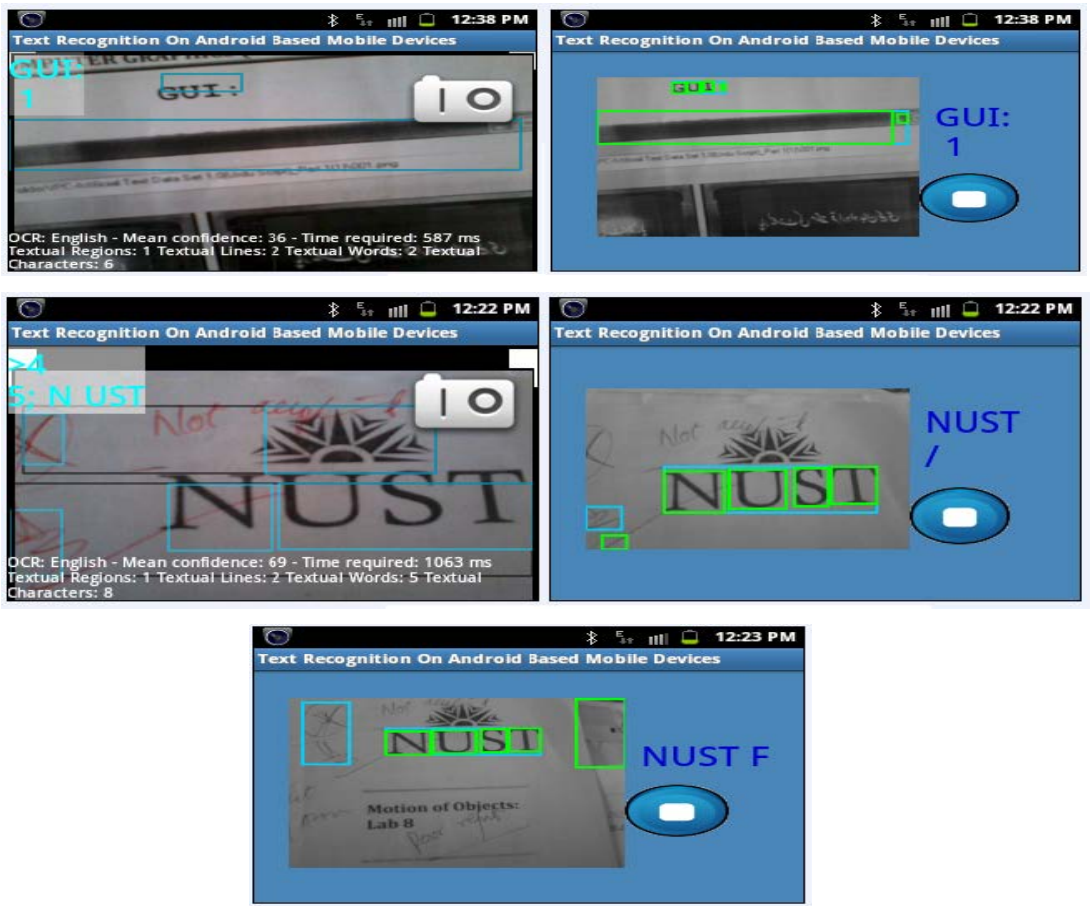


Figure 74: Test case # 6 (Text Recognition) Execution

Test Case: 7

Test Case ID	07
Test Case name	Handwriting Recognition
Input(s)	An Image containing handwritten textual occurrences
Output	ASCII text
Sequence of Action(s)	Capture image->Recognize text->ASCII text
Result	Success

Table 22 : Test Case (Recognition 7)

Execution of Test Case

Figure 75: Test case # 7 (Text Recognition) Execution

6.2.2 Application Testing

Now in the second phase of functional testing we would test each of our application.

Test Case: 8

Test Case ID	08
Test Case name	Text to Speech testing
Input(s)	An Image containing textual occurrences
Output	Spoken ASCII Text
Sequence of Action(s)	Capture image->Recognize text->ASCII text->Speak ASCII text
Result	Success

Table 23: Test Case (Recognition 8)

Execution of Test Case

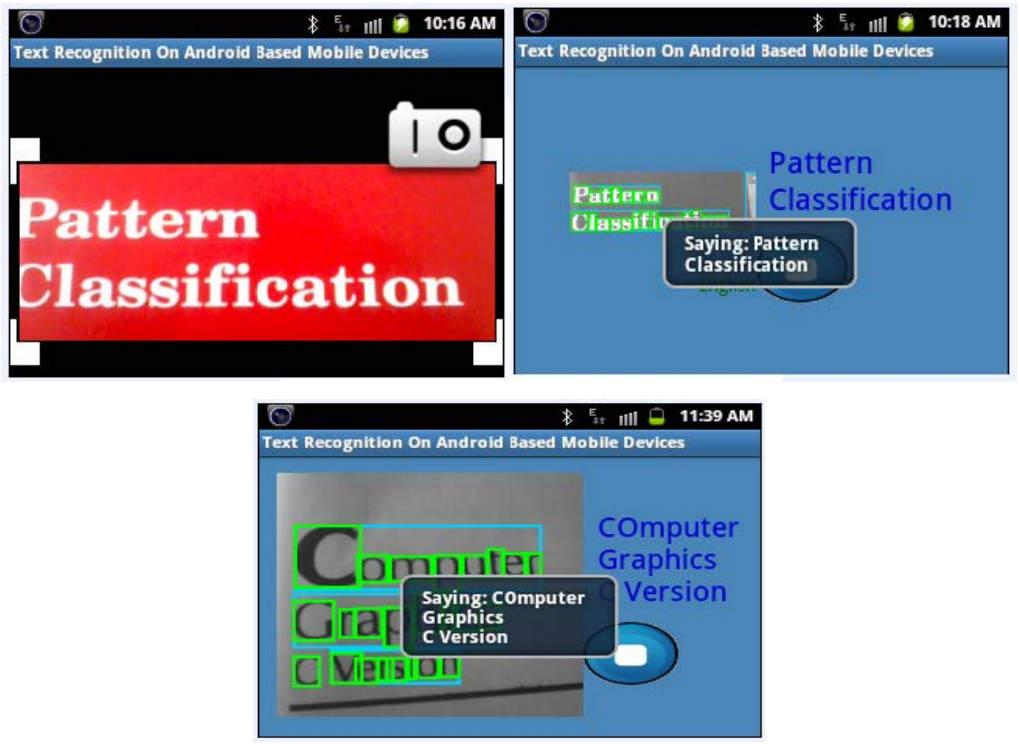


Figure 76: Test case # 8 (Text Application) Executions

Test Case: 9

Test Case ID	09
Test Case name	Translate ASCII text testing
Input(s)	An Image containing textual occurrences
Output	Translated ASCII text into another language
Sequence of Action(s)	Capture image->Recognize text->ASCII text->Translate ASCII text
Result	Success

Table 24: Test Case (Recognition 9)

Execution of Test Case

Figure 77: Test case # 9 (Text Application) Execution

Test Case: 10

Test Case ID	10
Test Case name	Transliterate ASCII text testing
Input(s)	An Image containing textual occurrences
Output	Transliterated ASCII text into another language
Sequence of Action(s)	Capture image->Recognize text->ASCII text-> Transliterated ASCII text
Result	Success

Table 25: Test Case (Recognition 10)

Execution of Test Case

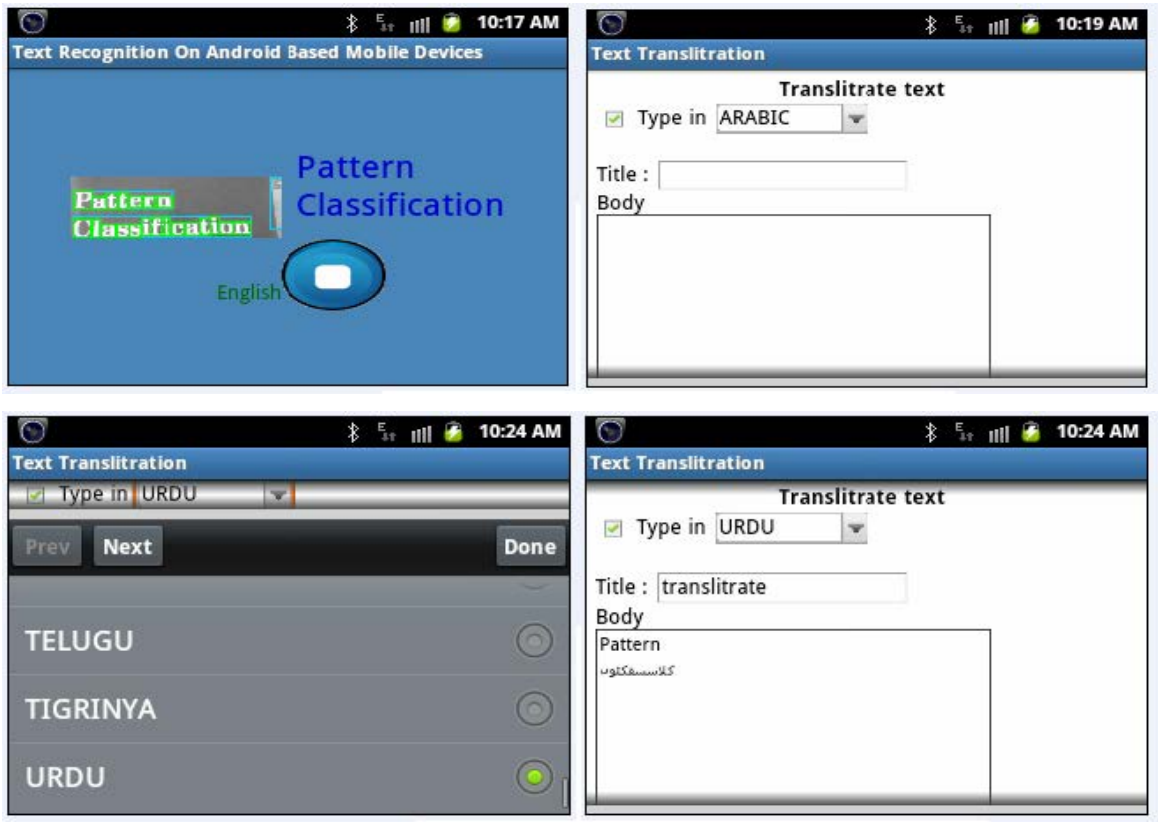


Figure 78: Test case # 10 (Text Application) Execution

Test Case: 11

Test Case ID	11
Test Case name	SMS ASCII text testing
Input(s)	An Image containing textual occurrences
Output	SMS ASCII text
Sequence of Action(s)	Capture image->Recognize text->ASCII text-> SMS ASCII text
Result	Success

Table 26 : Test Case (Recognition 11)

Execution of Test Case

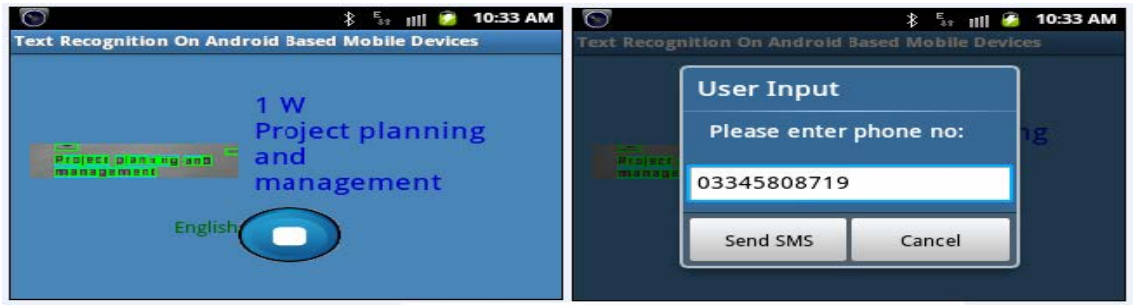


Figure 79: Test case # 11 (Text Application) Execution

Test Case: 12

Test Case ID	012
Test Case name	Email ASCII text testing
Input(s)	An Image containing textual occurrences
Output	Email ASCII text
Sequence of Action(s)	Capture image->Recognize text->ASCII text-> Email ASCII text
Result	Success

Table 27: Test Case (Recognition 12)

Execution of Test Case

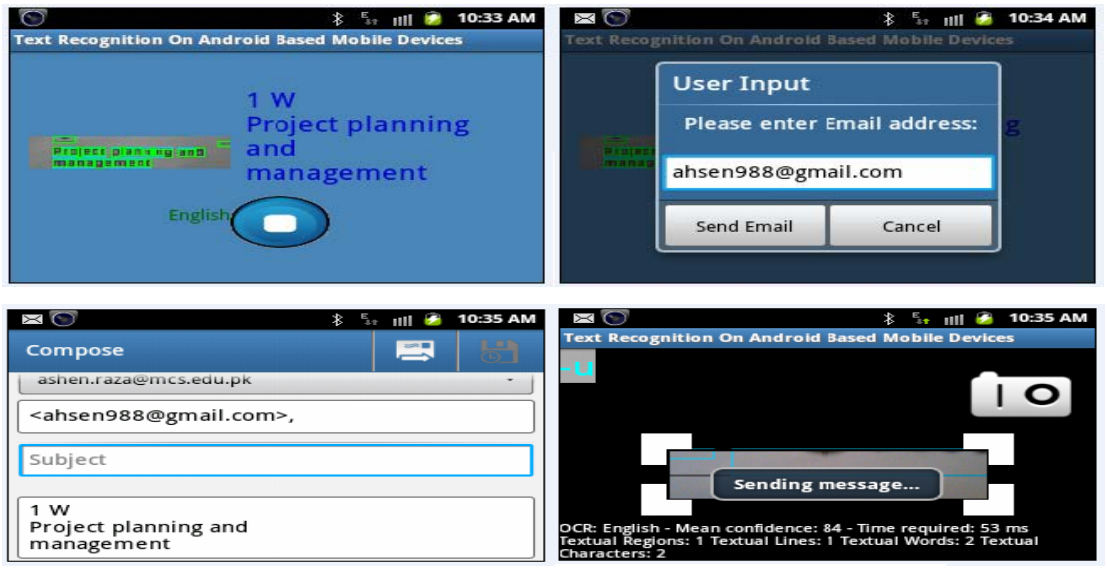


Figure 80: Test case # 12 (Text Application) Execution

Test Case: 13

Test Case ID	013
Test Case name	Add new contact testing
Input(s)	An Image containing textual occurrences
Output	Add new contact to contact's directory
Sequence of Action(s)	Capture image->Recognize text->ASCII text-> Add new contact
Result	Success

Table 28 : Test Case (Recognition 13)

Execution of Test Case

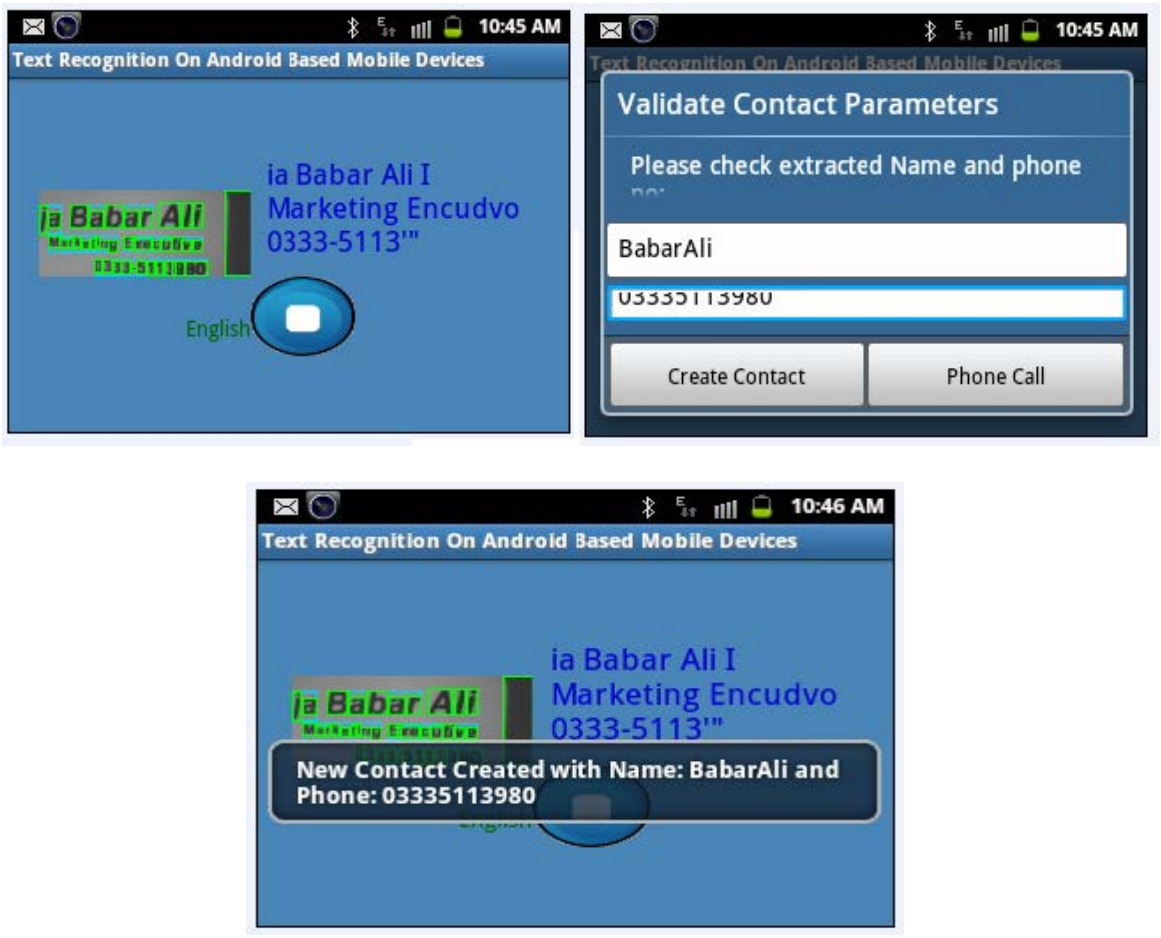


Figure 81: Test case # 13 (Text Application) Execution

6.3 Results

We here present our text recognition results on the basis of different parameters used for testing and evaluation.

Symbol	Recognized%	Unrecognized%
1	96.7	3.3
2	93.3	0
3	100	0
4	86.7	6.6
5	83.3	16.7
6	60	30
7	100	0
8	56.7	56
9	86.7	3.3
A	98.3	1.7
B	96.7	3.3
C	100	0
D	100	0
E	98.3	1.7
F	100	0
G	96.6	3.4
H	100 0	0
I	96.7	3.3
J	78.9	18. 4
K	100	0
L	100	0
M	94.7	5.2
N	100	0
O	98.7	1.3
P	89.4	7.9
Q	100	0
R	100 0	0
S	81.6	18.4
T	100	0
U	89.7	5.9
V	96.6	3.4
W	100	0 0
X	86.8	7.8
Y	93.1	6.9
Z	100	0

Table 29: Rate of recognition of individual symbols

WORD	Recognized %	Problem
THE	100	-
QUICK	93.3	EITHER U OR C NOT RECOGNIZED
BROWN	96.7	B NOT RECOGNIZED
FOX	86.8	X RECOGNIZED I OR N OR NOT RECOGNIZED
JUMPS	57.9	EITHER J OR P NOT RECOGNIZED
OVER	90	E RECOGNIZED AS F OR R RECOGNIZED AS 4
A	100	-
LAZY	86.7	Y RECOGNIZED AS 6
DOG	93.3	G NOT RECOGNIZED
PROJECT	60	T RECOGNIZED AS 4,R RECOGNIZED AS P WRONG WORD SEPERATION PR OBJECT OR PROJEC T
URBAN	70	B RECOGNIZED AS R OR NOT RECOGNIZED ;N RECOGNIZED AS H
SERVICE	38.7	V RECOGNIZED AS 6 ,7 OR YOR NOT RECOGNIZED ,C NOT RECOGNIZED
EXIT	100	-
TU	100	-
ES	86.7	S NOT RECOGNIZED
UN	96.7	U NOT RECOGNIZED
ROBOT	73.3	B RECOGNIZED AS R OR 8 OR NOT RECOGNIZED

Table 30: Rate of recognition of some tested words

CHAPTER 7

CONCLUSION AND FUTURE WORK

7. Conclusion & Future Work

Mobile phones have evolved from passive one-to-one communication device to powerful handheld computing device. Today most new mobile phones are capable of capturing images, recording video, and browsing internet and do much more. Exciting new social applications are emerging on mobile landscape, like, business card readers, sign detectors and translators. These applications help people quickly gather the information in digital format and interpret them without the need of carrying laptops or tablet PCs.

Today most of the mobile phones are equipped with good resolution digital cameras. This enables the mobile phones to run image based application that in past require external digital cameras. However there are still some limitations to what we can do on mobile platform. In context of running an OCR application these limitation pose interesting challenges for research and development. Mobile phones have limited power to run complex software, like OCR engines, using their own limited hardware and memory resources. Huge amount of pixels are to be processed in limited amount of on-board memory compared to desktop systems with GB of memory and additional virtual memory mechanism. Not to mention the more high resolution the images are the more computationally taxing task, it is to run the OCR on mobile phones. Another shortcoming is the absence of the hardware support for real number processing on the mobile platform. Real number operations are simulated using float emulation libraries which affects the efficiency of the software itself. In addition small displays panels make it a difficult task to design a good user interface.

Through this project titled as “Text Recognition on Android based Mobile Devices” we find the solution to printed text recognition problem on android based mobiles. Text

recognition procedure includes a combination of image processing, feature extraction and machine learning based approaches.

Initially some preprocessing operations are performed on the input image, followed by text segmentation into entities like textual lines, words and characters. Broken characters after this phase are reconstructed using static and dynamic classifiers. Well defined individual characters are then sent for classification to our classification module, which classify and output the input character in ASCII based on its matching to training set data. All characters converted to ASCII are then grouped together to form complete words and textual lines in post processing phase.

The system is trained with 20 training samples for each character at minimum. Multiple training sessions after iterative testing and evaluation process generated the final training data file. Our project purposefully uses the recognized text into six applications named as Text to speech, Translation of text, Transliteration of text, SMS, Email and automatically adding new contact from business cards. This training file was then used to classify the input character in our project.

We have developed a complete printed text recognition system, with text to speech and many other feature rich applications using existing technology. The motivation here was to provide a general purpose framework suitable enough to develop more exciting application over it. Moreover since the system is developed in a modular fashion it is very easy to make updates on core components. We can update any single module without affecting other modules in the core system. We can for example replace the existing OCR module with a better and more robust solution. Future work includes, refining the results by OCR using cross-referencing with the dictionary and other high

level semantic techniques, developing more interesting and useful applications using the core framework and making modifications to existing OCR solution for more robust results.

REFERENCES

[1] J. Liang, D. Doermann, and L. Huiping. Camera-based analysis of text and documents Asurvey. International Journal on Document Analysis and Recognition (IJDAR), 7:84–104, 2005.

[2] P. Clark and M. Mirmehdi, “Recognising text in real scenes,” International Journal on Document Analysis and Recognition, vol. 4, no. 4, pp. 243-257, 2002.

[3] R. Lienhart and A. Wernicke, “Localizing and Segmenting Text in Images and Videos,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, pp. 256-268, April, 2002.

[4] T. de Campos, B. Babu, and M. Varma, “Character recognition in natural images,” in VISAPP 2009, Lisboa, Portugal, 2009, pp. 92-103.

[5] D. Doermann, J. Liang, and H. Li, “Progress in camera-based document image analysis,” in ICDAR ’03 Proc. 7th Int. Conf. Document Analysis and Recognition, Edinburgh, UK, 2003, pp. 606-616.

[6] J. Liang, D. Doermann, and H. Li, “Camera-based analysis of text and documents: a survey,” International Journal on Document Analysis and Recognition, vol. 7, no. 2, pp. 84-104, July, 2005.

[7] M. Koga et al., “Camera-based kanji OCR for mobile-phones: practical issues,” in Proc. 8th Int. Conf. Document Analysis and Recognition, Seoul, Korea, 2005, pp. 635-639.

[8] K. S. Bae, K. K. Kim, Y.G. Chung, and W. P. Yu, “Character recognition system for cellular phone with camera,” in COMPSAC ’05 29th Annu. Int. Computer Software and Applications Conf., Edinburgh, UK, 2005, pp. 539-544.

[9] M. Laine and O. Nevalainen, “A standalone OCR system for mobile cameraphones,” in PIMRC ’06 17th Annu. IEEE Int. Symp. Personal, Indoor and Mobile Radio Communications, Helsinki, Finland, 2006, pp. 1-5.

[10] S. Senda, K. Nishiyama, T. Asahi, and K. Yamada, “Camera-typing interface for ubiquitous information services,” in PERCOM ’04 Proc. 2nd IEEE Annu. Conf. Pervasive Computing and Communications, Orlando, FL, 2004, pp. 366-370.

[11] A. Joshi et al., “OCRdroid: A Framework to Digitize Text Using Mobile Phones,” unpublished.

[12] T. M. Breuel, “The OCRopus open source OCR system,” in Proc. IS&T/SPIE 20th Annu. Symp., 2008.

[13] R. Smith, “An overview of the Tesseract OCR engine.” in ICDAR ’07 Proc. 9th Int. Conf. Document Analysis and Recognition, Curitiba, Brazil, 2007, pp. 629-633.

[14] R. Smith, D. Antonova, and D. Lee, “MOCR ’09 Adapting the Tesseract open source OCR engine for multilingual OCR,” in Proc. Int. Workshop Multilingual OCR, Barcelona, 2009.

[15] R. A. Wilkinson, M. D. Garris, and J. Geist, “Machine-assisted human classification of segmented characters for OCR testing and training,” in IS&T/SPIE Symp. Electronic Imaging: Science and Technology,, San Jose, CA, 1993, pp. 44-54.

[16] D. Blostein, E. Lank, A. Rose, R. Zanibbi, and W. P. Yu, “User interfaces for on-line diagram recognition,” in GREC ’01 Selected Papers from 4th Int. Workshop Graphics Recognition Algorithms and Applications, Kingston, Canada, 2001, pp. 92-103.

[17] E. Lank, J. Thorley, and S. J. Chen, “An interactive system for recognizing hand drawn UML diagrams,” in CASCON ’00 Proc. 2000 Conf. Centre for Advanced Studies on Collaborative Research, Toronto, Canada, 2000, pp. 7.

[18] S. Smithies, K. Novins, and J. Arvo, “Equation entry and editing via handwriting and gesture recognition,” Behaviour and Information Technology, vol. 20, no. 1, pp. 53-67, January, 2001.

[19] R. Zanibbi, K. Novins, J. Arvo, and K. Zanibbi, “Aiding manipulation of handwritten mathematical expressions through style-preserving morphs,” in GRIN ’01 Proc. Graphics Interface 2001, Ottawa, Canada, 2001, pp. 127-134.

[20] J. Mankoff, G. D. Abowd, and S. E. Hudson, “OOPS: a toolkit supporting mediation techniques for resolving ambiguity in recognition- based interfaces,” Computers & Graphics, vol. 24, no. 6, pp. 819-834, December 2000.

[21] Fragoso et al., “TranslatAR: a mobile augmented reality translator on the Nokia N900.

[22] http://www.jdamcd.com/?p=84

[23] http://www.appbrain.com/app/redlaser-barcode-scanner/com.ebay.redlaser

[24] http://www.czechdroid.blogspot.com/

[25] http://code.google.com/p/mezzofanti/

[26] http://www.google.com/mobile/goggles/#text

[27] http://jocr.sourceforge.net/

[28] http://www.gnu.org/software/ocrad/

[29] http://en.openocr.org/

[30] http://finereader.abbyy.com/

[31] http://www.nuance.com/for-individuals/by-product/omnipage/

[32] http://www.expervision.com/ocr-sdk-toolkit/openrtk-ocr-toolkit-sdk

[33] http://pointandfind.nokia.com/

[34] Megan Elmore and Margaret Martonosi. “A morphological image preprocessing suite for ocr on natural scene images”, 2008.

[35] Luo Xi-Ping, Li Jun, and Zhen Li-Xin. “Design and implementation of a card reader based on build-in camera”. In ICPR ’04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR’04) Volume 1. IEEE Computer Society, 2004.

[36] Ohbuchi Eisaku, Hanaizumi Hiroshi, and Hock Lim Ah. “Barcode readers using the camera device in mobile phones”. In CW ’04: Proceedings of the 2004 International Conference on Cyberworlds. IEEE Computer Society, 2004.

[37] “Multimedia Mobile Appliance with OCR”, Axis, Vol. 92, P. 62, Tokyo, 2001 (in Japanese).

[38] www.Forbes.com. Hitachi Mobile phone with OCR. http://forbes.com/technology/feeds/ infoimaging/2005/03/18/ infoimagingasiapulse _2005_03_18_ix_4427-0236-.html

[39] Watanabe Y, Okada Y, Kim Y-B, Takeda T (1998)Translation camera. In: Proc. 14th ICPR, pp 613–617

[40] Yang J, Gao J, Zhang Y, Waibel A (2001) “Towards automatic sign translation”. In: Proc. Human Language Technology.

[41] Yang J, Gao J, Zhang Y, Chen X, Waibel A (2001) “An automatic sign recognition and translation system”. In: Proc. workshop on perceptive user interfaces (PUI’01).

[42] Kamada H, Fujimoto K (1999) “High-speed, highaccuracy binarization method for recognizing text in images of low spatial resolutions”. In: Proc. ICDAR, pp 139–142.

APPENDIX A-1

USER MANUAL

Text Recognition using

based Mobiles

USER GUIDE

TABLE OF CONTENTS

1. Reading Instructions

This Manual is a guide to the system “Text Recognition using Android based Mobile”

It contains essential instructions for setup and operations.

The system provides a user friendly interface which allows you to directly interact and monitor the system.

This Manual should be read in the order given.

2. Installation

For installation ‘.apk’ file of our software is required .This apk file should be directly downloaded to the mobile’s SD Card. Only one click to execute this .apk file would install the software to the mobile’s applications. During installation and while operating the software internet connection must be active. Also Google translate application must be installed in the mobile previously to use the text to translate application.

3. How to use the system

Operation of Text Recognition System comprises of following steps:

1. Capturing of Image

2. Use of Recognized Text in the applications provided by the system.

These steps are in order and they encapsulate the application logic. User need not go into the detail. If detail analysis is required user should read the technical thesis document.

The Applications provided by the system are:

- Translation
- SMS
- Transliterate

- Email
- Auto-Contact
- Google Search

4. Procedure

The underlying procedures are specific for each application.

4.1 Translate

Steps for the successful completion of Translate application are given as under:

1. Capture the image containing the text you want to translate. But capture it when continuous preview shows the best match.
2. Long press the button below the recognized text.
3. Now select the TRANSLATE application
4. Tap on the text bar and select paste option
5. Press Enter
6. Translated text will be now available to you.

4.2 Email

Steps for the successful completion of Email application are given as under:

1. Capture the image containing the text you want to translate. But capture it when continuous preview shows the best match.
2. Long press the button below the recognized text.
3. Now select the EMAIL application
4. Add subject and email id of the recipient
5. Press Enter
6. Email will be successfully sent and message will be displayed

4.3 SMS

Steps for the successful completion of SMS application are given as under:

1. Capture the image containing the text you want to translate. But capture it when continuous preview shows the best match.
2. Long press the button below the recognized text.
3. Now select the SMS application
4. Add recipient's number
5. Press Enter
6. SMS will be successfully sent and message will be displayed

4.4 Auto-Contact

Steps for the successful completion of Auto-contact application are given as under:

1. Capture the image containing the text you want to translate. But capture it when continuous preview shows the best match.
2. Long press the button below the recognized text.
3. Now select the Auto-Contact application
4. Validate the information retrieved
5. Press Enter
6. Contact will be saved successfully and another option will be available to call the contact which can be availed by the user.

4.5 Transliterate

Steps for the successful completion of Transliterate application are given as under:

1. Capture the image containing the text you want to translate. But capture it when continuous preview shows the best match.
2. Long press the button below the recognized text.
3. Now select the Transliterate application
4. Tap the bar and select Paste option
5. Press Enter and wait
6. Transliteration will be done successfully

4.6 Google Search

Steps for the successful completion of Google Search application are given as under:

1. Capture the image containing the text you want to translate. But capture it when continuous preview shows the best match.
2. Long press the button below the recognized text.
3. Now select the Google Search application
4. Tap the bar and select Paste option

Press Enter and then surf the internet for your required information