IMAGE-TEXT TRANSLATION USING AUGMENTED REALITY

CS09-608(P) B.Tech Mini Project 2014

Done By

Albince Paliakkara - ETALECS005
Amritha Rajan - ETALECS006
Jithesh Vijayakumar - ETALECS029
Lakshmi Sasidharan - ETALECS035

Guided By

Helen K J
Associate Professor & HOD

Department of
Computer Science and Engineering
Government Engineering College
Thrissur - 680009

Abstract

Image-Text translation using augmented reality introduces an image text translator for the Malayalam language. The present scenario lacks implemented applications on the proposed idea. This project is intended to produce an application which is capable of converting English phrases extracted from an image source to corresponding Malayalam words and outputting the latter as a replica of former by the principles of augmented reality.

The project implementation has three main parts. First an android camera app which could capture an image from which text can be extracted as OCR. Second is the main Engine which converts the extracted text to the meaningful and lexical Malayalam phrase and finally the converted text takes its place in the original image using augmented reality. Finally the end product is the text in the image on the screen is completely translated to Malayalam.

Acknowledgement

We are very thankful to everyone who supported us. We are equally grateful to Prof. Helen Miss for guiding and correcting various documents with attention and care.She has taken pain to go through the project and make necessary correction as and when needed. She gave us moral support and guided us in different matters regarding the topic.She had been very kind and patient while suggesting us the outlines of this project and correcting our doubts. We thank her for her overall support.

We are also thankful to all developers of various softwares that we came into use during the course of our project. We would also like to thank the stackoverflow.com team for answering our doubts in the limited amount of time. A big thanks to google search engine for endless matching answers for our questions and also sharelatex.com for their online latex editor. We also thank our friends who has supported and helped us to complete this project successfully.

Contents

List of Figures

List of Tables

Nomenclature

APK	Android Application Package
AR	Augmented Reality
ICS	Ice Cream Sandwich
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IRSTLM	la Ricerca Scientifica e Tecnologica Language Modelling
NDK	Native Development Kit
OCR	Optical Character Recognition
OpenCV	Open Source Computer Vision
PC	Personal Computer
SDK	Software Development Kit
SILPA	Swathanthra Indian Language Processing Applications
STM	Statistical Translation Model
TCP	Transmission Control Protocol
WLAN	Wireless Local Area Network
WWW	World Wide Web

Chapter 1

Introduction

1.1 Problem Statement

The aim of the project is to develop an Android application to translate the image text from English to Malayalam using the techniques of image processing, language translation and augmented reality.

1.1.1 Current situation

Android platform currently have many image text translators of different languages. According to the information available, currently there is no translator available which can translate text present in images such as signboards, informative boards etc to Malayalam language. A Malayalam translation of information in such boards will be very much useful in the day to day life of common man having difficulty with the English language.

1.1.2 Proposed system

We are planning to develop an application in Android platform which can translate the English text present in signboards and other informative boards to Malayalam and superimpose them on the pre existing text in the image using augmented reality principles. The abstract operation of such a Android application is as follows:

- English text is extracted from the signboard and is translated to its equivalent Malayalam text.
- The resulting translated Malayalam text is superimposed on the existing image using the principles of augmented reality.
- The new image is made visible to the user.

1.2 Scope of Project

The number of android device users has drastically increased in the past two years. At this point of time, there are some who find difficult in handling English language in their day to day activities, because of which development of such an application in Android platform gain considerable significance. We are planning to develop an application which benefits the common man to overcome the difficulties with English language in his day to day activities to a certain extend.

1.2.1 Technical Feasibility

We have analysed the technical feasibility of the project and it is feasible based on the following factors.

1.2.1.1 Hardware Feasibility

The minimum hardware requirements for developing image text translator are given below:

- An Android Device with Good Camera (Above 5 MP)
- PC Client
- PC Server

1.2.1.2 Software Feasibility

- Open CV - Image Processing Library
- tesseract-ocr (tess-two library)
- Android SDK with ADT bundle
- Moses STM
- GIZA++ Word Aligner
- IRSTLM Opensource Language Modelling Tool Kit
- SILPA Transliteration
- GOOGLE Transliteration API
- Apache Web Server
- OS-Ubuntu 13.10

1.2.2 Financial Feasibility

1.2.2.1 Development Cost

For developing an application no particular cost is required. But devices are needed to demonstrate its working. The only cost required is the cost of devices needed for demonstration of the working Application.

1.2.2.2 Installation Cost

No particular installation cost is needed other than the cost of hardware devices.

1.2.2.3 Operational Cost

Execution of the application does not actually require any operational cost. The only operational cost required is the cost of power supplies to hardware devices as well as the connectivity charges.

1.2.2.4 Maintenance Cost

No particular maintenance cost is needed for this software. Only cost required is to update source translation corpus with the time,and its cost is negligible.

1.2.3 Operational Feasibility

The operational feasibility of the application is dependent on the following factors:

- Lighting Conditions
- Sharpness and Quality of text in the image
- Text font family and its size
- Language Corpus Availability
- Relative motion of the android device and the text image

1.3 Process Model

1.3.1 Iterative and Incremental Model

Iterative and Incremental model is selected for the project. We are planning to implement the system with basic facilities only. So many future enhancements are possible, which are listed below. Iterative and Incremental model can satisfy this requirement efficiently. Since it follows the plan-do-check-act for the improvement,backtracking can done easily in Iterative and Incremental model.

1.3.1.1 The Basic Working Model

The system consists of three independent software products :

- The Android Application
- The Local Server Application
- The Processing Client

The communication between all these softwares is critical in the working of the entire system.

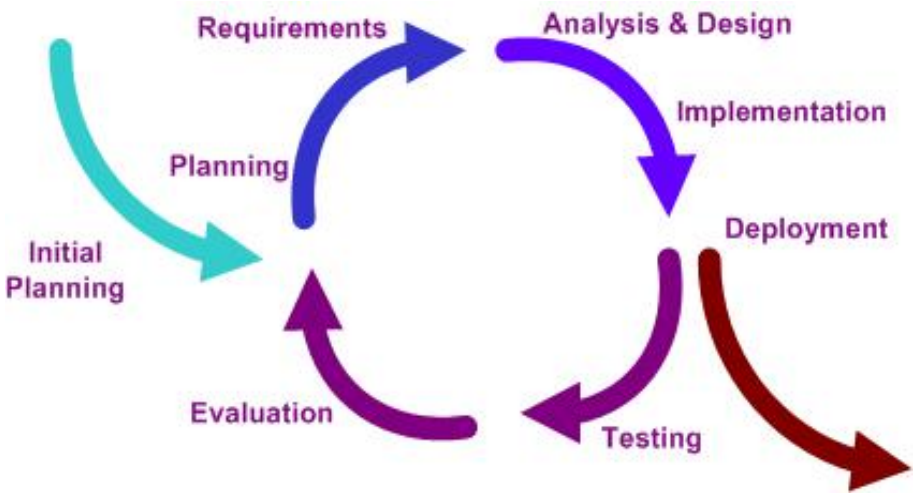
- The mobile application will be communicating with the local server through WLAN protocol presently . Later on it can be implemented through TCP/IP protocol.
- The local server application communicates with the client pc through TCP/IP protocol.

1.3.1.2 Model Description

Iterative and Incremental development is any combination of both iterative design or iterative method and incremental build model for development. The combination is of long standing and has been widely suggested for large development efforts.During software development,more than one iteration of the software development cycle may be in progress at the same time,and this process may be described as an 'evolutionary acquisition' or 'incremental build' approach." The relationship between iterations and increments is determined by the overall software development methodology and software development process. The exact number and nature of the particular incremental builds and what is iterated will be specific to each individual development effort.

Why Iterative and Incremental Model?

- Generates working software quickly and early during the software life cycle.
- More flexible less costly to change scope and requirements.
- Easier to test and debug during a smaller iteration.
- Customer can respond to each built.
- Lowers initial delivery cost.
- Easier to manage risk because risky pieces are identified and handled during the iteration.



Iterative and Incremental Mode can solve issues like:

- Resource wastage
- Costly modifications
- Unclear requirements
- Queued Waiting

1.3.1.3 Future Increments

The future enhancements of the proposed system that we have estimated are given below.

- Extending the application to support new languages
- Cross platform implementation of the Application
- Offline translation support
- Improving the User Interfaces according to user feed-backs

Chapter 2

Requirement Analysis

2.1 Method of requirement elicitation

The techniques we have used for gathering requirements from users were:

- Questionnaire
- Interview

2.1.1 Questionnaire

We delivered a set of questions to users for analysing their feedback,a sample Questionnaire that was produced before the users is given below:

- Are you an android user?
- Which all languages do you know?

Language	READ	WRITE	SPEAK

- Do you find difficulty in reading English signboards ?
- Have you ever wished to read English text on an image in Malayalam ?
- Would you opt to download and install an application capable of converting English image text to Malayalam?
- Do you like to send random application data to the central database so as to improve efficiency of our application ?
- Any suggestions /Remarks ?

2.1.2 Interview

The same set of questions which were used in the questionnaire was used for the interview too.

2.2 User Requirements

On the basis of requirement survey conducted among the users , we reached to a conclusion that more than 90% of the users are having android devices with them. 80% people are having difficulty in dealing with the English language in day to day activities.The survey also found out that 70% of the users would like to have an application which will translate the English text in the images to Malayalam.Another feature people wanted was to train the application from the random data which other users get synced with the central database.

Here are some of our findings :

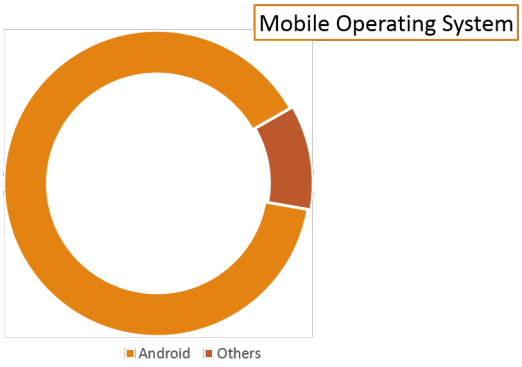


Figure 2.1: Mobile Operating System

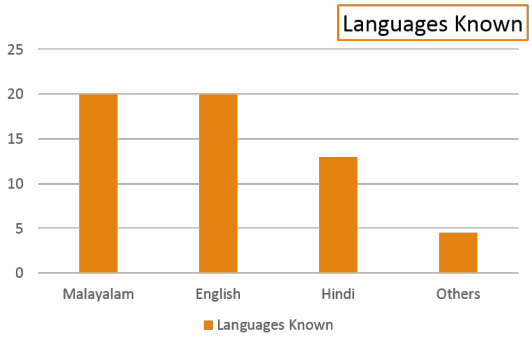


Figure 2.2: Languages Known to Users

2.3 Project Requirements

On the basis of the requirements demanded by the user the following project requirements are found out:

- Ability to read English text from image in android device.
- Ability to translate the text in the image to Malayalam.
- Removal of English text from image by image blurring.
- Adding the translated text over the blurred area.

2.4 Requirement validation

The goal of requirement validation is to seek out and correct problems before resources are committed to implementing the requirements. It is concerned with examining the requirements to certify that they meet the stakeholders intentions and to ensure that they define the right system. The key activities in requirement validation are conducting requirement reviews , demonstrating prototypes,validating the conceptual models etc. We are adopting requirement review as mechanism for requirement validation.

2.4.1 Requirement Review

Initially we have classified the requirements , prioritised them and then negotiated the lower priority requirements. Next step is reviewing the requirements in order to make sure that we have collected the right requirements . The system requirements listed above are complying with the final product.

Chapter 3

Software Requirements Specification

The software requirements specified in this chapter are applicable for the development of an application which translates image text from English to Malayalam using the principles of image processing, translation and augmented reality. This document encompasses those tasks that go into determining the needs or conditions to meet for the application being developed, taking into account the various limitations.

3.1 Product Scope

The need for doing this project is that many people use Android devices at present and at the same time have difficulty with English language. Because of which the development of such an application in Android platform gain considerable significance. We are planning to develop an application which benefits the common man to overcome the difficulties with English language in his day to day life to a certain extent.

3.2 Intended Audience and Chapter Overview

This application is intended to those who are having difficulty with English language in their day to day life. The application helps the targeted audience by extracting the English text on images and translating them to Malayalam language. We know android devices are getting popular and trendy these days. As these devices are portable and handy, the application gains more ease of access and is operational on the go.

3.3 Definitions

- **Augmented Reality**:-a technology that superimposes a computer-generated image on a user's view of the real world, thus providing a composite view.
- **Translation**:- the process of translating words or text from one language into another.
- **Optical Character Recognition**:-conversion of scanned or captured images of typewritten or printed text into machine-encoded/computer-readable text.

3.4 Document Conventions

This document follows IEEE standard format and the conventions followed for fonts, formatting and naming are the standards followed in Computer Science and Engineering Department of Government Engineering College Thrissur.

3.5 Overall Description

3.5.1 Product Perspective

According to the information available,currently there is no application available in the market which is capable of performing the proposed operation. It will be a great advantage for speakers of Malayalam language using android phone and who wants to read English sign boards in Malayalam. Here we are attempting to implement an application which is intended to serve their needs.

3.5.2 Product Functionality

The service offered by the application is given below.

- English text is extracted from the images like signboards and is translated to its equivalent Malayalam text.
- The resulting translated Malayalam text is superimposed on the existing image using the principles of augmented reality.
- The new image is made visible to the user.

3.5.3 Users and Characteristics

By user classes and characteristics we are broadly defining the users who need frequent use of this product and the various requirements of each particular user classes.

- The main stream users of the proposed system are the users who need to print data.
- Another important users of this system are Linux server systems users.

3.5.4 Operating Environment

The operational environment should satisfy the minimum hardware and software requirements.

3.5.4.1 Server Requirements

The minimum hardware requirements needed are,

- Processor:Intel i3 or above.
- RAM:2 GB or above.
- Hard Disk: Freespace 1 GB.
- Ubuntu Server Edition 12.04.4
- Apache Server With Django Web Framework 1.6.1
- Moses Language Engine

3.5.4.2 Phone Requirements

- Dual Core Processor with minimum 512 RAM.
- 5mp camera or above
- Android Based OS Version 4.0.1 (Ice Cream Sandwich) or above

3.5.5 Design and Implementation Constraints

The issues that will limit the options available to the developers are:
Non-availability of translation engine for English Malayalam translation. Since the translation engine is not present we have to develop it from the scratch and the efficiency of conversion is to be brought up to 4 word phrases.

The entire process of the mini project should be completed by April 2014.

3.5.6 User Documentation

User manuals or online help will be provided for the users to download , install and use the application and for troubleshooting. The user manual should describe the steps to be followed for installation and the possible errors that can occur during the application run time. It should also specify the cases where there is a possibility of an error.

3.6 Specific Requirements

3.6.1 External Interface Requirements

3.6.1.1 Hardware Interfaces

The hardware interface required for the application to communicate with the server is through Wi-Fi or Internet connectivity. The android app captures the image,finds the text on it, recognize it and sends the text to the server via connected network. Wi-Fi local area connectivity to the server as well as the Internet connectivity options can be used for sending process requests.

3.6.1.2 Software Interfaces

The application reads the text to be translated from the Android OS through the android base camera app in the device. The data transfer happens by the process of get and post request between the application and the server.

3.6.2 Functional Requirements

The functions of the application is to translate image text from English language to Malayalam. It will have the following phases:

- Read the Text from Image
- The application recognise the text from the image using the OCR engine and pass the image details and text to the server.
- Translation of the text.
- The text that has being received at the server side is being translated from the English language to Malayalam and the translated text and the image position is being communicated back to the application.
- Superimposing translated text over the image text
- The data which is being received back to the application from the server is being positioned over the former image text and aligned accordingly using the principles of augmented reality.

3.6.3 Behaviour Requirements

3.6.3.1 Organizational Structure

The service provided by the proposed application is :
Translate the image text from an image to Malayalam and the same is superimposed on the former using the principles of augmented reality.

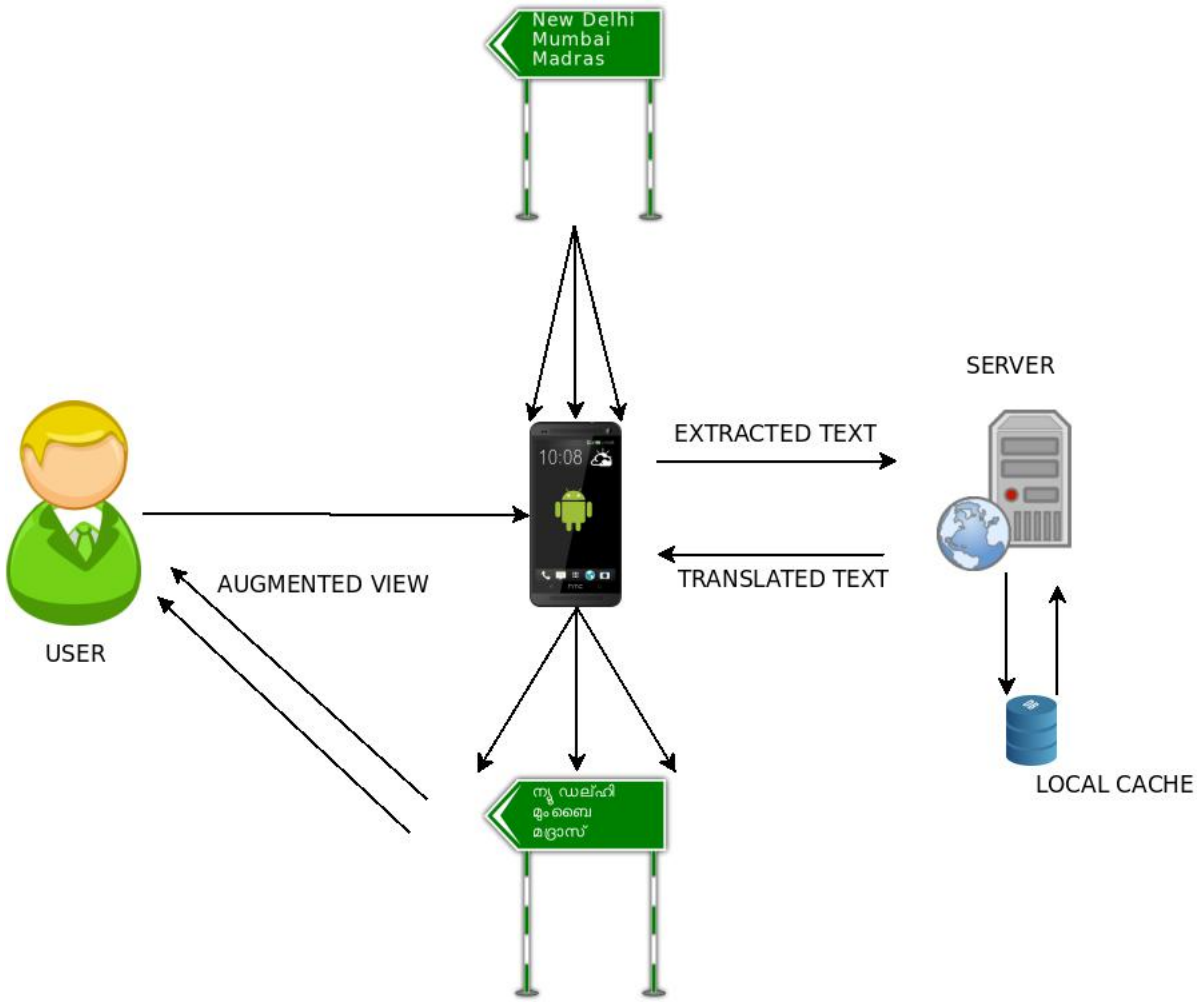


Figure 3.1: Organizational Structural View

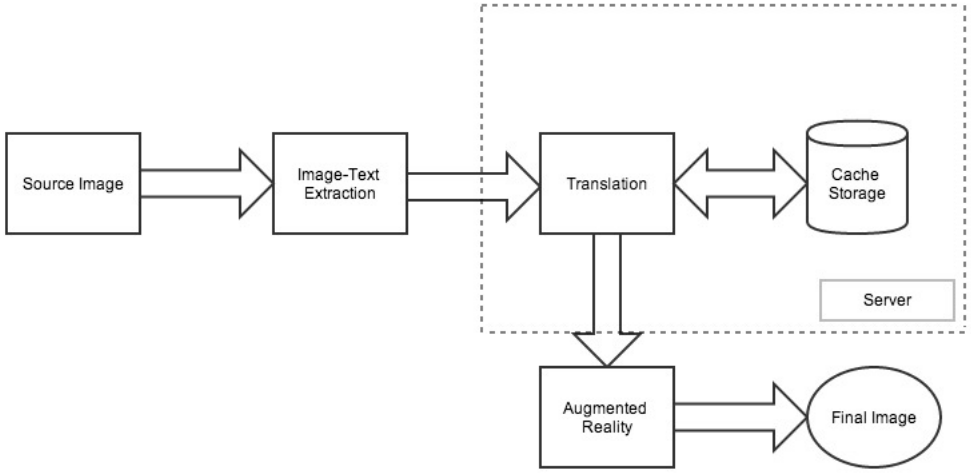


Figure 3.2: Block Diagram

3.7 Non-functional Requirements

3.7.1 Performance Requirements

The main performance requirements that the product should satisfy are:

- Speed:-The speed of translation depends on the complexity of translation request and it has to be brought down to 10sec.
- Resource usage:- The application should not use much system resources and degrade the system performance.

3.7.2 Software Quality Attributes

The most important quality requirements that the system should satisfy are,

3.7.2.1 Scalability

The system should have scalability property.That is the system can be modified for cross platform working by developing replicas of it on other platforms such as IOS, Blackberry and Windows.So the system should be scalable in order to incorporate future changes.

3.7.2.2 Maintainability

The system should be maintainable.Sometimes there might be bugs present in the application.It should be easy to correct when it is reported.

3.7.2.3 Adaptability

The application should have the ability to adapt to the modifications that the server application gets with the language training with little or no modification.

3.7.2.4 Reliability

The application should be reliable.The application should perform the translation of image text efficiently.

3.7.2.5 Testability

The application should be properly tested under various circumstances in order to assure its reliability.

3.7.2.6 Accuracy and Efficiency

The application should be accurate in the information, in our case the output of the system should be accurate, which will make the project an efficient one. The efficiency of the product is also determined by several other conditions and state. The speed and ease of use are some of those.

Chapter 4

Design

4.1 System Design

The application is designed to serve various users operating on android platforms from Android 4.0 (Ice Cream Sandwich) and above. The system design can be broadly divided into the Users Perspective and System Perspective

4.1.1 User Perspective

As soon as the application is opened the user is greeted with the option to select the image source for the translation purpose. The options includes :

- Select an image from gallery.
- Take image from camera.

In accordance to the image source selected, respective options for selecting the image from gallery or to take photo from camera is provided. Once the image has been obtained (from gallery or from camera) the application will execute the operations for extracting the text from the image. The extracted text is then sent to the server through the Internet connection. On receiving the translated text the application will produce the augmented image. This image is made visible to the user.

4.1.2 System Perspective

Image is either captured through camera or imported from the pre existing images in the phone. The selected image is fed into the image detail extractor. The image detail extractor extracts the image text ,font color and location of the text in the image. This text is then send to the translation server using a post request from the application. At the server, the input text is fed into the preconfigured moses translation engine. The engine outputs the translated text in Latin script. This is converted to Malayalam using SILPA transliteration tool. This transliterated text is send back to the client phone for further processing. This translated text along

with the original text location and font colour is given as the input to the augment image generating module. The augmented image generating module first blurs the section of image with text content and then superimposes the translated text onto the blurred portion of the original image. This generated image is given to the User interface as the final output of the system.

4.2 Data Flow Diagram

DFDs are used to represent the flow of data through different modules of the system. An integrated Data Flow Diagram is used to represent the overall system. Here we have included Data Flow Diagrams in different levels.

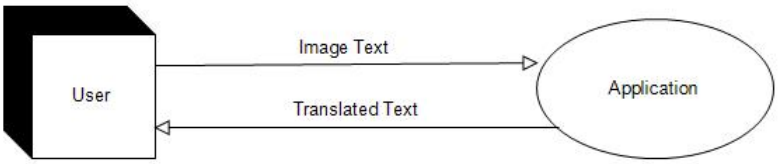


Figure 4.1: Level 0 Data Flow Diagram

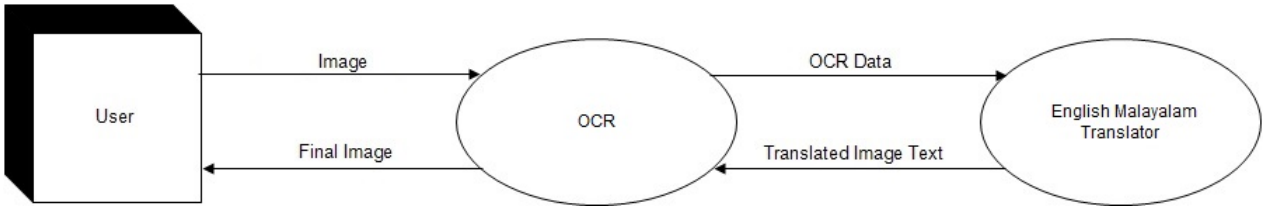


Figure 4.2: Level 1 Data Flow Diagram

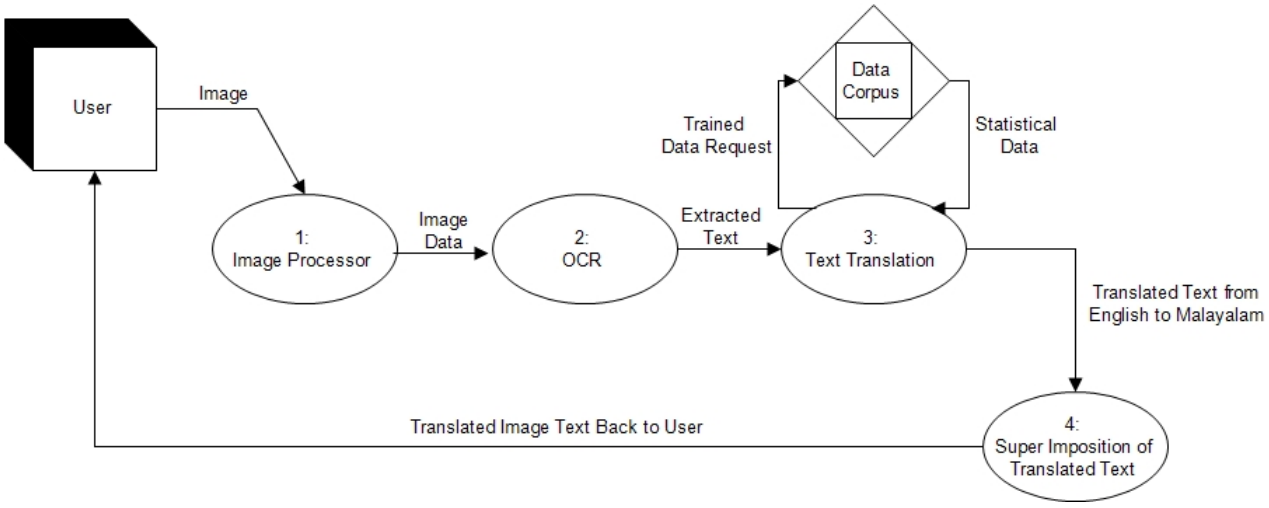


Figure 4.3: Level 2 Data Flow Diagram

4.3 Detailed Design

4.3.1 Sequence Diagram

A sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

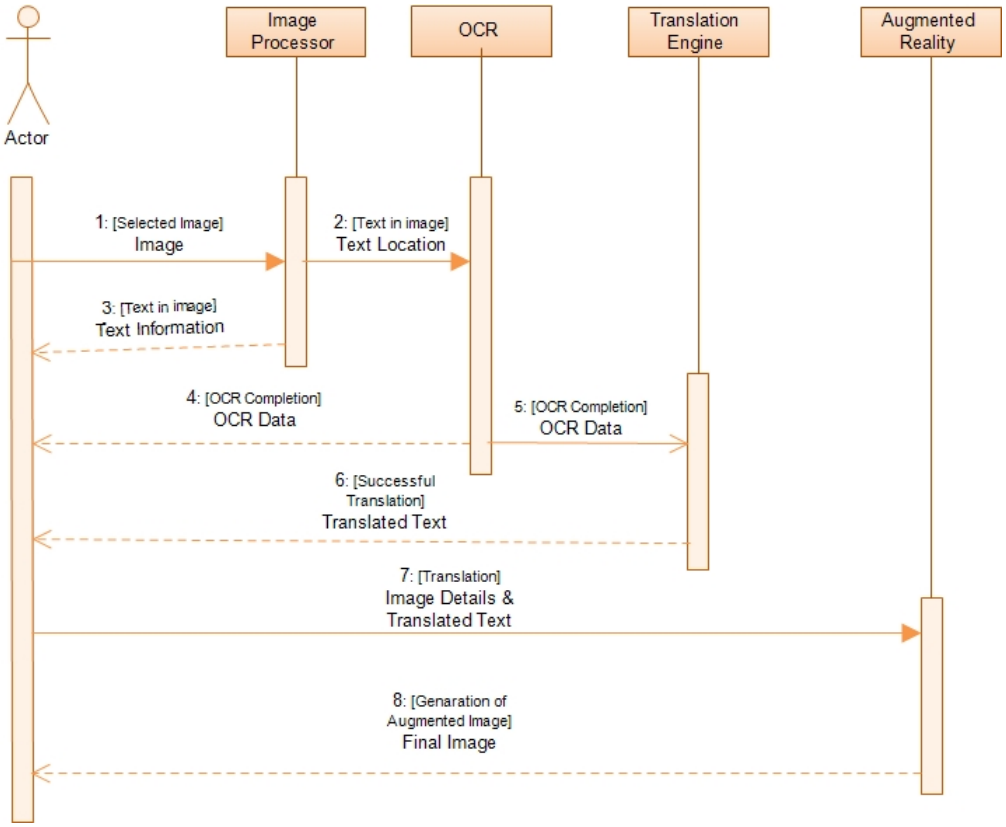


Figure 4.4: Sequence Diagram

4.3.2 Activity Diagram

Activity diagrams are graphical representations of work flows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams are intended to model both computational and organisational processes (i.e. workflows).Activity diagrams show the overall flow of control.

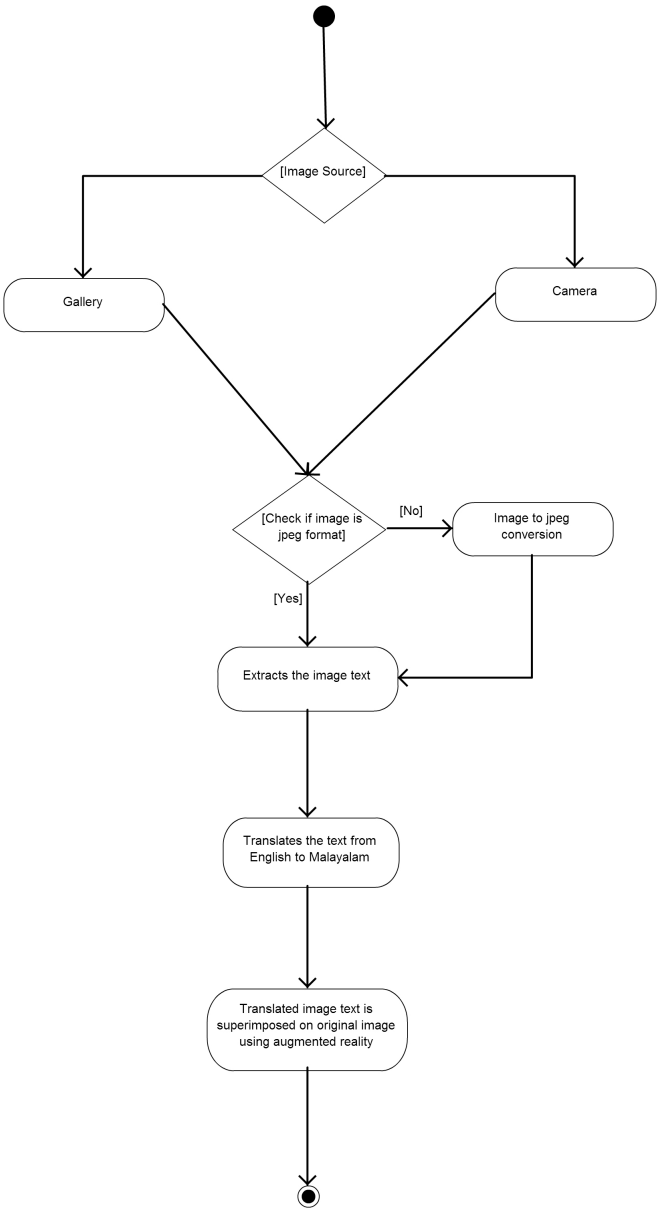


Figure 4.5: Activity Diagram

4.4 User Interface Design

The user interface for the project is being designed as follows:

Figure 4.6: Loading Splash Screen

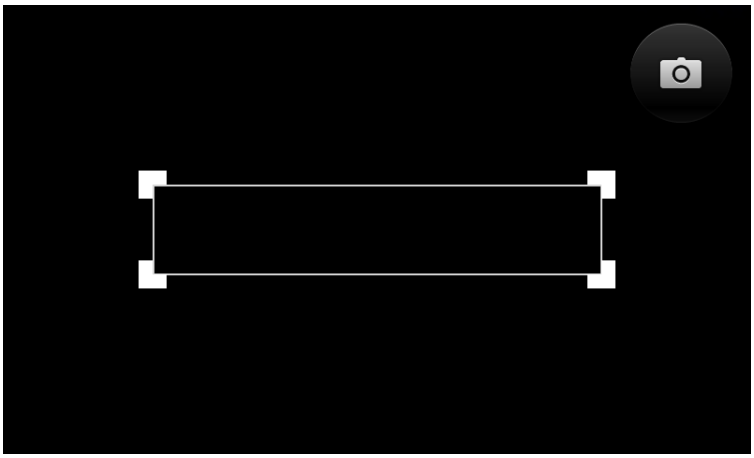


Figure 4.7: GUI Design

Chapter 5

Coding

5.1 Camera Manager

```
package com.gec.cse.miniproject.camera;

..
..

  public synchronized void openDriver(SurfaceHolder holder) throws
      IOException {
    Camera theCamera = camera;
    if (theCamera == null) {
      theCamera = Camera.open();
      if (theCamera == null) {
        throw new IOException();
      }
      camera = theCamera;
    }
    camera.setPreviewDisplay(holder);
    if (!initialized) {
      initialized = true;
      configManager.initFromCameraParameters(theCamera);
      if (requestedFramingRectWidth > 0 && requestedFramingRectHeight > 0)
          {
        adjustFramingRect(requestedFramingRectWidth,
            requestedFramingRectHeight);
        requestedFramingRectWidth = 0;
        requestedFramingRectHeight = 0;
      }
    }
    configManager.setDesiredCameraParameters(theCamera);
```

```
    SharedPreferences prefs =
        PreferenceManager.getDefaultSharedPreferences(context);
    reverseImage = prefs.getBoolean(PreferencesActivity.KEY_REVERSE_IMAGE,
        false);
  }

..
..


  public synchronized void startPreview() {
    Camera theCamera = camera;
    if (theCamera != null && !previewing) {
      theCamera.startPreview();
      previewing = true;
      autoFocusManager = new AutoFocusManager(context, camera);
    }
  }

  ..
  ..

  public synchronized void requestOcrDecode(Handler handler, int message) {
    Camera theCamera = camera;
    if (theCamera != null && previewing) {
      previewCallback.setHandler(handler, message);
      theCamera.setOneShotPreviewCallback(previewCallback);
    }
  }

  ..
  ..

  public synchronized void requestAutoFocus(long delay) {
       autoFocusManager.start(delay);
  }
```

5.2 OCR Activity

```
package com.gec.cse.miniproject;

import java.util.List;

import com.gec.cse.miniproject.R;
import com.gec.cse.miniproject.camera.CameraManager;

..
..


    // Initialize these once for performance rather than calling them
       every time in onDraw().
    paint = new Paint(Paint.ANTI_ALIAS_FLAG);
    Resources resources = getResources();
    maskColor = resources.getColor(R.color.viewfinder_mask);
    frameColor = resources.getColor(R.color.viewfinder_frame);
    cornerColor = resources.getColor(R.color.viewfinder_corners);

    //    bounds = new Rect();
    previewFrame = new Rect();
    rect = new Rect();

 ..
 ..

  public void setCameraManager(CameraManager cameraManager) {
    this.cameraManager = cameraManager;
  }

  @SuppressWarnings("unused")
  @Override
  public void onDraw(Canvas canvas) {
    Rect frame = cameraManager.getFramingRect();
    if (frame == null) {
      return;
    }
    int width = canvas.getWidth();
    int height = canvas.getHeight();

    // Draw the exterior (i.e. outside the framing rect) darkened
    paint.setColor(maskColor);
    canvas.drawRect(0, 0, width, frame.top, paint);
```

```
    canvas.drawRect(0, frame.top, frame.left, frame.bottom + 1, paint);
    canvas.drawRect(frame.right + 1, frame.top, width, frame.bottom + 1,
        paint);
    canvas.drawRect(0, frame.bottom + 1, width, height, paint);

    // If we have an OCR result, overlay its information on the viewfinder.
    if (resultText != null) {

      // Only draw text/bounding boxes on viewfinder if it hasn't been
          resized since the OCR was requested.
      Point bitmapSize = resultText.getBitmapDimensions();
      previewFrame = cameraManager.getFramingRectInPreview();
      if (bitmapSize.x == previewFrame.width() && bitmapSize.y ==
          previewFrame.height()) {


        float scaleX = frame.width() / (float) previewFrame.width();
        float scaleY = frame.height() / (float) previewFrame.height();


 ..
 ..

  public void addResultText(OcrResultText text) {
    resultText = text;

 ..
 ..


}
```

5.3 OCR Text

```
package com.gec.cse.miniproject;

..
..

    this.text = text;
    this.wordConfidences = wordConfidences;
    this.meanConfidence = meanConfidence;
    this.bitmapDimensions = bitmapDimensions;
    this.regionBoundingBoxes = regionBoundingBoxes;
    this.textlineBoundingBoxes = textlineBoundingBoxes;
    this.stripBoundingBoxes = stripBoundingBoxes;
    this.wordBoundingBoxes = wordBoundingBoxes;
    this.characterBoundingBoxes = characterBoundingBoxes;
  }

..
..

  public String getText() {
    return text;
  }

 ..
 ..

  public List<Rect> getCharacterBoundingBoxes() {
    return characterBoundingBoxes;
  }

  @Override
  public String toString() {
    return text + " " + meanConfidence;
  }
```

5.4 Translate Request

```
try {
        String source = URLEncoder.encode(sourceText, "UTF-8");
        HttpClient Client = new DefaultHttpClient();
        String URL = "localhost:8080/translate/android/?txtSrc="+source;
        String SetServerString = "";

        // Create Request to server and get response

         HttpGet httpget = new HttpGet(URL);
         ResponseHandler<String> responseHandler = new
             BasicResponseHandler();
         SetServerString = Client.execute(httpget, responseHandler);

          // Show response on activity
        return SetServerString;
      //return Translate.DEFAULT.execute(sourceText,
          Language.fromString(sourceLanguageCode),Language.fromString(targetLanguageCode));
    } catch (Exception e) {
      Log.e(TAG, "Caught exeption in translation request.");
      return Translator.BAD_TRANSLATION_MSG;
```

5.5 Moses Request Handle

```
echo '$post=[TxtSrc]' | moses -f phrase-model/moses.ini -v 2
```

Chapter 6

Testing

6.1 Testing Methods

There are a number of software testing methods:

1. Unit testing: Unit testing focuses verification effort on the smallest unit of the software design, the module. This is known as module testing. Since the proposed system has modules, the testing is individually performed on each module.

2. Integration testing : Data can be tested across an interface. One module can have adverse effect on another sub function. When combined it may not produce the desired function. Integration testing is a systematic technique for constructing the program structure while at the same time conducting test to uncover errors associated within the interface.

3. Validation testing: Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirements.

4. Output testing : After performing the validation testing, next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format. The output generated or displayed by the system under consideration is tested by asking the users about the format required by them.

5. Acceptance testing : User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required.

6.2 Types of testing done

6.2.1 Whitebox testing

Whitebox testing (also known as clear box testing, glass box testing, trans- parent box testing, and structural testing) tests internal structures or workings of a program, as opposed to the functionality exposed to the end user. In whitebox testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, eg. InCircuit Testing (ICT).

While whitebox testing can be applied at the unit. Integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

Techniques used in whitebox testing include:

1. API testing (application programming interface) testing of the application using public and private APIs
2. Code coverage creating tests to satisfy some criteria of code coverage (e.g.,the test designer can create tests to cause all statements in the program to be executed at least once)
3. Fault injection methods intentionally introducing faults to gauge the efficiency of testing strategies
4. Mutation testing methods
5. Static testing methods

Code coverage tools can evaluate the completeness of a test suite that was created with any method, including black box testing. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested. Code coverage as a software metric can be reported as a percentage for:

1. Function coverage, which reports on functions executed

2. Statement coverage, which reports on the number of lines executed to complete the test 100 percent statement coverage ensures that all code paths, or branches (in terms of control ow) are executed at least once. This is helpful in ensuring correct functionality, but not sufficient since the same code may process different inputs correctly or incorrectly

Sl No	**Modules**	**Function**	**Expected Output**	**Whether obtained the expected output or not**
1	App launcher	public void run() startActivity(i); finish();	Open application	Yes
2	Image Capture	public synchronized void requestAutoFocus(long delay) autoFocusManager.start(delay);	Open camera and capture image on touching the camera button	Yes
3	OCR	public String getText()	Detect English Text from the image	Yes
4	Translate	echo $post = [TxtSrc]\|moses - fphrase - model/moses.ini - v2$	Translate to text to Malayalam Language	Yes

Table 6.1: Whitebox Testing

6.2.2 Blackbox testing

Blackbox testing treats the software as a "black box", examining functionality without any knowledge of internal implementation. The tester is only aware of what the software is supposed to do, not how it does it. Blackbox testing methods include: equivalence partitioning, boundary value analysis, all pairs testing, state transition tables, decision table testing, fuzz testing, model based testing, use case testing, exploratory testing and specification based testing.

Specification based testing aims to test the functionality of software according to the applicable requirements. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behaviour), either is or is not the same as the expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or nonfunctional, though usually functional.

Specification based testing may be necessary to assure correct functionality, but it is insuffcient to guard against complex or highrisk situations.

One advantage of the black box technique is that no programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasise different areas of functionality. On the other hand, blackbox testing has been said to be like a walk in a dark labyrinth without a ash light. Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case, or leaves some parts of the program untested.

This method of test can be applied to all levels of software testing: unit, integration, system and acceptance. It typically comprises most if not all testing at higher levels, but can also dominate unit testing as well

Sl No	**Modules**	**Expected Output**	**Whether obtained the expected output or not**
1	App launcher	Open application	Yes
2	Image Capture	Open camera and capture image on touching the camera button	Yes
3	OCR	Detect English Text from the image	Yes
4	Translate	Translate to text to Malayalam Language	Yes

Table 6.2: Blackbox Testing

6.3 Advantages and Limitations

6.3.1 Advantages

- Open Source Software.
- Uses on device Camera.
- Android ICS+ Compatible.
- Low cost of implementation.

6.3.2 Limitations

- No offline support.
- Superimposition of translated text over the original text was to be sacrificed so as to prevent distortion of line and word spacing of the source image text.
- 65Mb of dataspace required.

6.4 Future Extensions if possible

- Extending the application to support new languages
- Cross platform implementation of the Application
- Offline translation support
- Improving the User Interfaces according to user feed backs

Chapter 7

Quality Assurance

7.1 Introduction

Software quality assurance (SQA) consists of a means of monitoring the software engineering processes and methods used to ensure quality. SQA encompasses the entire software development process, which includes processes such as requirements definition, software design, coding, source code control, code reviews, change management, configuration management, testing, release management, and product integration. SQA is organised into goals, commitments, abilities, activities, measurements, and verifications.

7.1.1 Purpose

The aim of the project is to develop an Android application to translate the imagetext from English to Malayalam using the techniques of image processing, language translation and augmented reality.

7.1.2 Scope

Every member in our team is responsible for the actions planned in this document such as documenting the results throughout the development of the project, reviewing the project progress, and testing the project quality, controlled by this plan. The following are the portions of the software lifecycle that are covered by the SQAP: User requirements, Software requirements, Design, Implementation and Testing.

The list of software items to be covered in each of the above mentioned lifecycle phases are given below:

Software Lifecycle Phase	Software Item
User requirements	User requirement document
Software requirement	Software Requirement Specification document
Design	Software Design Document
Implementation	Document including template of functions
Testing	Document showing testing done in project with summary results

The Software Quality Assurance Plan covers the software items. In addition, the SQAP is also covered by this quality assurance plan.

7.1.3 Project Overview

7.1.3.1 Background and Context

Our project is to create an application capable of detecting English text from image and translate it to Malayalam.

7.1.3.2 Project Objectives

Through this project our aim is to provide an efficient android application to detect English text from image and translate it to Malayalam.

7.2 Quality Assurance Strategy

To assure quality of software deliverable in each software development phase, we will use the test factor/test phase matrix. The matrix has two elements. Those are the test factor and the test phase. The risks coming from software development and the process for reducing the risks should be addressed by using this strategy. The test factor is that the risk or issue which is needed to be tackled, and the test phase is that the phase of software development which conducts the test. The matrix should be customised and developed for each project. Thus, we will adapt the strategy to our project through four steps.

In the first step, we will select the test factors and rank them. The selected test factors such as reliability, maintainability, portability or etc, will be placed in the matrix according to their ranks.

The second step is for identifying the phases of the development process. This phase should be recorded in the matrix. The last step is that deciding the test phase of addressing the risks. In this step, we will decide that which risks will be placed at each development phase.

The matrix forms a part of the quality assurance strategy and as mentioned above this matrix would be used in each of the project lifecycle phases to identify the risks associated in each of the phases with respect to the testing factors. The risks would also be accompanied with their mitigation strategies and in case the risk materialised into a problem, the respective mitigation would be applied. It is for

Testphase/Test	Requirements	Design	Coding	Testing factors
Correctness	Risk: The SRS may not be correct as per the goals of the SQAP Strategy : Formal Techview of SRS	Risk: Software design document may not be correct as per the SRS Strategy : Formal Techview of SRS	Risk: For lengthy code functions defined in software design document yearn more for code correctness	Risk : User may not give the correct input
Performance	Risk: User may have to compromise.	Risk: Software design document may not have the performance as per the requirement	Risk: Lack of syntactic structures in selected language selected.	Risk : May have several input cases which affect the performance.
Continuity of processing	Risk: Unsatisfiable requirements.	Risk: Improper way of designing requirements	Risk: Possibility of error occurrences	Risk: Possibility of error occurrences

these reasons, that a mention is made about the matrix here in a separate section of the document and not mixed with other sections of the document to avoid repetition.

7.3 Audits and Reviews

7.3.1 Work Product Reviews

Work Product	When reviewed by Quality Assurance	How reviewed by Quality Assurance
Software requirement specification	After a new release	All the requirements raised by the user are achieved.
Software document design	After modication	Android application developed. Development done in Java.
Coding	New release	All algorithm and designs coded and implemented perfectly.
Testing	New release	The product is subjected to both blackbox testing and white box testing. All user requirements are met.

7.4 Further Extension

We have completed our project in all aspects. All functional and non-functional requirements of all four modules are met. The source code was downloaded and compiled in Linux. The product is easily usable and user documentation for all the modules is provided.The system can be extended by adding more features .The project has gone successfully through all the phases of software development and has been tested per the requirements.

Further Extensions include :

- Extending the application to support new languages
- Cross platform implementation of the Application
- Offline translation support
- Improving the User Interfaces according to user feed-backs

Chapter 8

Conclusion

The project works has been completed through the iterative and incremental development life cycle. On our way working on this interesting project, we learned many things.We got an invaluable experience by working on this project and learned various technologies on a need based manner. The system we developed is just a minimal framework upon which anybody can work on to come up with interesting application ideas. We intend to extend the project to the above mentioned possibilities. The sacrifice on the augmented reality phase was a cut edge decision from our part on testing the developed application so as to ease the user using the user application. This project work has been completed in the given course of time sacrificing the augmented phase superimposition part with a substituted interface modification. We also expect that this project motivates others to build on this and create more wonderful, useful utilities.

Appendices

Appendix A

User Document

A.1 Installation

APK (application package files) are a compressed single-file package of an Android app. The APK must be executed once (by tapping on the file) for the application to be installed and able to run on the Android device.

A.1.1 Steps for Installation

1. Download the latest version of application from http://coderjithesh.in/miniproject/.
2. Allow application installs from Unknown Sources : Next you need to let your phone install any APK file, not just the one downloaded on the official App stores. Go to Settings >Security >tick Unknown sources

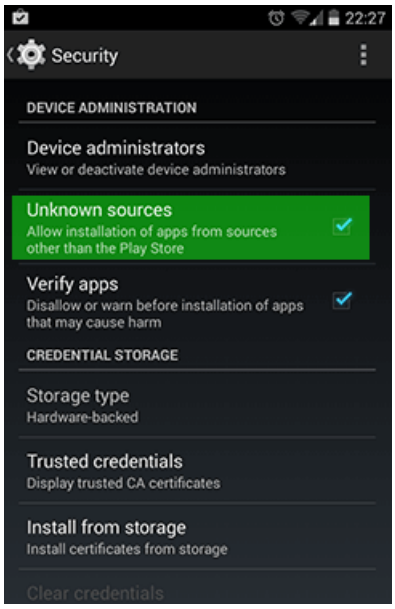


Figure A.1: Install from Unknown Sources

3. Install APK with the file manager : Launch the app, locate your Download folder (or wherever you saved your APK) and tap on them to install them. Once the installation process complete, the icon will appear in your application list.

4. Open Image Text Translator from the application list to use this application.

A.2 Using the Application

- Open Image Text Translator from the application list.

Figure A.2: Welcome Screen

- Adjust the viewfinder to select the text from any image.

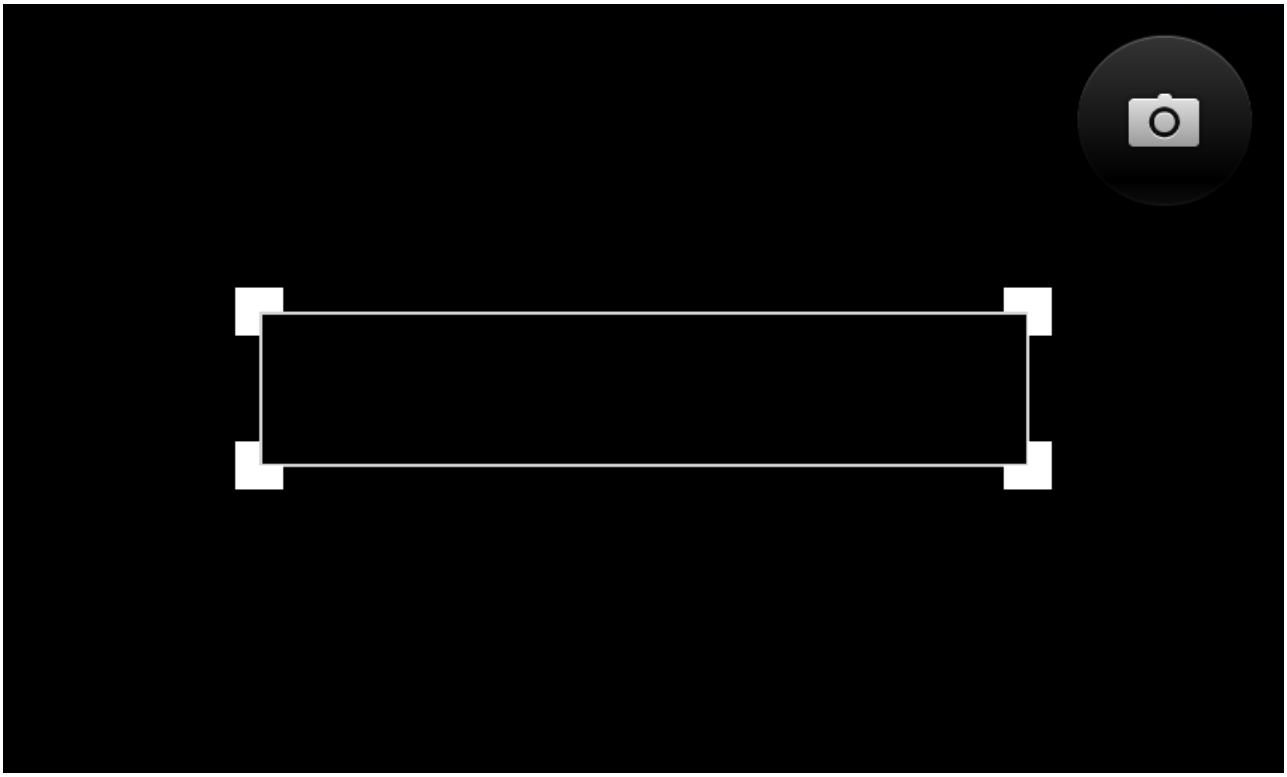


Figure A.3: User Screen

- Touch on camera button to focus and capture the text.

- Please wait for the translated output.

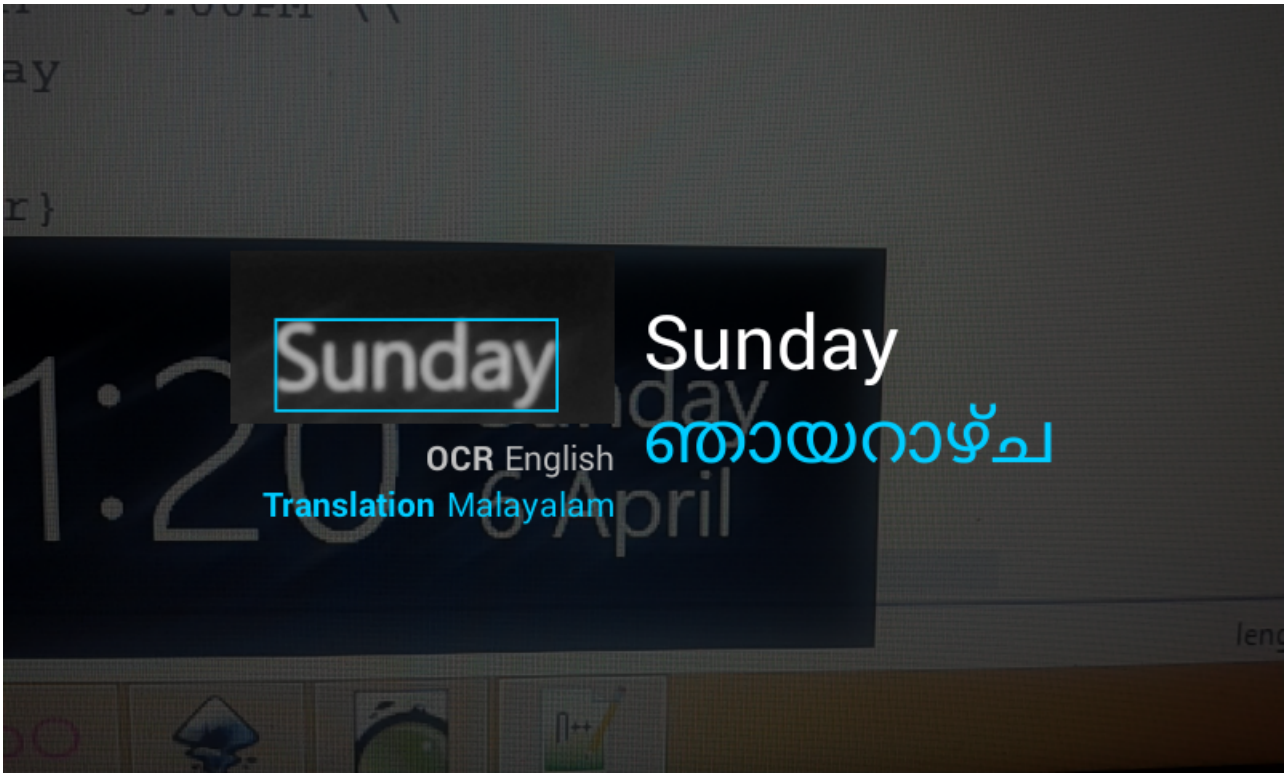


Figure A.4: Output Screen

Bibliography

[1] Ian Sommerville., *Software Engineering*, 8th Ed, June 2006 ISBN 978-0-321-31379-9.

[2] Mark L. Murphy., *The Busy Coder's Guide to Android Development*, Vol.2, pp.60-74, 2012.

[3] Remya Rajan, Remya Sivan, Remya Ravindran, and K.P Soman.A., Rule based machine translation from English to Malayalam. In: *Conference Proceedings on International Conference on Advances in Computing, Control, and Telecommunication Technologies,* pp.439-441, 2009.

[4] Nithya B, and Shibily Joseph., A Hybrid Approach to English to Malayalam Machine Translation. In: *International Journal of Computer Applications (0975-8887),* Volume 81 No.8, November 2013.

[5] Ayatullah Faruk Mollah, Nabamita Majumder, Subhadip Basu, and Mita Nasipuri., Design of an Optical Character Recognition System for Camera based Handheld Devices . In: *IJCSI International Journal of Computer Science Issues,* Vol. 8, Issue 4, No 1, July 2011.

[6] http://stackoverflow.com/, Question and Answer site for professional and enthusiast programmers.

[7] https://developer.android.com/, Google's site for android developers.

[8] http://forum.xda-developers.com/, Android developers forum.

[9] http://en.wikipedia.org/wiki/Software metric

[10] http://en.wikipedia.org/wiki/Software quality