PPU College Of Engineering and Technology

The Home of Competent Engineers and Researchers

Electrical and Computer Engineering Department

Communication and Electronic Engineering

Bachelor Thesis

Graduation Project

Mobile Phone Translator Application

Project team

Wajdi Hammad

Musa Abusabha

Mussab Almahareeq

Project supervisor

Alaa Shabaneh

Hebron – Palestine

December, 2012

COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Graduation Project

Mobile Phone Translator Application

Project Team

Wajdi Hammad Musa Abusabha Mussab Almahareeq

According to the system of the College of Engineering and Technology, and to the recommendation of the Project Supervisor, this project is presented to Electrical and Computer Engineering Department as a part of requirements of B.Sc. degree in Electrical Engineering – Communication and Electronic Engineering.

PALESTINE POLYTECHNIC UNIVERSITY

Project Supervisor signature

Dr. Alaa Shabaneh Tamimi

[illegible]……………..

Testing Group signature

Dr ……………… Dr…………………

Department Headmaster signature

Dr. Ramzi Qawasmeh

…………………………………

Hebron-Palestine

ABSTRACT :

mobile phone translator application is used to capture images of any text and make all the required image processing phases until getting a thresholded image of characters, then starting to extract letters and get features for these letter, the last phase is comparing these features with already existent database of letters and their corresponding features, and outputs the best matching letter and formulate the word in a text and translates it to any language .

The application is developed for Android phones on platform 2.3.3, API 10, Tesseract library is used version 3, mostly in the preprocessing phase. SQLite is used for the building the database required.

The first phase of the project is to handle getting an image either by capturing an image by the mobile camera, or browsing the gallery for an existent image on the phone. Applying all the required pre-image processing functions from conversion to grayscale, then removing the noise with filter, convert the captured image to threshold image, then using Tess-Two library to find words and characters segmentation, then surrounding these letter with a bounding rectangle to be able to extract each letter.

The second phase is recognition then proceeds as a two-pass process. In the first pass, an attempt is made to recognize each word in turn. Then Each word passed to an adaptive classifier as training data. The adaptive classifier then gets a chance to more accurately recognize text over the page. Since the adaptive classifier may have learned something useful too late to make a contribution, a second pass is run over the page, in which words that were not recognized well enough are recognized again.

A final phase involves about getting all recognized texts and prepare it for translation

The application uses a user-friendly interface that allows you to browse different operations done on the image in the preprocessing part, and control over the process of what to view according to user's need.

Acknowledgment

Over five years of university life many people have helped us. Without their help it would have been very difficult to get to where we are now, and we are immensely grateful.

Firstly all praise our lord the god almighty (Allah) the ultimate guide and the cherisher who gave us the courage and the power to complete this study with satisfying degree of perfection.

No one walks alone on the journey of life. just where do you start to thank those that joined us, walked beside us, and helped us, we are so grateful to our Parents. Thanks to our brothers and sisters for their supports and also thanks to good friends and extended family back home who wish us the best.

Every message or comment we receive from home makes us stronger and even surer about what we are doing. Thanks to everyone who has gone to the effort.

We would like to thank those amazing people we have met along the way who have helped us in small or major ways…

Our thanks for the supervisor Eng. Mashour Al-trayr, who gave us his attention and encouragement, who was abundantly helpful and offered invaluable assistance, support and guidance.

Also thanks for all lecturers and particularly communication staff who spend their expensive time and life to join them in their trip of human lights and experts.

Figures

Tables

Chapter 1

Introduction

Content

1.1 Overview

In the recent ten or twenty years the world witnessed major progress in ways of communication; which makes Individuals from different backgrounds communicate easily even though they speak different languages. Dictionaries remained the most common tool that people use to understand conversations, formal papers written in languages rather than the language they speak .

Dictionaries are great, however there are some disadvantages of using them to translate into other languages, for example translating a conversation or a long essay may take long time , translation using dictionaries can be boring, since you need to translate every word, then combine the words to form the original paragraph, so you lose your time.

So, we need to have a smart tool that could overcome all these drawbacks, in this project we are suggesting a way of implementing such tool.

We wish to present suitable application that allows easy translation, learning and understanding what was written on posters or magazines in other foreign languages.

1.2 The general idea

The general idea is to convert the words within a captured image into text, and then to translate it.

The application offers two options for translation, by Bing translate system, which needs to have internet connection, or through translation using a special dictionary, in this application (project) we will design a dictionary that contains scientific and engineering terms that the translator will use in case of no connection with the internet. The user can send the original text of the image, the translated one, or both to his email or his Facebook account. This gives the advantage of using the application to save time .

1.3 The project importance

This project helps to facilitate communication between the person and the foreign community without need to ask anybody, as well as it helps him to understand the content of banners in different places, the project helps students to facilitate the study process to some extent, through the translation of the meanings of incomprehensible words or new ones with ease , also saves time and effort in the process of printing the source is up printing through photography with making some adjustments if necessary.

1.4 Objectives

1- Extracting the text in the image and send it as text to a computer or email .
2- Translating the text in the image from one language to another language.
3- Developing our team skills using java in the Android environment .
4- Learn how to design mobile applications .

1.5 Motivation

Help people a foreign country that they do not master its language, or they have intermediate language they use to deal with people .This project will help in facilitating the process of human mobility and coexistence in this country by helping them understand what it contains in signs and banners written in a foreign language, which makes it easy to navigate and inferred addresses and different places easily.

1.6 Requirements

The project will be implemented using Java technologies for Android platform, and also will use Bing Translate system to translate text between languages .

1.7 Literature review

There are many application that make it easier for a tourist to get around a foreign city, this application makes you understand what is written on boarders, Banners, menus of restaurant, advertisements and other regulations. it uses the same technique that we use which is optical character recognition (OCR), some of these application are the following :

1-Google Translate: it's a simple mobile translation tool that supports more than 60 languages, offers an SMS translator, and even speaks some of your translations aloud. With its dead-simple interface and variety of input options, You can type text using your mobile device's keyboard or say the words aloud. If you're an Android user, you can also choose to handwrite on your touch screen or snap a photo and translate text using optical character recognition (OCR) .

2-**ABBYY Text Grabber + Translator** : it is an iPhone application to easily extract text from a variety of printed sources such as books, magazines, ads, timetables, etc. by using the device's camera. With its quick text capturing and translation features, you can digitize printed information and translate it into your native language anywhere anytime. All you need to do is to either take a snapshot of the text in any printed material with your iPhone, or load an earlier taken snapshot from your Smartphone's memory. As soon as the data recognition process is complete, the text will be available for editing, translating, copying to other applications, publishing to Facebook, Twitter, Ever note or sending by e-mail or SMS. It's Translation for more than 40 languages using the Google Translate technology.

3- **iPhone OCR Camera Translator :** Camera Translator for iPhone, iPod Touch is a powerful OCR translation application that allows you to translates words using iPhone and iPod Touch camera from English into Spanish, French, Italian, German and Russian languages. Camera Translator app. has a built-in OCR (Optical Character Recognition) technology that converts words in image to machine readable words. You don't need to type a word to translate, just aim the camera to the chosen word and iPhone OCR Camera Translator will give you a detailed translation immediately.

4-Cam Lingual : It's an Apple application available for iPhone, iPad, and iPod touch devices lets you translate any writing, simply by taking a picture of it. Cam Lingual detects text from pictures using an online OCR service, and translates it using an online translation service. You can either take photos from the camera or choose a picture from your device's photo album to translate instantly. it can translate 28 different input languages, and 52 different output languages include.

5-Word Lens: it is an iPhone-based optical character recognition (OCR) application used for language translation. In real-world usage, it will translate a foreign language sign in an instant using your camera. (English/spinach) [1].

1.8 Time schedule

The project plan follows the following time schedule, which includes the relation between the tasks of the study and the analysis system.

The following table explains the expected timing plan .

Table 1.1: Project Timing Plan Table.

Expected work	Duration of time .
Choose the idea	3 weeks .
Collect information	14 weeks
Data analysis	10 weeks
Design option	7 weeks
Documentation	14 weeks
Programming	13 weeks
Testing	5 weeks

1.8.1 Schedule Table :

The table below shows the activities and efforts done on the project and the time associated with each one .

Table 1.2: Schedule Table

Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Task	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Idea choosing (1st semester)	■	■	■													
Information collection (1st semester)				■	■	■	■	■	■	■	■	■	■			
Data analysis (1st semester)								■	■	■	■	■				
Design options (1st semester)										■	■	■	■	■		
Documentation (1st & 2nd semester)	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
Programming (2nd semester)	■	■	■	■	■	■	■	■	■	■	■	■	■			
Testing (2nd semester)												■	■	■	■	■

1.9 Estimated cost :

The expected cost of the project is the price of a mobile phone running on the Android operating system, which is estimated at about 500 $.

1.10 Report contents:

This report is mainly divided into three chapters:

1.10.1 Chapter one: includes the introduction, which makes a general overview of the system, problem statement, project objectives, and time planning, it give readers a general view about the project and discusses the related works .

1.10.2 Chapter two: discusses the theoretical background.

1.10.3 Chapter three: presents the general system design concepts. It includes system objectives, general system block diagram, description of system operation, hardware design implementation for each part of this project.

1.10.4 Chapter four: Present the project Layout and Interface Design , Activity Life Cycle.

1.10.5 Chapter five: discusses Android Native Development Kit, Android Application Structure, Building the application.

1.10.6 Chapter six: Testing .

Chapter 2

Background and Literature Review

Content

2.1 Overview

In this chapter we discuss the main heading of the mobile applications and the Mobile application development tools, which makes applications development easier than before, also we talk about the mobile operating system and its features, common programming language used to develop mobile applications which is Java language, explain about the eclipse platform environment that will we use to develop our application .

2.2 Localmobile applications

Nowadays mobile devices are a necessity. Except making calls, there are many featureswhich are gaining popularity like camera, music, global position system (GPS), Accelerometer etc. These are kind of built-in features that are provided by all the major available mobile Operating Systems OS's such as OS, and Windows phone. All these mentioned mobile OS's are very much popular in the market because of their uniqueness [2].

Mobile phones may have one of different platform such as: J2ME, Android, Apple iPhone, Windows Mobile,Blackberry,Palm,Nokia (C/C++, Python),S40,Symbian (S60, S80)

In general, there are three different solutions for smart phone development; native, worldwide Web, and hybrid.

Native app is the application that works natively. Native app code is written specifically for a particular phone's operating system. Web app is the application that renders via a Web browser using Web application solutions including HTML, CSS, and JavaScript. Hybrid app is the combination of a native app and a Web app. Some hybrid apps that are non-native apps are developed using JME, .NET CF (Compact Framework) and BREW (Binary Runtime Environment for Wireless) [1] .

One of the important factors in choosing features to integrate into an application is selecting those features that are unique and useful to the target users. There are many hardware features available for smart phones, three of the important and unique features are touch screen, GPS integration, and built-in Wi-Fi. Other features include camera, phone, address book, e-mail, and location tracking.

Mobile application development tools are developed to make application development easier, they simplifies the development to greater extent with other advantages. There are some advantages of mobile tools to overcome mobile operating system limitations[2].

- Developed application would be platform independent. Application could be deployed on any mobile OS.
- Development requires very less coding and is timesaving.
- Deployment of application becomes lot easier than in Mobile OS.

2.3 Mobile operating systems

Most of the newly introduced OS's are very rich in feature, and convenient for developers to create and deploy new applications. The major advantage in these OS's is the built-in features, such as Wi-Fi, Gallery, Bluetooth and Contact etc., then developers need not to develop these from scratch which is a big relief and is timesaving. Although all new OS's are very efficient, they still need a lot of time and investment to develop new applications. Beside these issues, platforms have other limitations as well. The Intensive competition does not give much space for companies to be slow in launching new application. So all these OS's has definitely made developer life comparatively easy but simultaneously has arose many challenges to be competent in the market [1] .

2.3.1 Android platform

The Android platform is a software stack for mobile devices including an operating system, middleware and key applications. Developers can create applications for the platform using the Android SDK. Applications are written using the Java programming language.

Google introduced Android as an OS which runs the powerful applications and gives the users a choice to select their applications and their carriers. The Android platform is made by keeping in mind various sets of users, who can use the available capacity within Android at different levels. Android is gaining strength both in the mobile industry and in other industries with different hardware architectures [3].

Android supports all the following platforms:

- Windows XP (32-bit), Vista (32- or 64-bit), and 7 (32- or 64-bit).
- Mac OS X 10.5.8 or later (x86 only).
- Linux (tested on Linux Ubuntu Hardy Heron).

Note that 64-bit distributions must be capable of running 32-bit applications. We also need to download the Android SDK (which is free) and the Java Development Kit (or JDK, which is also free and because Android applications are developed in the Java programming language, we need to understand the Java language. Android also uses XML quite heavily to define various resources inside the application, so we should understand XML too.

Android can run on many devices with different screen sizes and resolutions. Besides being cross-compatible, android comes with the tools that help developing cross-compatible applications. Google allows applications to run only on compatible devices. If any application requires a front-facing camera, for example, only phones with a front-facing camera will be able to see app in the Android Market. This arrangement is known as feature detection.

2.3.2 Android Architecture

Android Architecture consists of number of layers as Applications, Application framework, Libraries, Android runtime and Linux kernel, which are shown in the following figure.

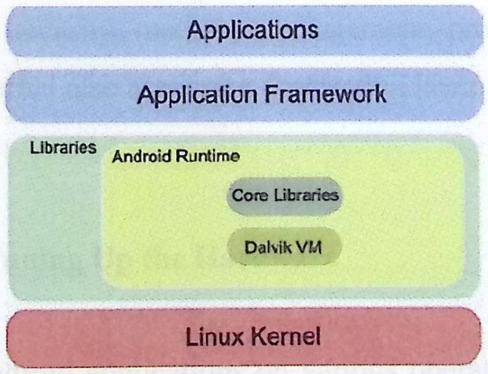


Figure 2.1: Android architecture layers.

Application layer is the uppermost layer which provides a set of core applications including an email, SMS program, calendar, maps, browser, contacts and others.

All applications are written using the Java programming language, it should be mentioned that applications can be run simultaneously; it is possible to hear music and read an email at the same time.

The Application Framework is a software framework, that is used to implement a standard structure of an application for a specific operating system. With the help of managers, content providers and other services programmers, it can reassemble functions used by other existing applications.

Layer which is present below Application framework consist of two parts as Libraries , which are all written in C/C++. They will be called through a Java interface. This includes the Surface Manager, 2D and 3D graphics, Media Codec like MPEG-4 and MP3, the SQL database SQLite and the web browser engine Web Kit.

Second part is Android Runtime which includes a set of core libraries, which provides most of the functionality available in the core libraries of the Java programming language.

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. The Dalvik VM executes files in the Dalvik Executable (.dex) format, which is optimized for minimal memory footprint.

The lowest layer is Linux Kernel, Android basically relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack [3].

2.4 Requirements and Tuning Up the Hardware

We can develop Android applications on various operating systems, including Windows, Linux and Mac OS X. We do the development in this project under a Windows 7 operating system, but we can develop using Mac OS X or Linux instead.

2.4.1 Hardware tools

Google exposes a plethora of functionality in Android, thus giving developers the tools needed to create top-notch, full featured mobile apps. Google has gone above and beyond by making it simple to tap into and make use of all the devices available hardware. To create a spectacular Android app, you should take advantage of all that the hardware has to offer. If you have an idea for an application that doesn't need hardware assistance, that's okay too. Android phones come with several hardware features that we can use to build our apps, as shown in Table 2:1[4].

Android Device Hardware	
Functionality Required	Hardware
Where am I ?	GPS radio
Which way am I walking ?	Built in compass
Is my phone facing up or down ?	Proximity sensor
Is my phone moving ?	Accelerometer
Can I use my Bluetooth headphone ?	Bluetooth radio
How do I record video ?	Camera

Table2.1:Android Device Hardware

2.4.1.1 Touchscreen

Android phones have touch screens, a fact that opens a ton of possibilities and can enhance users' interaction with your apps. Users can swipe, flip, drag and pinch to zoom, for example, by moving a finger or fingers on the touch screen. You can even use custom gestures for your app, which opens even more possibilities. Android also supports multi-touch, which means that the entire screen is touchable by more than one finger at a time. Hardware buttons are old news. You can place buttons of any shape anywhere on the screen [4].

2.4.1.2 SD Card

Android gives us the tools we need to access (save and load) files on the device's SD Card, it's a portable storage medium that we can insert into various phones and computers. If a device is equipped with an SD Card, you can use it to store and access files needed by your application. Android 2.2 allows you to install apps on the SD Card, but maybe your users have phones that don't get Android 2.2. Just because some users don't have the option of installing apps on the SD Card doesn't mean that you have to bloat your app with 20MB of resources and hog the phone's limited built-in memory. You can download some or all of your application's resources from your Webhost and save them to the phone's SD Card. This makes your users happy and less likely to uninstall your app when space is needed [4].

2.4.2 Development tools

Various Android tools are needed while writing Android applications. In the following sections, we outline some of the most popular tools that we will use in our day-to-day Android development process.

2.4.2.1 Internet

Thanks to the Internet capabilities of Android devices, real-time information is easy to obtain. As a user, you can use the Internet to see what time the next movie starts or when the next commuter train arrives. As a developer, you can use the Internet in your apps to access real-time, up-to-date such as weather, news and sports scores. You can also use the Web to store some of your application's assets, which is what Pandora and YouTube. Why not offload some of your application's intense processes to a Web server when appropriate? This can save a lot of processing time in some cases and also helps keep your Android app streamlined. This arrangement is called client -server computing-a well-established software architecture in which the client makes a request to a server that is ready and willing to do something. The built-in Maps app is an example of a client accessing map and GPS data from a Web server.

2.4.2.2 Audio and video support

The Android OS makes including audio and video in your apps a breeze. Many standard audio and video formats are supported. Including multimedia content in your apps couldn't be any easier. Sound effects, instructional videos, background music, streaming video and audio from the Internet can all be added to your app with little to no pain.

2.4.3 Security

Android allows your apps to do a lot imagine if someone released an app that went through the contact list and sent the entire list to a server somewhere for malicious purposes. This is why most of the functions that modify the user's device or access its protected content need to have permissions to work. Suppose that you want to download an image from the Web and save it to the SD Card. To do so, you need to get permission to use the Internet, so that you can download the file. You also need permission to save files to the SD Card. Upon installation of the application, the user is notified of the permissions that your app is requesting. At that point, the user can decide whether he wants to proceed with the installation. Asking for permission is as easy as implementing one line of code in your application's manifest file.

2.4.4 Google APIs

The Android OS isn't limited to making phone calls, organizing contacts or installing apps. You have much more power at your fingertips. As a developer, you can integrate maps into your application. To do so, you have to use the maps APIs that contain the map widgets [4].

2.4.5 Assembling Your Toolkit

Now it's a time to be an Android developer, we need a computer and get cracking on installing the tools and frameworks necessary to build first blockbuster application.

2.4.6 Android source code

We should be aware that the full Android source code is open source, which means that it's not only free to use, but also free to modify. If we'd like to download the Android source code and create a new version of Android, we're free to do. We can also download the source code.

2.4.7 Linux 2.6 kernel

Android was created on top of the open-source Linux 2.6 kernel. The Android team chose to use this kernel, because it provided proven core features to develop the Android operating system on. The features of the Linux 2.6kernel include (but aren't limited to) the following:

- **Security model**: The Linux kernel handles security between the application and the system.
- **Memory management:** The kernel handles memory management for you, leaving you free to develop your app. Process management: The Linux kernel manages processes well, allocating resources to processes as they need them.
- **Network stack:** The Linux kernel also handles network communication.
- **Driver model:** The goal of Linux is to ensure that everything works.

Hardware manufacturers can build their drivers into the Linux build [4].

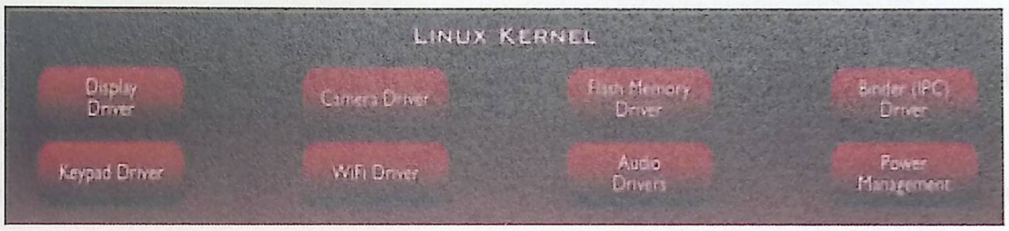


Figure 2.2: Some of the Linux kernel features.

2.4.8 Android framework

Atop the Linux 2.6 kernel, the Android framework was developed with various features,these features were pulled from numerous open-source projects. The output of these projects resulted in the following:

- **The Android run time**: The Android run time is composed of Java core libraries and the Dalvik virtual machine.
- **Open GL (graphics library):** This cross-language, cross-platform application program interface (API) is used to produce 2D and 3D computer graphics.
- **Web Kit:** This open-source Web browser engine provides the functionality to display Web content and simplify page loading.
- **SQLite:** This open-source relational database engine is designed to be embedded in devices.
- **Media frameworks:** These libraries allow you to play and record audio and video.
- **Secure Sockets Layer (SSL)**: These libraries are responsible for Internet security.

The following Figure show the most common Android libraries [4].

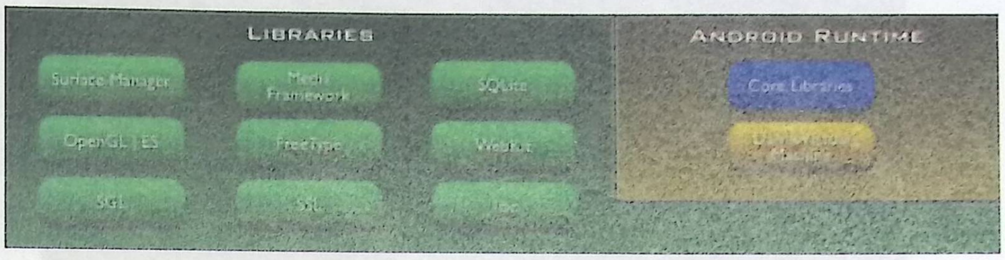


Figure 2.3: Common Android libraries.

2.4.9 Application framework

The Android team has built on a known set of proven libraries, all exposed through Android interfaces. These interfaces wrapped up the various libraries and made them useful to the Android platform as well as useful to a developer. Android has all these libraries built in the background and exposes these features to developer without having to build any of the functionality that they provide [4]:

- Activity manager: Manages the activity life cycle.
- Telephony manager: Provides access to telephony services as well as some subscriber information, such as phone numbers.
- View system: Handles the views and layouts that make up user interface (UI).
- Location manager: Finds out the device's geographic location.

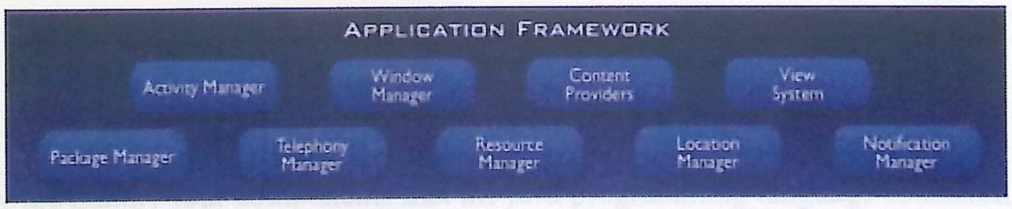


Figure 2.4: A glimpse at part of the Android application framework.

From kernel to application, the Android operating system has been developed with proven open-source technologies. This allows the developer, to build rich applications that have been fostered in the open-source community. The Figure 2.5 show the full picture of how the Android application framework stacks up.

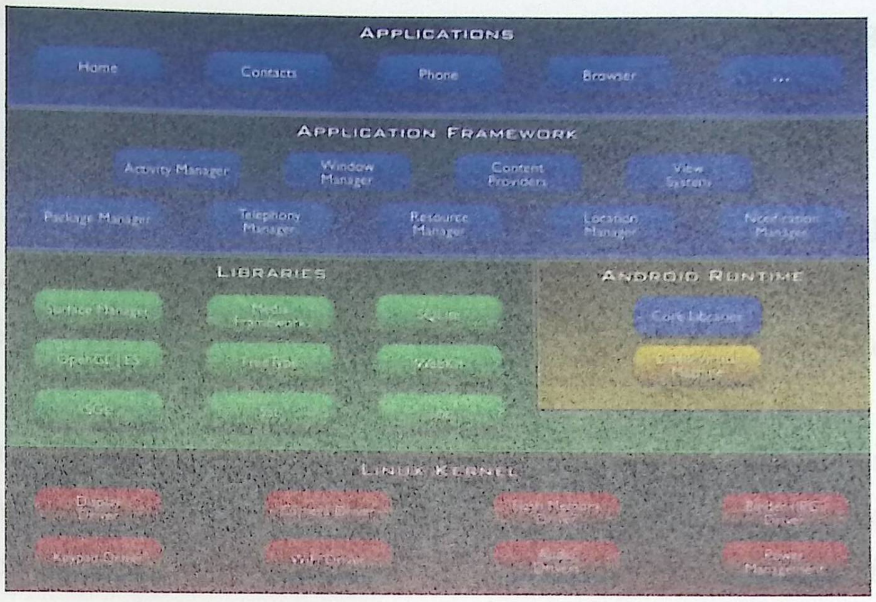


Figure 2.5: How the Android application framework stacks up.

2.5 Java knowledge

Java programming language is one of the glorious tools that make programming Android a breeze compared with programming for other mobile platforms. Whereas other languages insist that you manage memory, deal locate and allocate bytes, and then shift bits around like a game of dominoes ,Java has a little buddy called the Java Virtual Machine (JVM) that helps take care of that for you. The JVM allows to focus on writing code to solve a business problem by using a clean, understandable programming language (or to build that next really cool first-person shooter game) instead of focusing on the plumbing just to get the screens to show up.

2.5.1 Java architecture

There are mainly four components that the java programming language consistof, which are the following:

1- Java compiler.
2- Java virtual machine.
3- Java run time environment.
4- Java class libraries.

We can describe the relation between these four components in brief during program execution as follow:

- You write you source code with the programming instructions.
- When you run your code it will be compiled using the language compiler that translate it from the source to java class files.
- Run the class files on java virtual machine, where JVM is written specifically for a specific operating system.
- Java class libraries: that enables you to access system resource; by calling method in the classes that implement the Java Application Programming Interface. As your program runs, it fulfils your program's Java API calls by invoking method in class file that implement the java API [5].

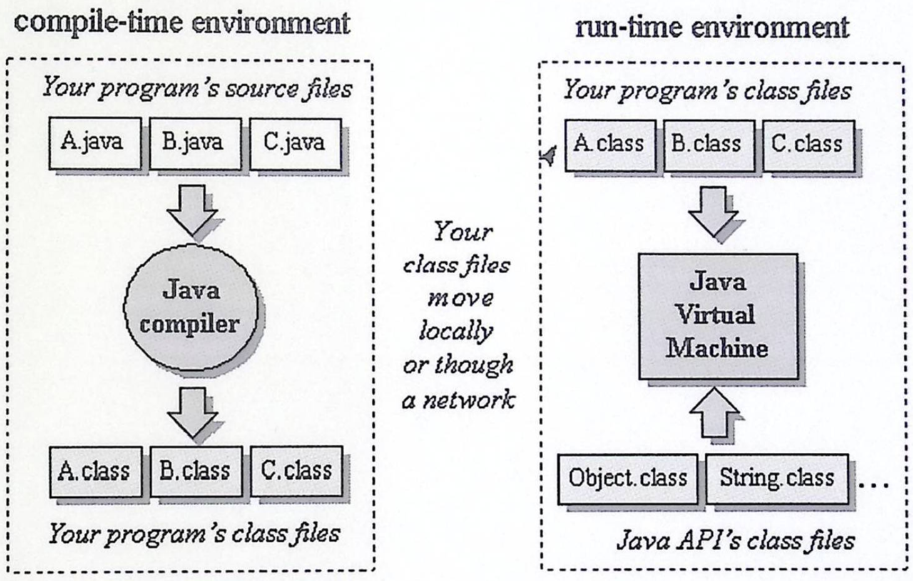


Figure 2.6:java main architecture.

2.6 Eclipse Platform

Eclipse is a multi-language software development environment comprising a workspace and an extensible plug-in system. It is written mostly in Java. It can be used to develop applications in Java and, by means of various plug-ins.
The Eclipse Platform is designed for building integrated development environments (IDEs), and arbitrary tools.

One of the key benefits of the Eclipse Platform is realized by its use as an integration point. Building a tool or application on top of Eclipse Platform, enables the tool or application to integrate with other tools and applications also written using the Eclipse Platform [7].

Chapter 3

System design

Content

3.1 Introduction

Mobile Translator application is an Android application used to capture images of text and make all the required image processing phases until getting a threshold image of characters only, then starting to extract letters and get features for these letter, the last phase is comparing these features with already existent database of letters and their corresponding features, and outputs the best matching letter and formulate the word in a text and translates it to any language .

The application is developed for Android phones on platform 2.3.3, API 10, Tess-Two library is used version 3. SQLite is used for building the required database.

This chapter will show the design of the system and discuss all parts of the project. Also, it will introduce an overall explanation for every unit used in the system. Additionally, features and detailed schematic diagrams for each component will be shown.

3.2 System Design and modeling

figure 3.1 shows the general design of the system:

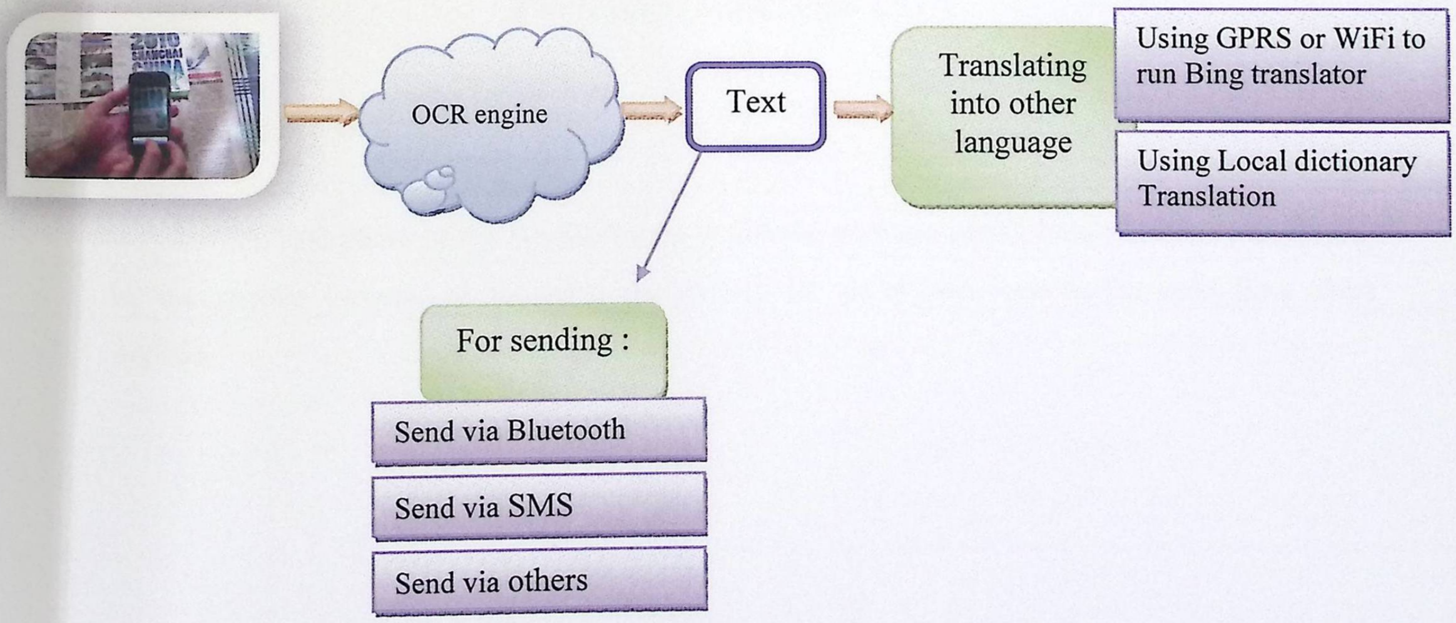

OCR engine
Text
Translating into other language
Using GPRS or WiFi to run Bing translator
Using Local dictionary Translation
For sending :
Send via Bluetooth
Send via SMS
Send via others

Figure 3.1: system design

The first phase of the project is to handle getting an image either by capturing an image by the mobile camera, or browsing the gallery for an existent image on the phone. Applying all the required pre-image processing functions from conversion to grayscale, then removing the noise with filter, convert the captured image to threshold image, then using Tess-Two library to find words and characters segmentation, then surrounding these letter with a bounding rectangle to be able to extract each letter.

The second phase is recognition then proceeds as a two-pass process. In the first pass, an attempt is made to recognize each word in turn. Then Each word passed to an adaptive classifier as training data. The adaptive classifier then gets a chance to more accurately recognize text over the page. Since the adaptive classifier may have learned something useful too late to make a contribution, a second pass is run over the page, in which words that were not recognized well enough are recognized again.

A final phase involves about getting all recognized text and prepare it for translation .

The application uses a user-friendly interface that allows the user to browse different operations done on the image in the preprocessing part, and control over the process of what to view according to user's need.

3.2.1 Over all Flow diagram of project :

Figure 3.2 shows the overall flow chart :

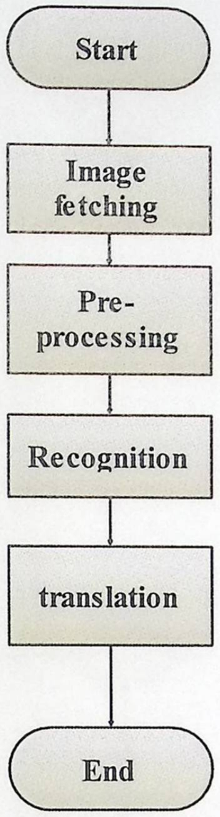


Figure 3.2 : Overall Flow Chart

3.2.2 Getting image :

It's The first phase of the application in which the user can get an image either by capturing by the mobile camera, or browsing the gallery for an existent one as the next flow chart explain.

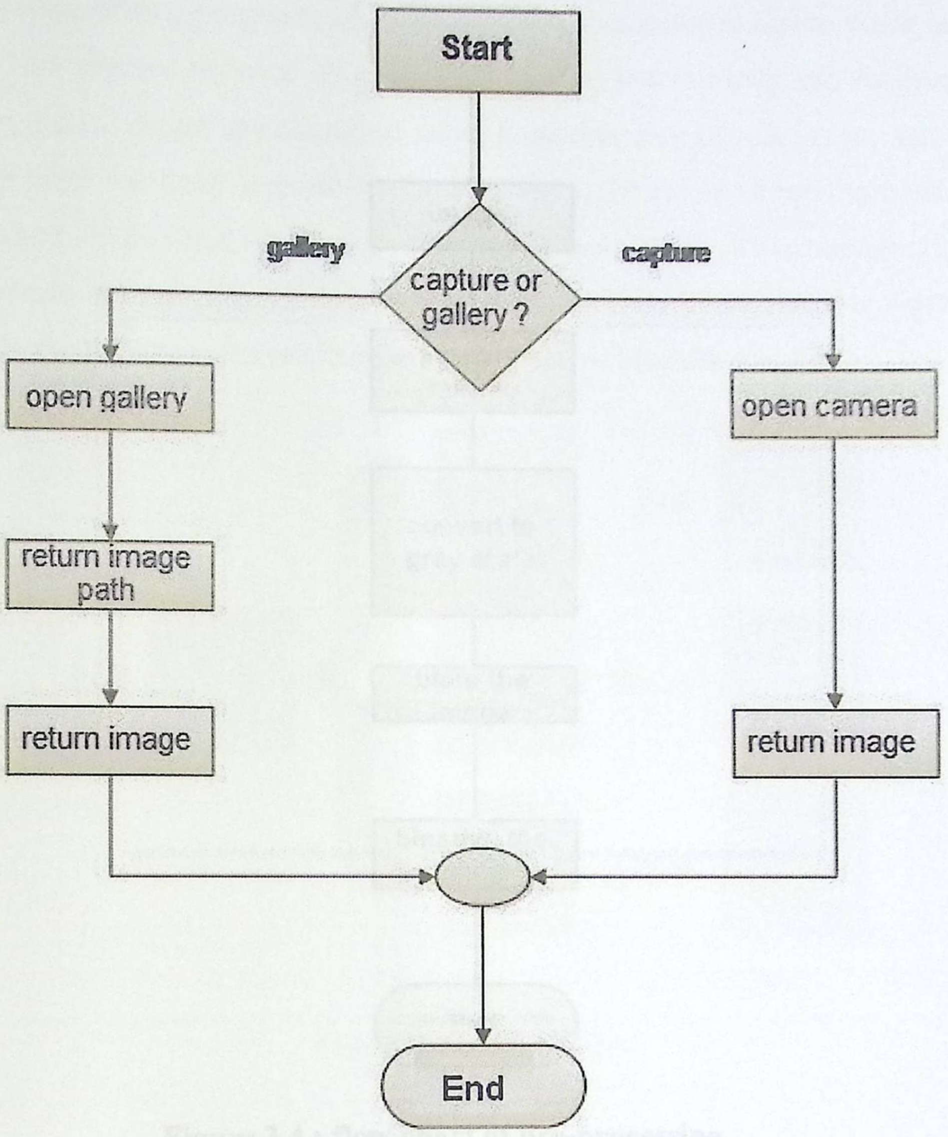


Figure 3.3: get image Flow Chart

3.2.3 Pre-Processing

The next figure shows the all required pre-image processing functions from conversion to grayscale, then removing the noise with a median filter .

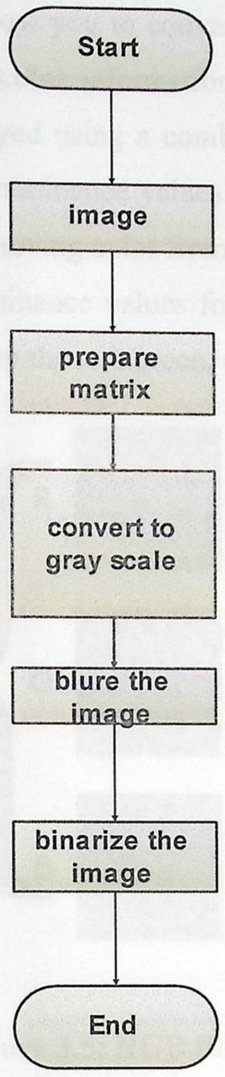


Figure 3.4 : flow chart of pre-processing

1. **Grayscale**

Grayscale is a range of shades from black to white. Therefore, a grayscale image contains only shades of gray and no color.

While digital images can be saved as grayscale (or black and white) images, even color images contain grayscale information. This is because each pixel has a luminance value, regardless of its color. Luminance can also be described as brightness or intensity, which can be measured on a scale from black (zero intensity) to white (full intensity).

Many image editing programs allow you to convert a color image to black and white, or grayscale. This process removes all color information, leaving only the luminance of each pixel. Since digital images are displayed using a combination of red, green, and blue (RGB) colors, each pixel has three separate luminance values. Therefore, these three values must be combined into a single value when removing color from an image. There are several ways to do this. One option is to average all luminance values for each pixel. Another method involves keeping only the luminance values from the red, green, or blue channel.[7]

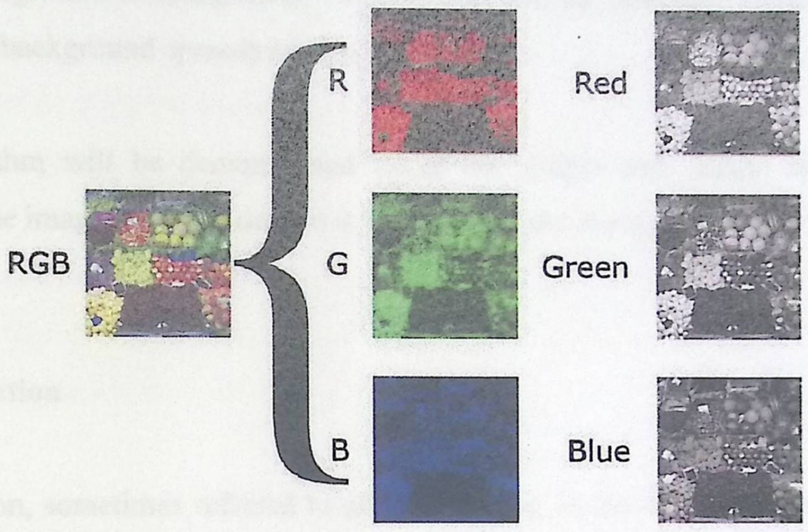


Figure 3.5: RGB filters

2. Thresholding

Document binarization is the first step in most document analysis systems. The goal of document binarization is to convert the given input grayscale or color document into a bi-level representation. This representation is particularly convenient because most of the documents that occur in practice have one color for text (e.g. black), and a different color (e.g. white) for background. These colors are typically chosen to be high-contrast so the text is easily readable by humans. Therefore, most of the document analysis systems have been developed to.

Work on binary images. The performance of subsequent steps in document analysis like page segmentation Or optical character recognition heavily depend on the result of binarization algorithm. Over the last three decades many different approaches have been proposed for binarizing a grayscale document. Color documents can either be converted to grayscale before binarization or techniques specialized for binarizing colored documents directly can be used.[8]

Otsu Thresholding:

Otsu's thresholding method involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels each side of the threshold, i.e. the pixels that either fall in foreground or background. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum.

The algorithm will be demonstrated using the simple 6x6 image shown below. The histogram for the image is shown next to it. To simplify the explanation, only 6 greyscale levels are used.

3.2.4 Segmentation

segmentation, sometimes referred to as Line finding, is the delineation of each line of text in a document or page. In computer vision, Segmentation is the process of partitioning a digital image into multiple segments. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics [19].

There are many ways to apply segmentation :

1. **Thresholding**

It's the simplest way of image segmentation (discussed in previous part) .

The task of thresholding is to separate the foreground from the background, or the ink from the paper. The grayscale histogram extracted from a scanned image typically consists of two peaks: one high and one low corresponding to the white background and black foreground respectively. The crucial task of thresholding is to find the "optimal" threshold gray-scale value between valley between the two peaks.[20]

2. **Edge Detection**

Edge detection is a fundamental tool in image processing, machine vision and computer vision, particularly in the areas of feature detection and feature extraction, which aim at identifying points in a digital image at which the image brightness changes sharply or, more formally, has discontinuities.

Region boundaries and edges are closely related, since there is often a sharp adjustment in intensity at the region boundaries. Edge detection techniques have therefore been used as the base of another segmentation technique. [21]

3.2.5 Tess-Two architecture

Tess-Two is built with various parts that other similar OCR engines have and missing some parts that other OCR engines have. Since it was originally an integral part of Hewlett Packard's commercial image recognition software, Tess-Two is missing some key components that other OCR engines have built in. For instance, most OCR solutions contain a preprocessing phase, where the page in question is laid out, de-skewed, etc. Tess-Two, on the other hand, never needed this feature because HP's proprietary (and subsequently, not open source) page layout analysis was bundled together with Tess-Two. For this reason, Tess-Two assumes that its input is a binary image (black and white) with optional polygonal text regions defined .
The following figure explains the main component of Tess-Two library [18]:

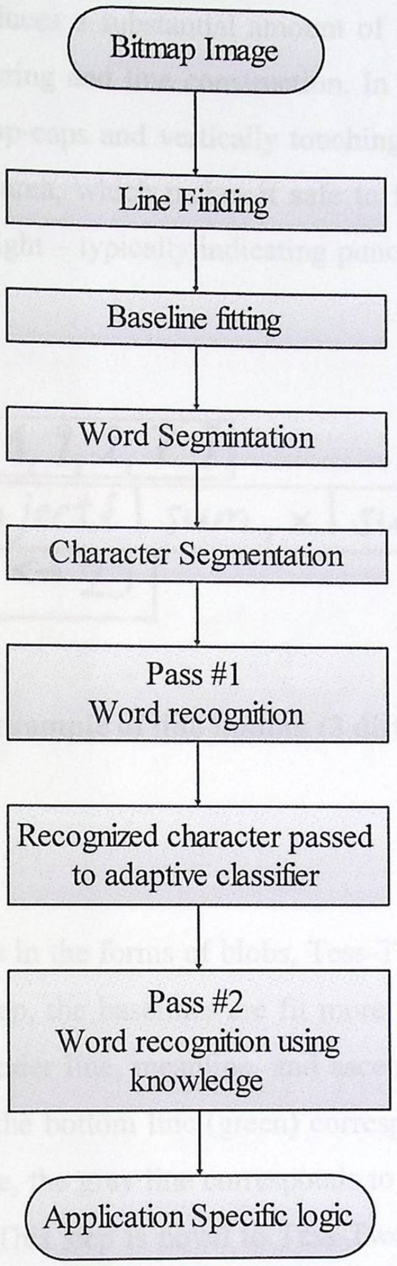
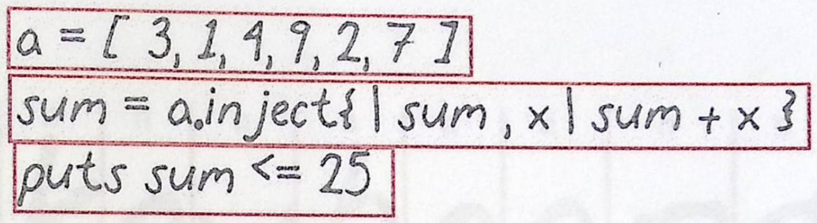


Figure 3.6: Tess-Two component

3.2.5.1 Line Finding and Word Finding

Line Finding

line finding algorithm is designed so that a skewed page can be recognized without the need to be de-skewed. Unlike Tess-Two engine, most OCR engines de-skew the page in order to ease the process line finding. This can, however, lead to a loss of image-quality as text is stretched and pulled, which introduces a substantial amount of noise. The key aspects of the line finding algorithm is blob filtering and line construction. In this step, the engine's simple percentile height filter remove drop-caps and vertically touching characters. The mean height approximates the text size in the area, which makes it safe to filter out blobs which are too small as compared to the mean height – typically indicating punctuation, diacritical marks, and noise [18].

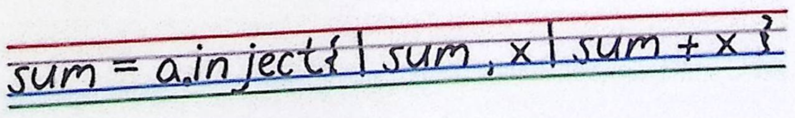


```
a = [ 3, 1, 4, 9, 2, 7 ]
sum = a.inject{ | sum , x | sum + x }
puts sum <= 25
```

Figure 3.7: An example of line finding (3 distinct lines)

Baseline Fitting

After capturing the text lines in the forms of blobs, Tess-Two engine examines the blobs on a more granular level. In this step, the baselines are fit more precisely using a four parallel lines called a fitted baseline, descender line, meanline, and ascender line. Figure 3.8 shows an example of baseline fitting, where the bottom line (green) corresponds to the ascender line, the blue line corresponds to the baseline, the gray line corresponds to the meanline, and the red line corresponds to the descender line. This step is novel to Tess-Two, which helps it handle pages with curved baselines, such as those caused by scanning artifacts and book splines [18].



```
sum = a.inject{ | sum , x | sum + x }
```

Figure 3.8: An example of baseline fitting

3.2.5.2 Fixed Pitch Detection and Chopping

Words which contain characters all with equal widths are treated as a special case in Tess-Two. Tess-Two tests text lines to figure out whether or not they are of equal width, or fixed pitch. When it finds fixed pitch text, Tess-Two splices the word equally based on the pitch, and the word is marked ready for word recognition. In the next section, we'll expand as to why this is necessary. Figure 3.9 shows and example of fixed pitch (pitch of n) text and how Tess-Two might chop it [18].

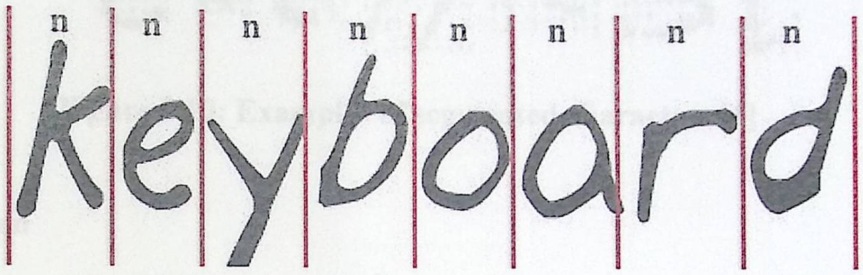


Figure 3.9: An example of fixed pitch detection, with pitch = n

Word & Character Segmentation

Line finding is often followed by a routine that separates the blobs found within a line into words. Segmenting characters relies upon the fact that there exists a uniform physical spacing between characters. For machine-printed text, this is almost always the case, and for monospaced type, this is always the case. Figure 3.10 shows how an OCR engine segments characters on a page [18].

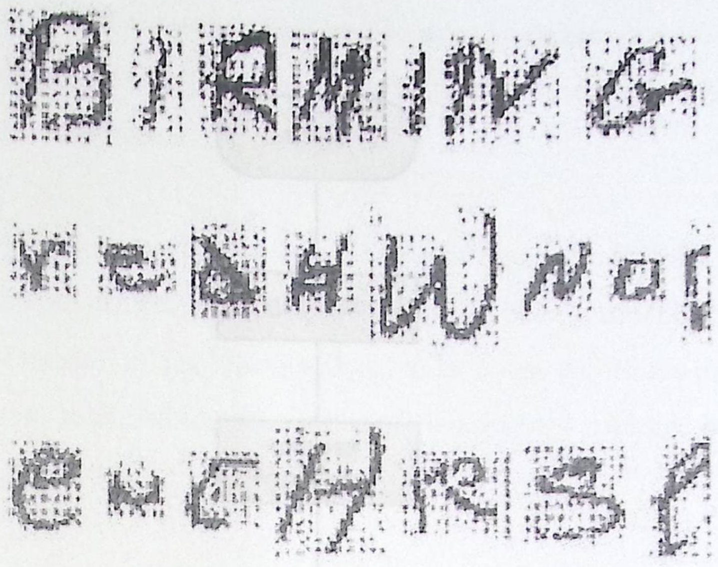


Figure 3.10: Examples of segmented characters [1]

Word Recognition

The first pass in word recognition talk about using a static classifier which had been trained on a mere 20 samples of 94 characters from 8 fonts in a single size, but with 4 attributes (normal, bold, italic, bold italic), making a total of 60160 training samples. The recognition starts by comparing each recognized character with all trained characters and calculating the mean confidence of matching and choosing the corresponding letter which have higher mean confidence [17].

The static classifer performs poorly at discriminating between different characters or between characters and non-characters or broken one. For more granular classification, a more font-sensitive adaptive classifier is trained by the output of the static classifier.

The adaptive classifer uses the same features and method as the static classifier for character recognition .

3.2.6 Translation:

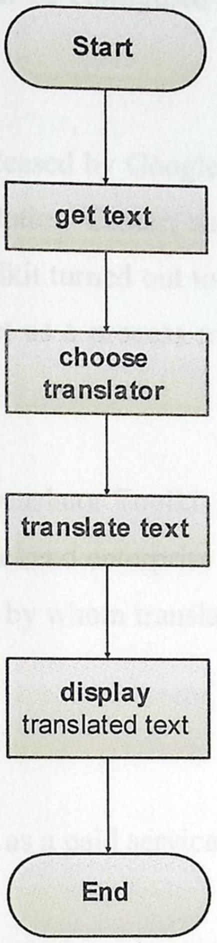


Figure 3.11: translation flow chart

Translator's trials:

Google API

Google Translator Toolkit is a web application designed to allow translators to edit the translations that Google Translate automatically generates. With the Google Translator Toolkit, translators can organize their work and use shared translations, glossaries and translation memories. They can upload and translate Microsoft Word documents, OpenOffice, RTF, HTML, text, and Wikipedia articles.

3.2.6 Translation:

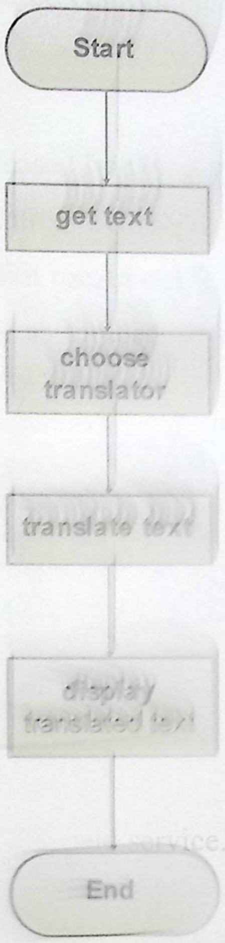


Figure 3.11: translation flow chart

Translator's trials:

Google API

Google Translator Toolkit is a [illegible]

translations that Google Translat[illegible]

translators can organize their [illegible]

memories. They can upload [illegible]

HTML, text, and Wikipedia [illegible]

Google Translator Toolkit is supported by Google Translate, a web-based translation service. Google Translator Toolkit can be configured to automatically pre-translate uploaded documents using Google Translate.

Google Translator Toolkit was released by Google Inc. on June 8, 2009 .This product was expected to be named Google Translation Center, as had been announced in August 2008. However, the Google Translation Toolkit turned out to be a less ambitious product: "document rather than project-based, intended not as a process management package but simply another personal translation memory tool".

The significance of the Google Translator Toolkit is its position as a fully online software-as-a-service that mainstreams some backend enterprise features and hitherto fringe innovations, presaging a radical change in how and by whom translation is performed.[9]

Reason to not use:

Google Translate API is available as a paid service. It need 20$ per 20,000 character.[10]

Bing

Bing Translator is a service provided by Microsoft as part of its Bing services to translate texts or entire web pages into different languages. All translation pairs are powered by Microsoft Translator technology, developed by Microsoft Research, as its backend translation software. Computer-related texts are translated by Microsoft's own syntax-based statistical machine translation technology.

In addition to standard text and web page translation, Bing Translator includes several additional :

features:

When translating an entire web page, or when the user selects "Translate this page" in Bing search results, the Bilingual Viewer will be shown, which allows users to browse the original web page text and translation in parallel, supported by synchronized highlights, scrolling, and navigation.

Four Bilingual Viewer layouts are available:

1- Side by side
2- Top and bottom
3- Original with hover translation
4- Translation with hover original

Website owners can add a translation widget to their website for translating it into other languages supported by Bing Translator; this is done by inserting an HTML code snippet on the web page

- Any-to-Any language translation pairs
- Automatically detect the language of the text or website being translated
- Ability to easily reverse the translation direction
- The user can play back a spoken version of the translation through text-to-speech (not supported in every language).[11]

1. **Translation concepts**

Bing Translate is a tool that automatically translates text from one language to another language.

The source text is the text to be translated. The source language is the language that the source text is written in. The target language is language that the source text is translated into.

2. **Languages :**

Microsoft Translator continually adds to the list of supported languages for the Translation and the following table contains some of them :

Language Code	English Name
Ar	Arabic
Bg	Bulgarian
Ca	Catalan
zh-CHS	Chinese (Simplified)
zh-CHT	Chinese (Traditional)
Cs	Czech
Da	Danish
Nl	Dutch
En	English
Et	Estonian
Fa	Persian (Farsi)
Fi	Finnish
fr	French

Table 3.1: languages [12].

3.3 Android

3.3.1 Graphical User Interface Manual

The application's interface is simple and can be easily understood by the non-technical users .

- **The application has only 4 screens**

A- Startup screen (Main) .

B- Capture screen .

C- Text showing and translation screen .

D- Image Operations menu .

A - The startup screen of the application has 4 buttons :

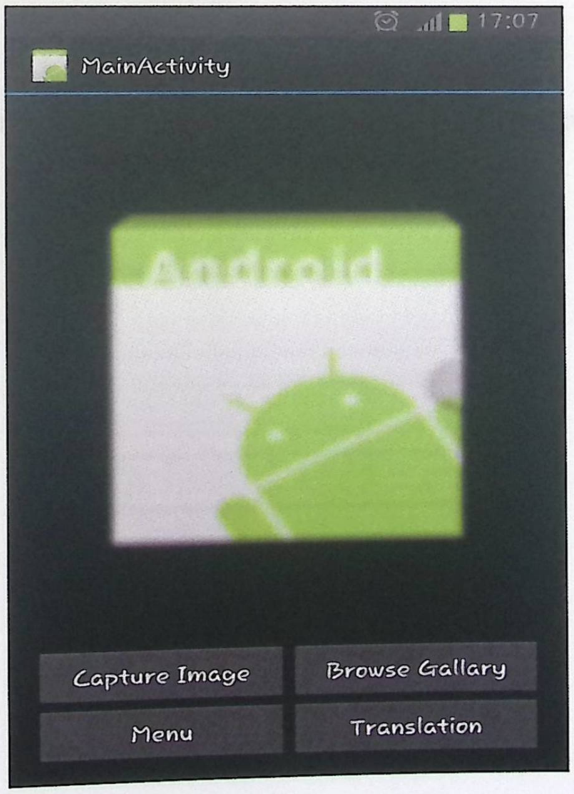


Figure 3.12 : screen shot startup screen

1- Capture image

It takes the user to the camera application where he can capture any image containing words.

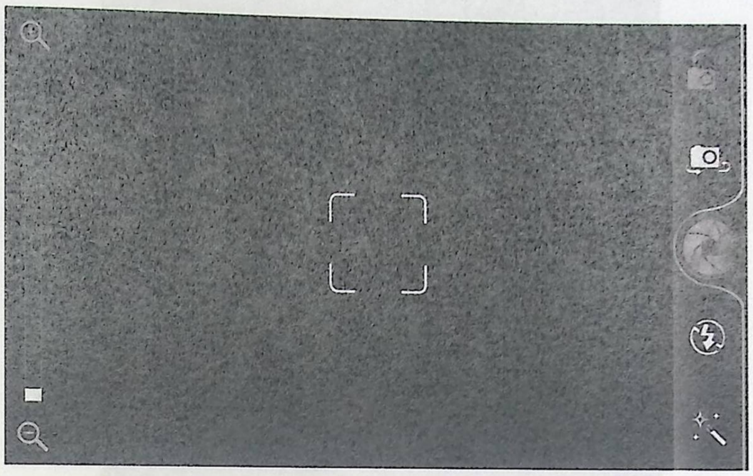


Figure 3.13 : screen shot of Capture image layout

2- Browse Gallery

Redirects the user to the gallery to select a pre-captured or any saved image

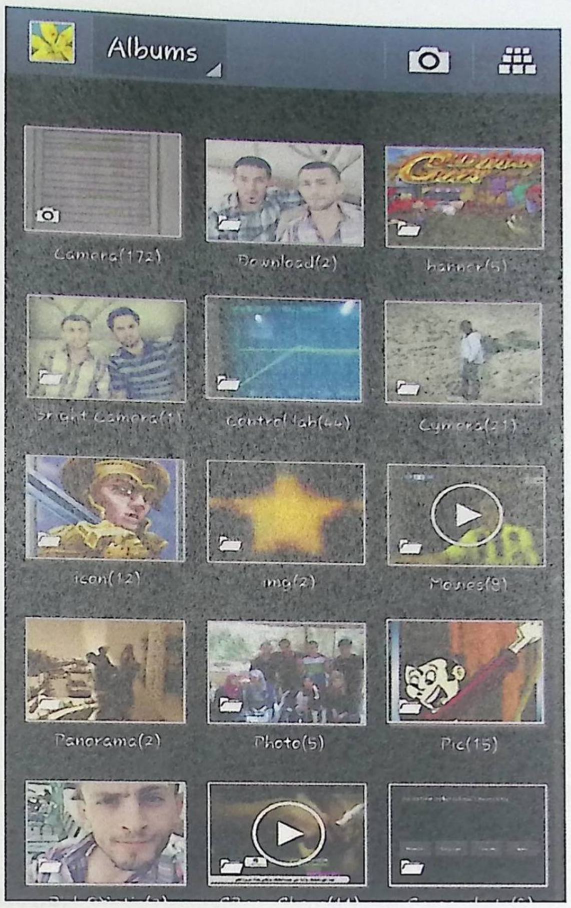


Figure 3.14 : Screen shot of gallery

Once the users clicks the image it is returned to the Startup activity

B- Text showing and translation screen .

After capturing the image the translation screen appears with the recognized text

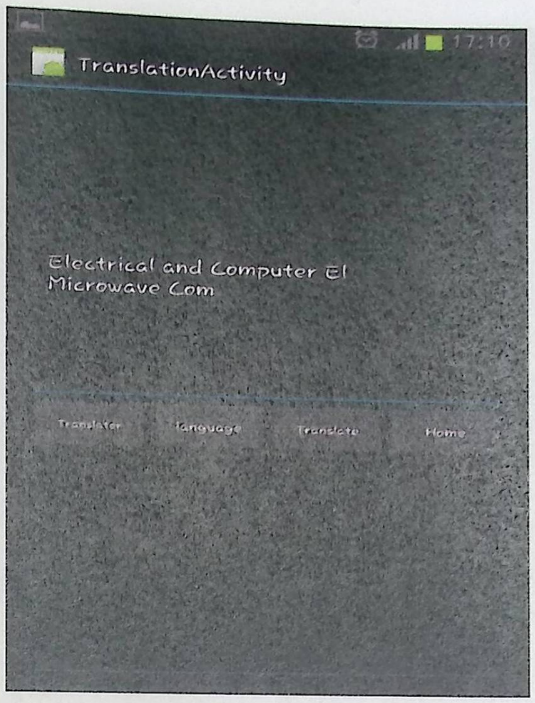


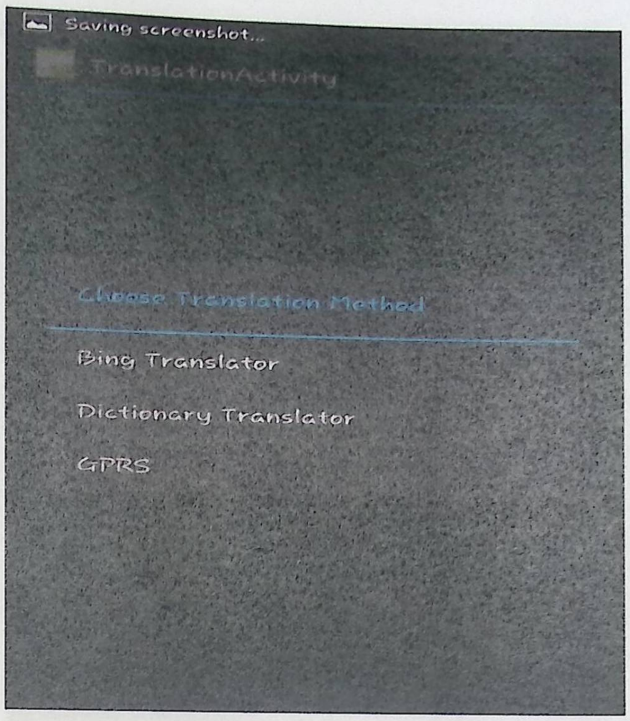
Figure 3.15 : Screen shot after capturing the image

The user has four options:

- **Home :** Takes the current image and returns it to the startup activity .

- **Translator** : Enables the user to choose type of translator he want to use .The default translator has been assigned to be Bing translator by which The user can translate the shown text to the languages supported by the Bing API jar file and the user can browse the language through the menu.

after clicking on Translator button the following selection dialog appears so the user can choose any translator he want to use :


Saving screenshot...

TranslationActivity

Choose Translation Method

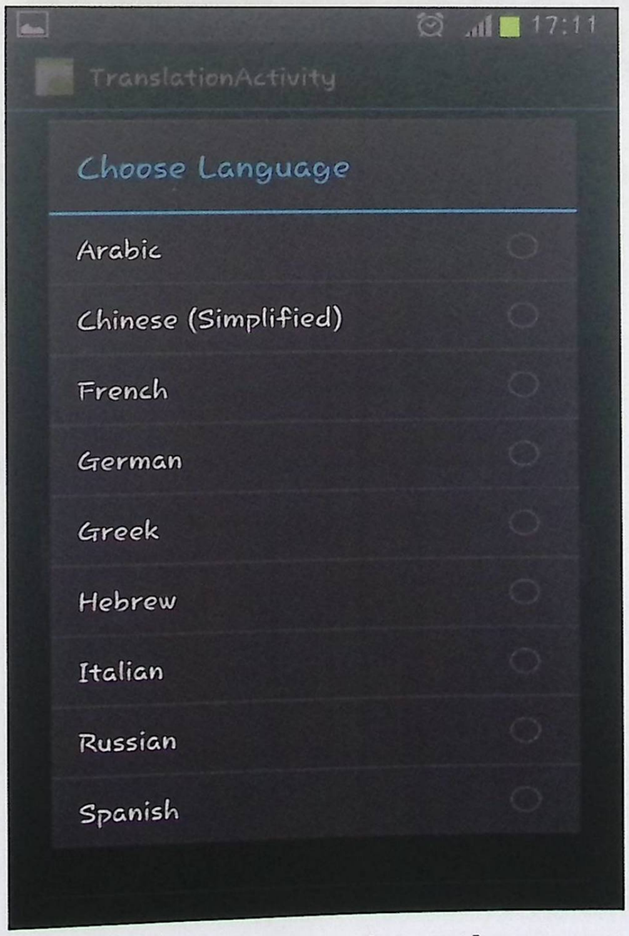
Bing Translator

Dictionary Translator

GPRS

Figure 3.16 : Screen shot of available translator

- **Language :** shows a selection dialog of different target languages to translate into .


17:11

TranslationActivity

Choose Language

Arabic

Chinese (Simplified)

French

German

Greek

Hebrew

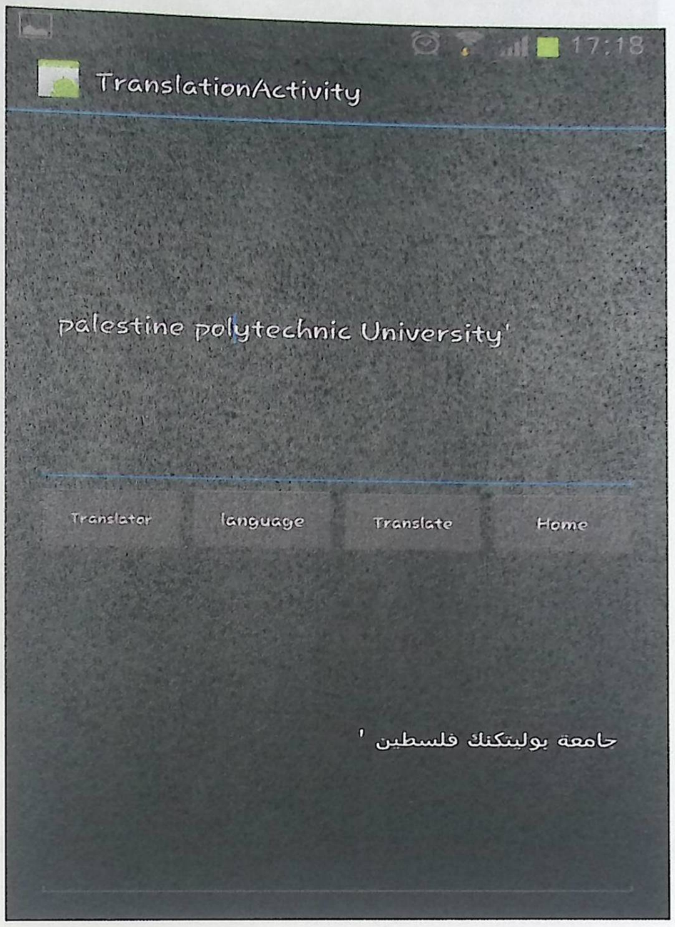
Italian

Russian

Spanish

Figure 3.17 screen shot of target languages

- **Translate :** which take the text and provide its translation .



palestine polytechnic University'

جامعة بوليتكنك فلسطين '

Figure 3.18 screen shot of text after translation

C- Image Operations menu :

It takes the user to the second screen of the activity, in this screen the user can select from three options, each of them returns the image after the operation selected to the main screen

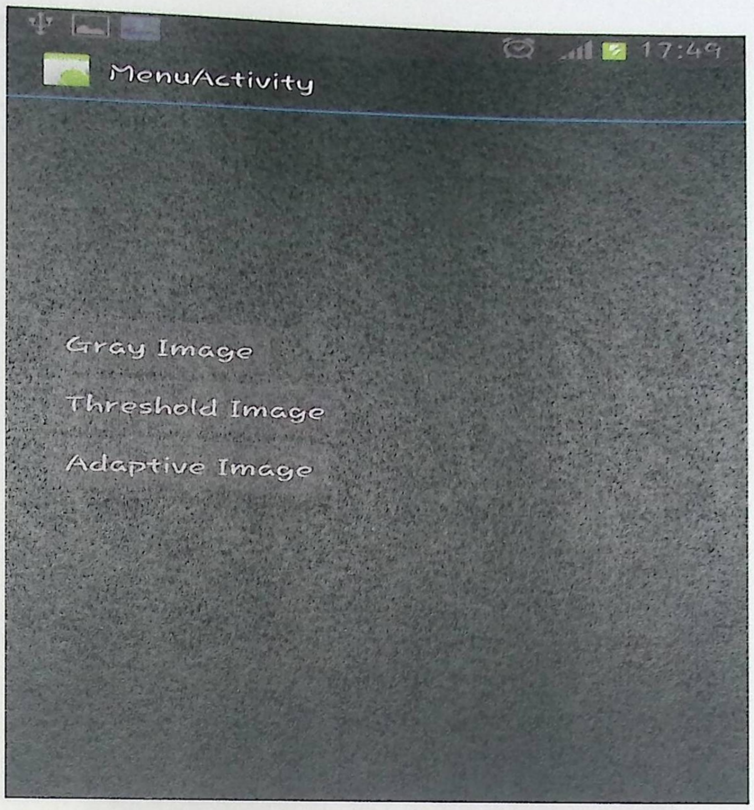


Figure 3.19: screen shot of menu layout

- **Gray Scale :** It Converts the colored image to a gray image and displays it .

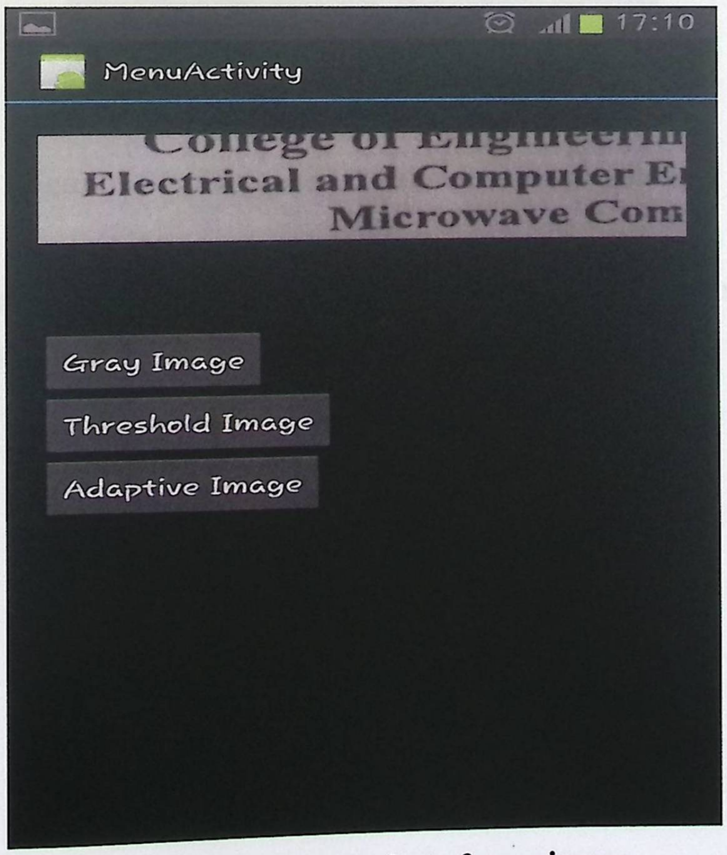


Figure 3.20 : screen shot of gray image

- **Threshold Image :** It converts the colored image to a binarized image of only 2 values for a pixel (0 or 1) and displays it

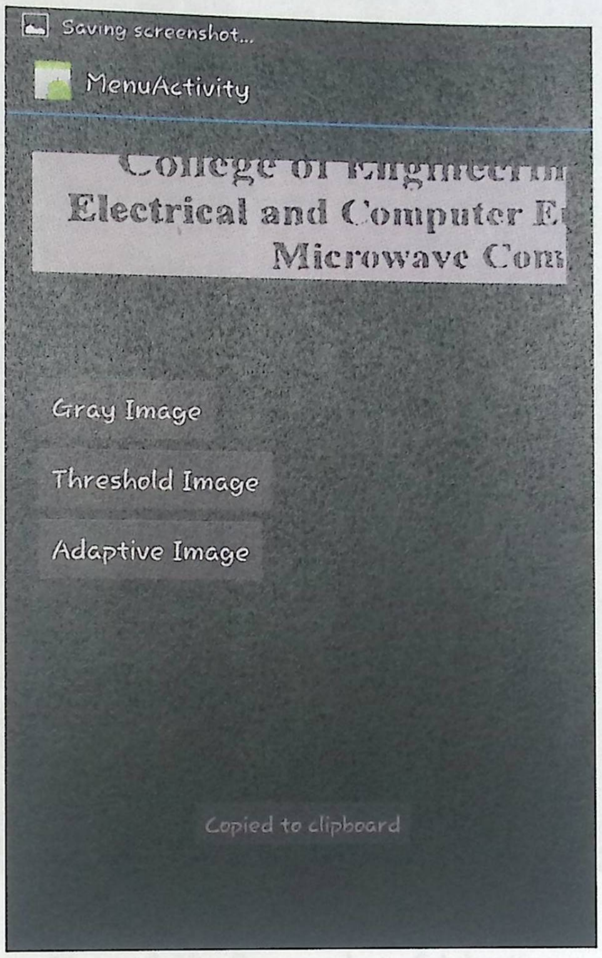


Figure 3.21: Screen shot of Black and white image

- **Adaptive Image :** It shows grayed image after blurring; image enhancement and distortion reduction .

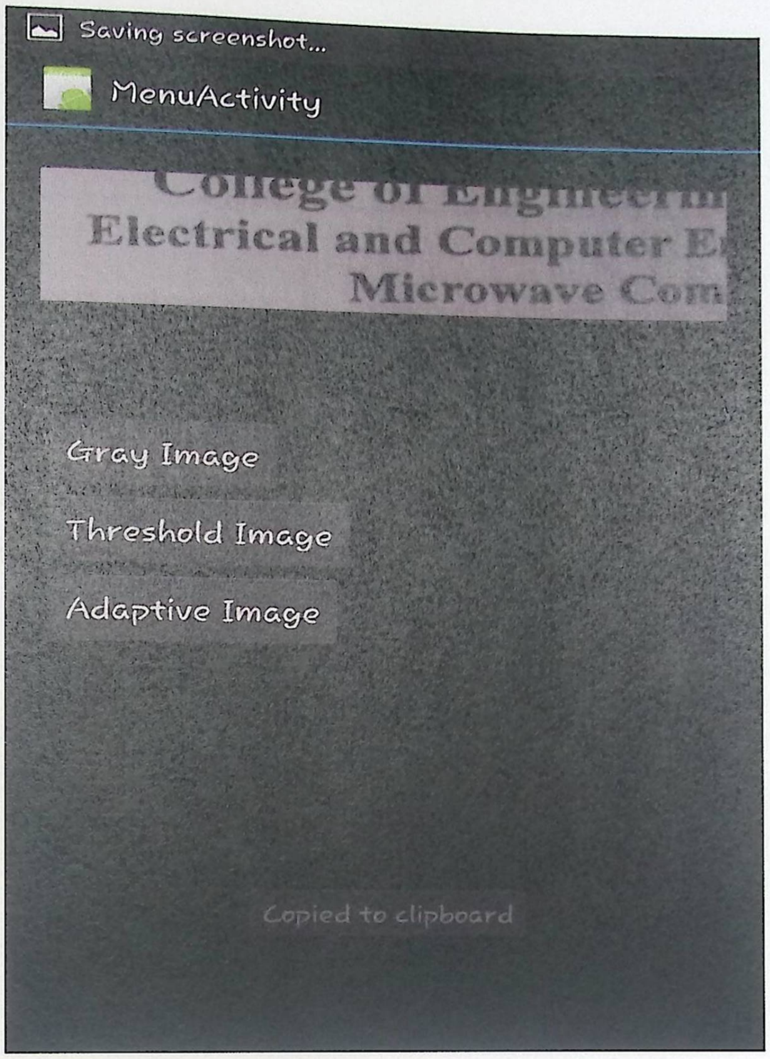


Figure 3.22 : Screen shot of Adaptive

Chapter 4

System Implementation

Content

4.1 Introduction :

4.2 Project Layout and Interface Design:

The Layout and Interface Design consist of main parameters and objects that must be handled. In the next section we are going to explain these important parameters.

4.2.1 An Activity:

An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window, in which the developer can place his/her UI with setContentView (View). While activities are often present to the user as full-screen window.

The Activity class is an important part of an application overall cycle, and the way activities are launched and put together is a fundamental part of the platform's application model.

4.2.2 Activity Life Cycle:

Activities in the system are managed as an *activity stack*. When a new activity is started, it is placed on the top of the stack and becomes the running activity, the previous activity always remains below it in the stack, and will not come to the foreground again until the new activity exits.

An activity has essentially four states:

- If an activity in the foreground of the screen (at the top of the stack), it is *active* or *running*.
- If an activity has lost focus but is still visible, it is *paused*. A paused activity is completely alive (it maintains all state and member information and remains attached to the window manager), but can be killed by the system in extreme low memory situations.

- If an activity is completely obscured by another activity, it is *stopped*. It still retains all state and member information, however, it is no longer visible to the user so its window is hidden and it will often be killed by the system when memory is needed elsewhere.
- If an activity is paused or stopped, the system can drop the activity from memory by either asking it to finish, or simply killing its process. When it is displayed again to the user, it must be completely restarted and restored to its previous state. [13]

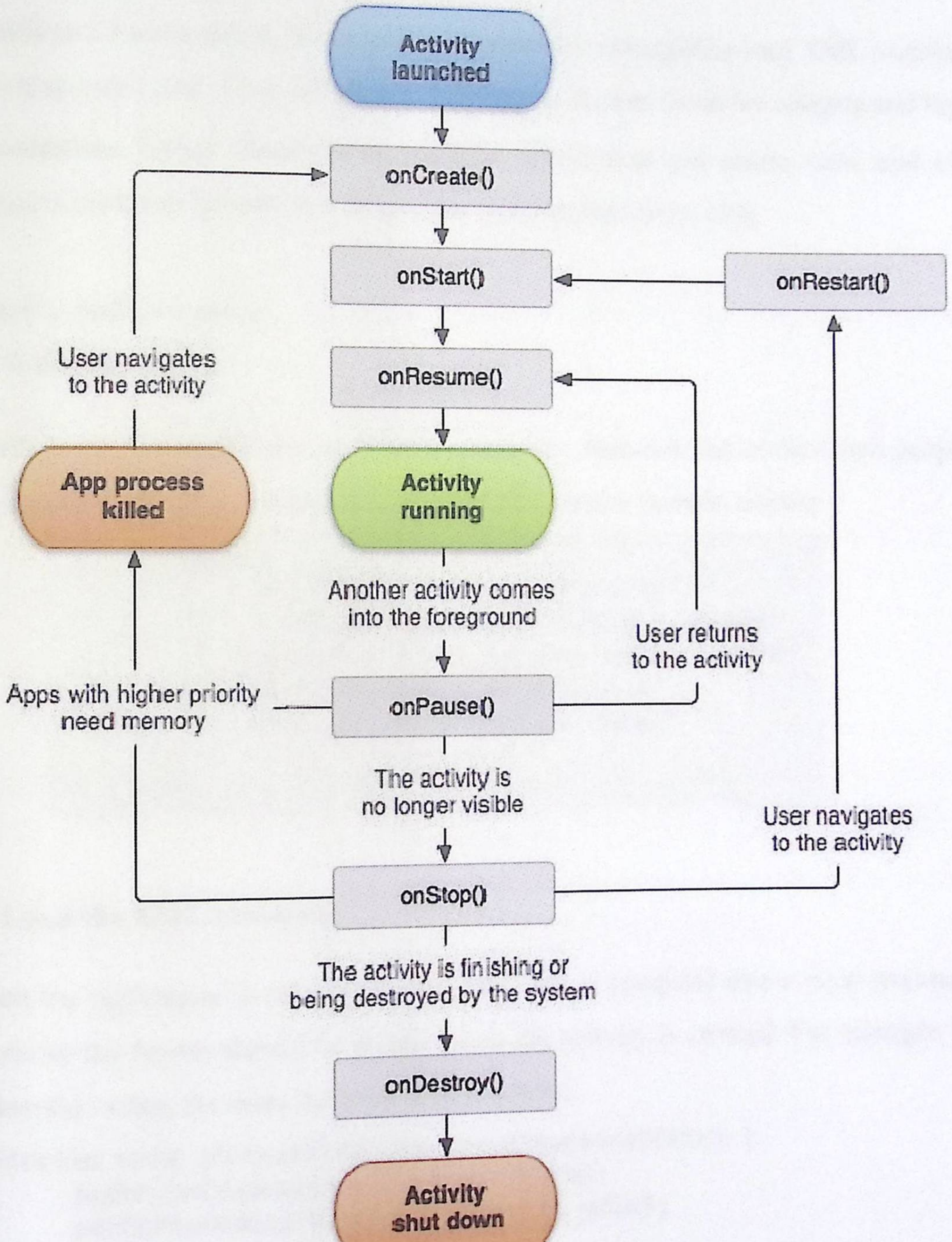

Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Figure 4.1 : Flow chart of activity life cycle[13]

4.2.3 Layouts:

Layout is the architecture for the user interface in an activity. It defines the layout structure and holds all the elements that appear to the user (Like: Buttons, ImageView, TextView,Menus,…….)

Layout contents can be declared in two ways:

- **Declare UI elements in XML**. Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses, such as those for widgets and layouts.
- **Instantiate layout elements at runtime**. application can create view and viewgroup objects (and manipulate their properties) programmatically. [14]

So the Steps to make a Layout:

1. **Write the XML.**

In which we should declare our objects (example : Buttons) and declare their properties.

For example, in order to add a button to run the camera in main activity :

```
<Button
        android:id="@+id/CaptureBtn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="0.5"
        android:text="Capture Image"

/>
```

2. **Load the XML resources.**

When the application is compiled each XML file is compiled into a view resource and for every activity the layout should be loaded when the activity is created. For example The main layout launches when the main Activity is first created

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        }
```

3. **Attributes.**

Every View and ViewGroup object supports its own variety of XML attributes. Some of the Attributes are of the seen objects (ex: Buttons and TextViews….) And some other attributes are Layout parameters which are attributes that describe certain layout orientations of the View object, as defined by that object's parent ViewGroup object.

And every Object should have its distinct ID to uniquely identify it when using it or when handling its events .

For example The ID for capture button is declared in the XML file as:

```
android:id="@+id/CaptureBtn"
```

and for browse gallery :

```
android:id="@+id/BrowseGalleryBtn"
```

and then after identifying each object ID we can reference and fetching it from the layout to handle it's events. for example :

```
Button CaptureImageBtn = (Button) findViewById(R.id.CaptureBtn);
```

Then another important Issue in the layout design must be handled which is adjusting the size and position of every object with respect to each other so the next step counts as a big deal in the layout handling .

4. **Layout Parameters:**

Every ViewGroup class implements a nested class that extends ViewGroup.LayoutParams. This subclass contains property types that define the size and position for each child view, as appropriate for the view group. As you can see in figure 4.2, the parent view group defines layout parameters for each child view (including the child view group).

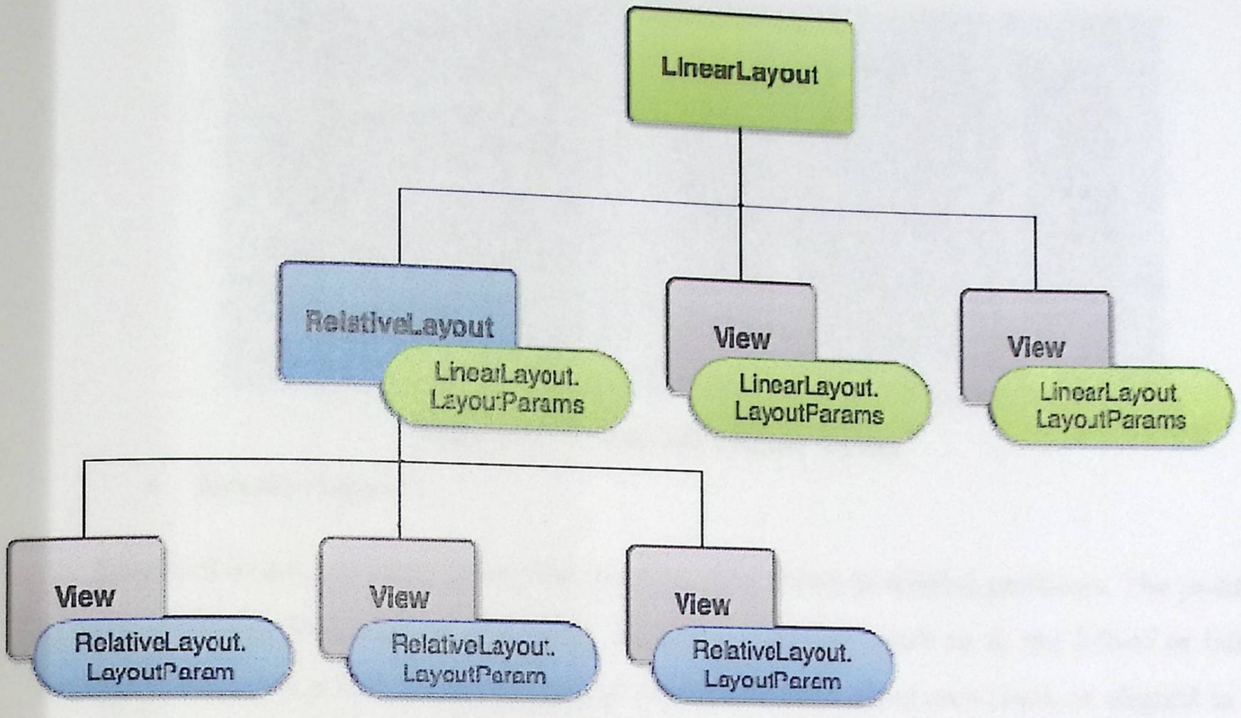


Figure 4.2 : Layout parameters classification

Every LayoutParams subclass has its own syntax for setting values. Each child element must define LayoutParams that are appropriate for its parent, though it may also define different LayoutParams for its own children.

you will use one of these constants to set the width or height:

- wrap_content tells your view to size itself to the dimensions required by its content .
- fill_parent (renamed match_parent in API Level 8) tells your view to become as big as its parent view group will allow.

And these are examples for the common layouts used in holding the objects:

- **Linear Layout:**

LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally. You can specify the layout direction with the android:orientation attribute.

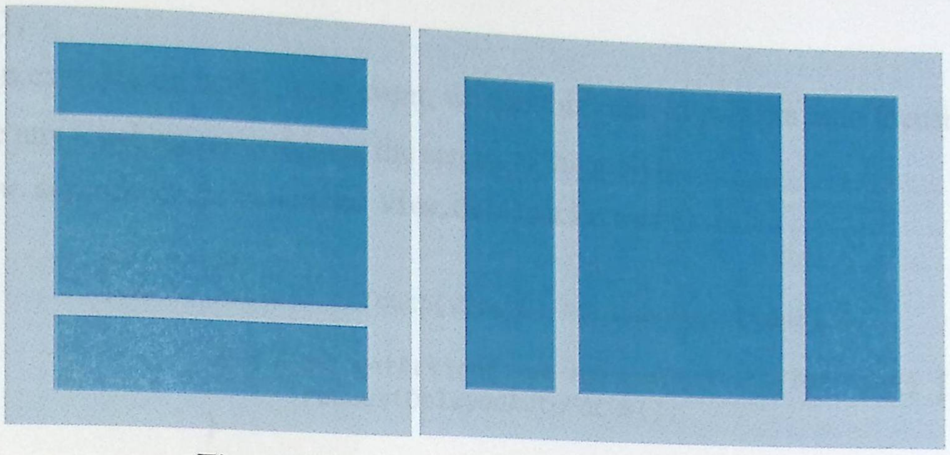


Figure 4.3 : Android's linear layout

- **Relative Layout :**

RelativeLayout is a view group that displays child views in relative positions. The position of each view can be specified as relative to sibling elements (such as to the left-of or below another view) or in positions relative to the parent RelativeLayout area (such as aligned to the bottom, left of center).

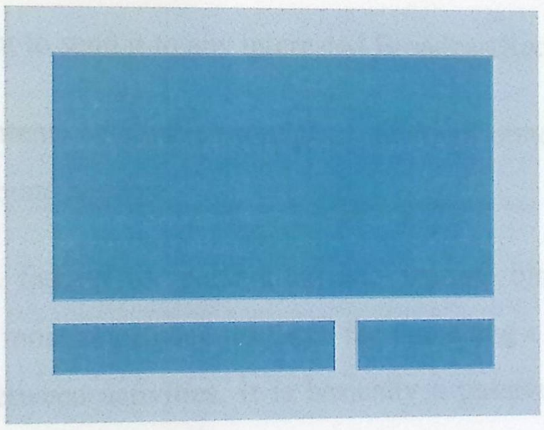


Figure 4.4 : Android's relative layout

5. Handling Click and Touch Events:

In Order to use every object added in our layout we must implement the methods we need to do and events we need to launch when this object is pressed or touched, for example we need to activate Capture Image button when it pressed :

```
OnClickListener CaptureImageBtn = new OnClickListener() {

        @Override
        public void onClick(View v) {

                // Do something in response to the click
                // here can initiate the camera
        }
        };
```

And as an example on using touch event, we use this code to perform auto focusing when the user make any touching movement at the screen of the mobile :

```
viewfinderView.setOnTouchListener(new View.OnTouchListener() {

                    @Override
                    public boolean onTouch(View v, MotionEvent event) {

                            if (event.getAction() == MotionEvent.ACTION_MOVE) {
                                    requestDelayedAutoFocus();
                            }

                            return true;
                    }

        });
```

6. The Intent :

Intent is an abstract description of an operation to be performed. It can be used with:

- startActivity to launch an Activity,
- broadcastIntent to send it to any interested BroadcastReceiver components,
- startService(Intent) or bindService(Intent, ServiceConnection, int) to communicate with a background Service.

An intent provides a facility for performing late runtime binding between the code in different applications. Its most significant use is in the launching of activities, where it can be thought of as the glue between activities. It is basically a passive data structure holding an abstract description of an action to be performed.

And the important Intent type used in this project is :

- `public Bundle getExtras ()`

Returns The map of all extras previously added with putExtra(), or null if none have been added.

7. Android Manifest.XML file:

Every application must have an AndroidManifest.xml file in its root directory. The manifest presents essential information about the application to the Android system, information the system must have before it can run any of the application's code.

The main functions of the manifest file is:

- It names the Java package for the application. The package name serves as a unique identifier for the application.
- It describes the components of the application — the activities, services, broadcast receivers, and content providers that the application is composed of.
- It determines which processes will host application components.
- It declares which permissions the application must have in order to access protected parts of the API and interact with other applications.
- It also declares the permissions that others are required to have in order to interact with the application's components.
- It declares the minimum level of the Android API that the application requires.
- It lists the libraries that the application must be linked against.

Now we will discuss the main operations of every Activity and The life cycle of every Activity and how's most of the activities handles the operations between Them:

The Layout and Interface Design Consists of :

- Four Activities: Main Activity, Menu Activity, Translation Activity, and Capturing Activity .
- Four Layouts: activity_main.xml, activity_menu.xml, activity_translation.xml, and capture.xml.
- Classes: OcrProcessing.java and DBAdapter.java

4.2.4 Main Activity :

This Activity appears firstly after launching the Project. And it starts other Activities by pressing on the buttons on its layout it triggers the Oncreate( ) of other Activities, and also it organizes the passing of the parameters between the classes in the project.

The Operations Handled by this Activity:

This Activity shows the main Layout (activity_main.xml) which has four Buttons:

1. Capture Image Button.
2. Browse Gallery Button.
3. Translation Button.
4. Menu Button.

- **Triggering the Camera and Gallery Intents:**

When pressing The Camera or The Gallery Buttons, Their OnclickListener() for both Intents are triggered which opens the camera or the gallery Activities That is already embedded in the Android software:

```
OnClickListener BrowseGallery = new OnClickListener() {

        @Override
        public void onClick(View v) {
    // create intent
  Intent intent = new Intent();
  intent.setType("image/*");
  intent.setAction(Intent.ACTION_GET_CONTENT);
  //return the image
  startActivityForResult(Intent.createChooser(intent, "Select Picture"),1);
         }
        };
```

And when the on click listener for the Camera is applied the capture activity is then called to start a specific camera class adapted from Zxing project for Barcode detection and translation, and we will get handle this in the next parts.

And after the camera or the Gallery function is opened and the image is captured or browsed the result of the Intent is then passes to the used library Tess-Two after its decoded into a bitmap image using BitmapFactory.decodestream() .

4.2.5 Capture Activity

By pressing the Capture Image Button on the main Layout in the Main Activity The *OnClickListener( )* function is implemented which open the internal built in camera but using this camera will return a high resolution and very detailed image which spend more time and need high memory size during the Ocr processing and that’s sometime leads the project to be crashed, so we searched so long about solution can solve this issue and finally we tend to use less resolution and detailed image .

we found an open source android project called Zxing for barcode recognition which uses its own designed camera with 480*360 dimensions, and we can obtain any dimensions suit with our needs so we chose 600*400 dimensions at the beginning Knowing that we can choose different dimensions.

The process of extracting the specific code that draw the camera layout from an existence project is hard and requires lot of time and effort because we just need the part that handle the camera and not anything else. So we firstly invoke all Zxing project and begun deleting for all code not necessary until we finally get the desired code which containing eight classes and main java class to be invoked which in turn will call the layout of the camera activity which is capture.xml

Now we can resize camera resolution as we need by modifying the operators defined in cameramanegar.java class which wraps the camera service object and expects to be the only one talking to it. The implementation encapsulates the steps needed to take preview-sized images, which are used for both preview and decoding .

```
private static final int MAX_FRAME_WIDTH = 600; // originally 480
private static final int MAX_FRAME_HEIGHT = 400; // originally 360
```

The layout capture.xml contains a Framelayout as a container. we can be defined within XML layout resources or programmatically in the application's Java code. A child view within a frame layout is always drawn relative to the top left-hand corner of the screen. There are multiple child views exist, then they are drawn, in order, one atop the other. This means that the first view added to the frame layout will display on the bottom of the stack, and the last view added will display on top.

We have added three child controls to this frame layout:

- A SurfaceView we have named it preview_view and it Provides a dedicated drawing surface embedded inside of a view hierarchy.
- A ViewfinderView overlaid on top of the camera preview to add the viewfinder rectangle and partial transparency outside it, as well as the laser scanner animation and result points, we defined it in the xml file as follow:

```
<com.example.translation.ViewfinderView
     android:id="@+id/viewfinder_view"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:background="#0000"/>
```

- A RelativeLayout contained a text view operates as shutter button when the user touch it the captured image then passed to ocr directly.

When the user click on capture button in main activity the Oncreate( ) cycle of the capture activity will begin execution and here the parameter required by the camera will be initialized and defined as follow :

```
      //defining the surface that would hold the camera
viewfinderView = (ViewfinderView) findViewById(R.id.viewfinder_view);
cameraManager = new CameraManager(getApplication());
viewfinderView.setCameraManager(cameraManager);
```

after that the process calls the OnResum( ) cycle which in turn allow the surfaceview to appear and then set up the camera preview surface and start previewing as follow :

```
// Set up the camera preview surface.
          surfaceView = (SurfaceView) findViewById(R.id.preview_view);
          surfaceHolder = surfaceView.getHolder();
          if (!hasSurface) {
               surfaceHolder.addCallback(this);
               surfaceHolder.~~setType~~(SurfaceHolder.~~SURFACE_TYPE_PUSH_BUFFERS~~);
          }
// if the surface has been created then open camera
          if (hasSurface) {

               initCamera(surfaceHolder);
          }

/** Initializes the camera and starts the handler to begin previewing. */
     private void initCamera(SurfaceHolder surfaceHolder) {

          if (surfaceHolder == null) {
               throw new IllegalStateException("No SurfaceHolder provided");
          }
          try {

               // Open and initialize the camera
               cameraManager.openDriver(surfaceHolder);

               // Creating the handler starts the preview, which can also throw
               // RuntimeException.
               handler = new CaptureActivityHandler(this, cameraManager);

          } catch (IOException ioe) {
                Toast.makeText(this," Could not initialize camera. Please try
restarting device", Toast.LENGTH_LONG).show();
            } catch (RuntimeException e) {
                Toast.makeText(this," Could not initialize camera. Please try
restarting device", Toast.LENGTH_LONG).show();
            }
            }
```

In this stage the user can focus the camera on top of any text he want to detect and if any success detection happened the onPause( ) cycle will invoke to stop previewing in order to prevent conflicting with other camera based as follow :

```
@Override
    protected void onPause() {
        if (handler != null) {
            handler.quitSynchronously();
        }

        // Stop using the camera, to avoid conflicting with other camera-based

        cameraManager.closeDriver();
        super.onPause();
        }
```

As we said in this stage the user can capture any image contained a characters and directly it passed to image processing class which make different ocr processing using Tess-Two library ,and here the user can deal with two situation either the ocr provided no results so a toast with ocr failure message will appear and then capturing will be resumed, or if any result returned then capturing will be paused and all results passed to translation activity :

```
// Checking for failure to recognize text
 if (ResultText == null || ResultText.equals("")) {

Toast toast = Toast.makeText(this, "OCR failed. Please try again.",
Toast.LENGTH_SHORT);
    toast.setGravity(Gravity.TOP, 0, 0);
     toast.show();

 }

//Sending Results to Translation Activity
    if(ocrResult.getText() != null || ocrResult.getText() != "")
    {
    Intent ToTranslationActivity = new Intent(this , TranslationActivity.class);
    ToTranslationActivity.putExtra("text",ocrResult.getText() );
    ToTranslationActivity.putExtra("image_send",ocrResult.getBitmap());//send img
to
ToTranslationActivity.putExtra("image_send2",ocrResult.getBitmap2());//send
thresholded img to
ToTranslationActivity.putExtra("image_send3",ocrResult.getBitmap3());//send adaptive
img to
startActivity(ToTranslationActivity);
        }
```

4.2.6 Menu Activity :

when pressing the Menu Button The Menu Activity starts that displays 3 options of the image which are GrayScale image, Thresholded image, and adaptive image. The user can pick to choose the processed image to display so when pressing the menu button the Onclick Listener for the menu is performed and by creating an Intent it starts the activity of the Menu

```
OnClickListener StartMenuActivity = new OnClickListener() {

        @Override
        public void onClick(View v) {

                Intent MenuActivity = new Intent(this,MenuActivity.class);
                MenuActivity.putExtra("GryImg", lastBitmap);
                MenuActivity.putExtra("ThresholdedImg", lastBitmap2);
                MenuActivity.putExtra("AdabtiveImage", lastBitmap3);
                startActivity(MenuActivity);

        }
};
```

And on the OnActivity result we return the option chosen by the user so as to send this Option to the Image Processing Class to return to the user the Image required to be displayed, the following code for example shows how we get the gray image from intent :

```
OnClickListener GrayImageReturned = new OnClickListener() {
        @Override
        public void onClick(View v) {
         lastBitmap = getIntent().getParcelableExtra("GryImg");//received
GrayImage from Main Activity
                setimage(lastBitmap);
        }};
```

4.2.7 Translation Menu Activity:

This activity is started when the ocr library succeed to extract any information from captured image, it displays the options of the translator type that the user can use and the languages that he can translate to, and returns back the result so that it is sent to the desired translator to translate the text to the desired language.

This activity displays the activity_translation.xml layout which has four buttons:

- Translator button to display list of three translator types we have implemented in this project , the default translator was assigned to be Bing translator that need an internet connection to make translation, additionally there are two additional translator one of them doesn't need internet connection and it is immediate dictionary containing only english words, the other one it is not a translator but it is a connection method use to make translation without needing internet connection but mobile network.
- Language button displaying the most common languages to translation to.
- Home button to return to the main activity layout.
- Translate button to perform translation using any type of provided translators.

Bing Translator :

Bing Translate is a tool that automatically translates text from one language to another. The service can be used in web or client applications to perform language translation, and other language-translated operations and supports users who are not familiar with the default language of a page or application, or those desiring to communicate with people of a different language group.

The Microsoft Translator API is available through the Windows Azure Marketplace which offer a free service usage limit of 2 million characters per month as well as paid monthly subscriptions for higher volumes.
In order to get our Bing API we must follow this guide :

1. Creating an Azure marketplace account as follow :
   - Logging to http://datamarket.azure.com/ using a Microsoft live account.
   - selecting the free 2,000,000 character/month subscription package.
2. Creating an Azure application as follow :
   - Head on over to the application registration page. Once there, we fill out all the necessary information and copy the Client secret and Client ID. We'll need these later

Using the client key and secret from above, we have all we need to call the API. We are going to implementing a client to call the API, so we will need to first authenticate in order to get an access token, then use that in the header of our requests for authorization.

The source text is the text to be translated. The source language is the language that the source text is written in. The target language is language that the source text is translated into.

before realizing there was a really great client for the translator written already called Microsoft-translator-java-api to Provides a Java wrapper around the Bing Translator. So we must install this library in the libs folder of the project classpath .Here is sample java code to invoke Bing translator using this library which takes care of authentication and translation requests:

```
    //setting the API key and secret
      Translate.setClientId("MicrosoftTranslatorJavaAPI");
      Translate.setClientSecret("0VHbhXQnJrZ7OwVqcoX/PDZlyLJS9co3cVev1TPr8iM=");
                 try {
//setting the Language this is done through "Language. ARABIC"
//sending text to be translated  this is done through function "execute"

translatedText = Translate.execute(RecTextView.getText()
            .toString(), Language.ENGLISH, TargetLanguage);
      } catch (Exception e) {
            translatedText = e.toString();
      }
```

Local Dictionary:

Also user can use a built-in dictionary to get translation if there is no available internet connection, so we will show how this dictionary is built as a database to give a meaning of a word and if it not exist it allows us to add it with its meaning.

That was achieved via the help of SQLite .SQLite is an Open Source Database which is embedded into Android which supports standard relational database features like SQL syntax.

SQLite in Android

SQLite is available on every Android device. Using SQLite database in Android does not require any database setup or administration. The database is saved in the directory data/data/APP_NAME/databases/FILENAME.

So for our project the dictionary must exist in this directory :

data/data/com.example.translation/databases/EnglishDictionary

We don't need manually put the file in project dictionary so we decided to attach it in the assets folder so we can programmatically download it in the proper directory :

```
try {
String destPath = "/data/data/" + getPackageName() +
      "/databases/EnglishDictionary";
fileexists = true;
if (fileexists) {
CopyDB( getBaseContext().getAssets().open("EnglishDictionary"),
new FileOutputStream(destPath));
        }


} catch (FileNotFoundException e) {
        e.printStackTrace();
} catch (IOException e) {
        e.printStackTrace();
}
```

A java class is needed to be constructed representing the database. In that class we shall:

1. Set variables of the Database that shall never change, table attributes for example Identifier _id should be used for the primary key of the table. Several Android functions rely on this standard.

   Example :
   ```
   private final static String ROW_NAME_id = "word";
   private final static String ROW_MEAN_id = "mean";
   ```

2. Set Database name :

   Example :
   ```
   private final static String DATABASE_NAME = "EnglishDictionary";
   ```
3. Define the Database tables :

   Example:
   ```
   private final static String TABLE_NAME = "wordmeaning";
   ```

4. Sett the Database version

   Example:
   ```
   private final static int DATABASE_VERSION = 1;
   ```

now to create and upgrade the database in the application we usually subclass SQLiteOpenHelper. In the constructor of that subclass we call the super() method of SQLiteOpenHelper, specifying the database name and the current database version :

```
private static class DatabaseHelper extends SQLiteOpenHelper {
    public DatabaseHelper(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
            // TODO Auto-generated method stub
```

In this class we need to override the onCreate() and onUpgrade() methods. onCreate() is called by the framework :

```
public void onCreate(SQLiteDatabase db) {
                    try {
                          db.execSQL(DATABASE_CREATE);
                    } catch (SQLException e) {
                          e.printStackTrace();
                    }
               }
```

if the database does not exists. onUpgrade() is called, if the database version is increased in our application code. This method allows us to update the database schema. :

```
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
                    // TODO Auto-generated method stub
               }
```

Both methods receive a SQLiteDatabase object as parameter which represents the database. SQLiteOpenHelper provides the methods getReadableDatabase() and getWriteableDatabase() to get access to an SQLiteDatabase object; either in read or write mode.

When the user pressed on translate button A Cursor represents the result of a query and basically points to one row of the query result. This way Android can buffer the query results efficiently; as it does not have to load all data into memory. So if the result found it will return and shown in the translation box as follow :

```
String str;
str = RecTextView.getText().toString();
String result="";
int pos;
String temp=str+" ";
if(temp.indexOf(" ")>-1)
while(temp.indexOf(" ")>-1){
      pos=temp.indexOf(" ");
      result =result + dicsionarytranslate(temp.substring(0,pos)) +" ";
      temp=temp.substring(pos+1);
}
else {

      result=dicsionarytranslate(temp);
}
TranTextView.setText(result);
```

and if not exist a dialog box will appears asking the user if he need to add this word and its meaning to the dictionary :

4.3 Tesseract engine

We've been using tesseract to convert image into text. The quality of the images ranges wildly, and we are looking for tips on what sort of image processing might improve the results. we've noticed that text that is highly pixellated - for example that generated by fax machines - is especially difficult for tesseract to process - presumably all those jagged edges to the characters confound the shape-recognition algorithms .

Tesseract library was working well with desktop applications but not supporting Android, however a lot of its code was written in C, and then some more was written in C++ so its consider as a native language for android platform so we have been worked hardly to build up these libraries with android.

Our project is set up to build on Android SDK Tools r19 and Android NDK r8c. building up tesseract libraries with android is a very difficult process that requires accuracy with every step we will do, so it spent more effort and time that forced us calling experts to give us help building up that libraries, we will show how we work building in the next chapter.

Android actually uses the standard Java way to communicate with native code called JNI (Java Native Interface). It defines conventions and mechanisms that Java code and C/C++ code use to interact.

Now we can import the OCR engine as a library in Eclipse. File -> Import -> Existing Projects into workspace -> tess-two directory. Right click the project, Android Tools -> Fix Project Properties. Right click -> Properties -> Android -> Check Is Library.

We Configure our project to use tess-two project as a library project by making right click on project name -> Properties -> Android -> Library -> Add, and then choosing *tess-two*. We are now ready to OCR any image using this library .

After adding the Tess-Two the eclipse program gave us a wrapper called TessBaseAPI( ) that we can use to pass any image to the native tesseract library which in turn make necessary operations to extract text.

Now to work with this library we firstly declared a variable of type TessbaseAPI ( ) called TessApi like this :

```
        private TessBaseAPI TessApi;
```

Then we must initialize every field provided in tesseract, that fields was existed in the commercial version of Tesseract OCR and now after converting it to run on android we need a way to specify the value of these fields, so we first need to determine which segmentation mode we need to use, this controls how much processing the OCR engine will perform before recognizing text and we set it to be as auto segmentation to make fully page segmentation like the follow:

```
private int pageSegmentationMode = TessBaseAPI.PageSegMode.PSM_AUTO;
TessApi.setPageSegMode(pageSegmentationMode);
```

To communicate with the native code to set the value of previous variable we use this constructer :

```
public void setPageSegMode(int mode) {
        nativeSetPageSegMode(mode);
    }
```

Now we must determine the set of character to recognize by the engine and we denoted it by white list character and black list character for ones we don't need to recognized it :

```
private String characterBlacklist = "!?@#$%&*()<>_-+=/.,:;'\"ABCDEFGHIJKLMNOPQRSTUVWX
                                    YZabcdefghijklmnopqrstuvwxyz0123456789";

private String characterWhitelist = "";

TessApi.setVariable(TessBaseAPI.VAR_CHAR_BLACKLIST, characterBlacklist);
TessApi.setVariable(TessBaseAPI.VAR_CHAR_WHITELIST, characterWhitelist);
```

To communicate with the native code to set the value of previous variables we use this constructer :

```
public boolean setVariable(String var, String value) {
        return nativeSetVariable(var, value);
    }
```

Now we need to provide an image for Tesseract engine to recognize. There are two methods depending on user desire, if he want to use his mobile camera or if he want to load saved image in his gallery, so when he capture an image it passes to the engine through this command :

```
        TessApi.setImage(ReadFile.readBitmap(bitmap));
```

The image is set on the Bitmap variable but we need to convert it to a PIX variable so we use the command : `ReadFile.readBitmap(bitmap)`

To send the PIX data to the native code we use the following constructer :

```
public void setImage(Pix image) {
        nativeSetImagePix(image.getNativePix());
   }
```

In the case of loading pre saved image we must obtain the absolute path of the image and then use the following constructer to pass the image :

```
TessApi.setImage(file);
```

And then To send the PIX data to the native code we use the following constructer :

```
public void setImage(File file) {
        Pix image = ReadFile.readFile(file);

        if (image == null) {
            throw new RuntimeException("Failed to read image file");
        }

        nativeSetImagePix(image.getNativePix());
   }
```

At this stage tesseract performs all its processing algorithm beginning from line finding and ending with character recognition as detailed in the previous chapter .

Now we need to get handle with the result obtained by OCR engine. Here we built a new class to handle all results arrived from the OCR engine and we call it result.java . the important thing we need to get is the recognized text so we use the following command to get that text :

```
ResultText = TessApi.getUTF8Text();
```

The order getUTF8Text(); calls the constructer allocated to deal with native code :

```
public String getUTF8Text() {
 // Trim because the text will have extra line breaks at the end

        String text = nativeGetUTF8Text();

        return text.trim();
   }
```

And then the received text passed to the result class .

Chapter 5

Trails

Content

5.1 Android Native Development Kit (NDK)

1- Introduction:

Android programming depend on writing all code in Java, but sometimes it is not enough and we need to go to a native level and write part of the application in C/C++.

In our case, this is important when we already have some computer vision functionality which is written in C or C++, and we want to use it in the Android application, but do not want to rewrite the C code to Java. In this case the only way is to use JNI mechanism. It means, that a class with native methods wrapping the C functionality must be added into the Java part of the Android application.

NDK has a maintenance cost and add technical complexity to the app, but it's not difficult to install or be used.

2- Android Application Structure:

Usually code of an Android application has the following structure:

- root folder of the project/
  - jni/
  - libs/
  - res/
  - src/
  - AndroidManifest.xml
  - default.properties
  - other files ...

Where

- The libs folder will contain native libraries after successful build.
- The jni folder contains C/C++ application source code and NDK's build scripts Android.mk and Application.mk.

3- NDK System and Software Requirements:

step1: Setup NDK

As it was previously mentioned, to compile C++ code for Android platform you need Android Native Development Kit (NDK).

So first we need to get the latest version of NDK. we can get NDK from the official Android site.

Download the NDK zip for Windows and extract it somewhere, but again, be s ure that there are no spaces in the path.

step2: Installing C/C++ Support on Eclipse:

Eclipse installation does not support either C or C++. we don't need the full support. But we would like to have syntax coloring and basic syntax checking. Thus we have to add some Eclipse features via the update mechanism. By going to Help then Install New Software menu item. And Choosing Juno as the update site. after the item tree load, we checked Eclipse C/C++ Development Tools in the Programming Languages branch then the tools began installing .

Step3: Installing Cygwin

Android is Linux based, and thus it is no surprise that when we build native code for it, we need some UNIX tools. On Windows, NDK supports Cygwin 1.7.x and above.

Cygwin is:

- a collection of tools which provide a Linux look and feel environment for Windows
- a DLL (cygwin1.dll) which acts as a Linux API layer providing substantial Linux API functionality.

So Cygwin emulates UNIX environment on windows.[15]

Step 4: Making NDK application:

1. Making a folder called jni in the root of the project (right-click the project node, New Folder) then we create a file called Android.mk (New – File) within that folder with the following contents :

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
# Here we give our module name and source file(s)
LOCAL_MODULE     := ndkfoo
LOCAL_SRC_FILES := ndkfoo.c
include $(BUILD_SHARED_LIBRARY)
```

then after filling out our project content the Android.mk file becomes as follow :

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE := libtess

# tesseract (minus executable)

BLACKLIST_SRC_FILES := \
  %api/tesseractmain.cpp \
  %viewer/svpaint.cpp

TESSERACT_SRC_FILES := \
  $(wildcard $(TESSERACT_PATH)/api/*.cpp) \
  $(wildcard $(TESSERACT_PATH)/ccmain/*.cpp) \
  $(wildcard $(TESSERACT_PATH)/ccstruct/*.cpp) \
  $(wildcard $(TESSERACT_PATH)/ccutil/*.cpp) \
  $(wildcard $(TESSERACT_PATH)/classify/*.cpp) \
  $(wildcard $(TESSERACT_PATH)/cube/*.cpp) \
  $(wildcard $(TESSERACT_PATH)/cutil/*.cpp) \
  $(wildcard $(TESSERACT_PATH)/dict/*.cpp) \
  $(wildcard $(TESSERACT_PATH)/image/*.cpp) \
  $(wildcard $(TESSERACT_PATH)/neural_networks/runtime/*.cpp) \
  $(wildcard $(TESSERACT_PATH)/textord/*.cpp) \
  $(wildcard $(TESSERACT_PATH)/viewer/*.cpp) \
  $(wildcard $(TESSERACT_PATH)/wordrec/*.cpp)

LOCAL_SRC_FILES := \
  $(filter-out $(BLACKLIST_SRC_FILES),$(subst $(LOCAL_PATH)/,,$(TESSERACT_SRC_FIL

LOCAL_C_INCLUDES := \
  $(TESSERACT_PATH)/api \
  $(TESSERACT_PATH)/ccmain \
  $(TESSERACT_PATH)/ccstruct \
  $(TESSERACT_PATH)/ccutil \
  $(TESSERACT_PATH)/classify \
  $(TESSERACT_PATH)/cube \
  $(TESSERACT_PATH)/cutil \
  $(TESSERACT_PATH)/dict \
  $(TESSERACT_PATH)/image \
  $(TESSERACT_PATH)/neural_networks/runtime \
  $(TESSERACT_PATH)/textord \
  $(TESSERACT_PATH)/viewer \
  $(TESSERACT_PATH)/wordrec \
  $(LEPTONICA_PATH)/src

# jni

LOCAL_SRC_FILES += \
  tessbaseapi.cpp

LOCAL_C_INCLUDES += \
  $(LOCAL_PATH)

include $(BUILD_SHARED_LIBRARY)

```

The Android.mk file is an important file for the NDK build process to recognize our NDK modules. In our case we named our module libtess and told the build tool that it consists of many source files named as shown in the previous code under the label `TESSERACT_SRC_FILES` and `LOCAL_C_INCLUDES`.

2. Creating a tessbaseapi.cpp in the same jni folder to work as an interface to all native code used by Tesseract, so it used to initialize OCR with necessary parameter discussed previously as passing captured image and choosing segmentation mode and etc... and to get results that out from the OCR such as recognized text .

As an example the following is a C++ language code used to set image into the OCR engine :

```
void Java_com_googlecode_tesseract_android_TessBaseAPI_nativeSetImagePix(JNIEnv *env,
                                                                          jobject thiz,
                                                                          jint nativePix) .


  PIX *pixs = (PIX *) nativePix;
  PIX *pixd = pixClone(pixs);

  native_data_t *nat = get_native_data(env, thiz);
  nat->api.SetImage(pixd);


  }
```

And for getting result text we use this code :

```
jstring Java_com_googlecode_tesseract_android_TessBaseAPI_nativeGetUTF8Text(JNIEnv *env,
                                                                           jobject thiz)


  native_data_t *nat = get_native_data(env, thiz);

  char *text = nat->api.GetUTF8Text();

  jstring result = env->NewStringUTF(text);

  free(text);

  return result;
}
```

we might notice that the name of the C function is not just random – it matches the Java class name.

In addition, what the function does is it uses the JNIEnv object to create a Java string from a literal, and returns the string to the caller.

4- Building the application :

In order to create a binary library from the C source that we wrote, we will use a combination of Cygwin and Android NDK tools.

After launch the Cygwin console we use the command line to do the following steps to build libraries :

- we write "git clone git://github.com/rmtheis/tess-two tess" to get the fork of Tesseract files for android .

- Now we write : cd /cygdrive<project-directory>/tess-two to enter inside Tess-Two dicrectory.
- We must set the path of directory containing the Tesseract configure file and source folders to its respective source path before compiling, To set the variables, we must run the following shell command:

```
# export TESSERACT_PATH=path-to-tesseract
```

- Also we must running the following command in the Terminal before ndk-build in next step :

```
#export TESSERACT_PATH=${PWD}/external/tesseract-3.01
```

- Now we will build the ndk to compile the source of c and cpp file with the help of Android NDK tool by running the next command and wait until finish:

```
<ndk-full directory>/ndk-build
```

That will generate .so files inside lib folder .

- We must update PATH variable for the commands to function, otherwise we would see a command not found error. For Android SDK, we add the location of the SDK's tools and platform-tools directories to the PATH environment variable. For Android NDK, we used the same process to add the android-ndk directory to the PATH variable.
- Now write " android update project –path" to update project paths .
- Finally write " ant release " to finish building .

a successful build of the ndk-build tool will create an "com_googlecode_tesseract _android" file in the jni folder. That contains all binary library that will be included into the application .apk package and will be available for the Java code of the app to link to. We just need to Refresh the project after selecting the project root to update the it with the changes that have been made in the Cygwin console.

The ndk-build command has to be repeated every time we modify the C/C++ source of your NDK code. Eclipse ADT does not support NDK so we need to do it from the Cygwin console.

5- Native Activity Architecture

After the NDK part is actually finished. What we need to do is to change the Java code of the TessBaseApi class to use the NDK code :

```
static {

        // loading the library - name matches jni/Android.mk
           System.loadLibrary("tess");

         nativeClassInit();
        }
// this is where we call the native code
   private native void nativeSetImagePix(int nativePix);
  private native String nativeGetUTF8Text();
```

5.2 JJIL

We tried to use JJIL library for java, many functions didn't work like finding contours and bounding box. The news group of JJIL is not active and it wasn't used for over a year, so we left the library and searched for other libraries.

5.3 JavaCV

This library was working well with desktop applications but not supporting Android so we excluded it

5.4 Leptonica

Leptonica is a pedagogically-oriented open source project containing software that is broadly useful for image analysis applications.

This application useful for making image enhancements so we invoked it as the way we have invoked Tesseract to make some enhancements on the captured image before it take its way to Tessercat engine . So we aim to increase the accuracy in recognition using this project. [16]

Featured operations:

- Affine transformations (scaling, translation, rotation, shear) on images of arbitrary pixel depth.
- Binary and grayscale morphology, rank order, and convolution

- Seed fill and connected components
- Image transformations combining changes in scale and pixel depth
- Pixelwise masking, blending, enhancement, arithmetic ops, etc.[17]

So we need to use some of these feature to get better results .

What we need to do is to change the Java code of the TessBaseApi class to use the NDK code of this application :

```
static {
       // loading lebtonica library - name matches jni/Android.mk
           System.loadLibrary("lebt");
       // loading tess-two library - name matches jni/Android.mk
           System.loadLibrary("tess");
         nativeClassInit();
        }
```

Chapter Six
Testing and enhancements

6.1 Introduction :

After finishing the project we have tested it on some samples with different fonts and colors also with different quality in order to test our project ability to recognize these words and calculating the accuracy in the results. As we noticed that the accuracy can be increased if the image noise and distortion was reduced before it comes into the OCR engine. So in this chapter we will try to examine the results before and after adding the enhancements on the captured image.

OCR engine calculates a confidence level for each character it detects. Word and page confidence levels can be calculated from the character confidences using algorithms inside the OCR. The OCR doesn't know whether any character is correct or not, it can only be confident or not confident that it is correct. It will give it a confidence level from 0-9.

6.2 PreProcessig Enhancements

It contains on converting the colored image into a gradient scale from white to black depending on each pixel luminance. We use the Leptonica library for conversion to gray scale. This method give us better accuracy of mean confidence of 9 but sometime it work badly when we use samples with overlapped text colors.

6.3 Image sharpening method

Performs unsharp masking using symmetric smoothing filters of odd dimension for edge enhancement, this method enhanced the accuracy even when we use colored image.

6.4 Image Binarization Method

Creates a binary version of the image using the global thresholding technique. to automatically perform histogram shape-based image thresholding, or, the reduction of a gray level image to a binary image. This method works better with overlapped color image.

6.5 Image adaptive mapping methods

Using this method to reduce the noise and distortion to get better looking for the image before it enters the OCR. This method increases the confidence for each character and then getting more accuracy.

Finally, we merge between all previous method to get best accuracy and submit one method for enhancements to all captured pictures beginning by converting the image into grayscale then blurring and thresholeding.

References

1. http://www.jiscdigitalmedia.ac.uk/guide/optical-character-recognition-ocr
2. ww.r2integrated.com/portals/0/pdfs/indusry-trends/R2i_WP_SmartphoneComparison.pdf
3. www.enggjournals.com/ijcse/doc/..3-IJCSESP24.pdf
4. Wiley Publishing , "AndroidTM Application Development for Dummies" , page 8 – 23, Nov. 2011
5. www.artima.com/insidejvm/ed2/introachP.html
6. http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.html
7. http://www.techterms.com/definition/grayscale
8. http://www.dfki.uni-kl.de/~shafait/papers/Shafait-efficient-binarization-SPIE08.pdf
9. http://kv-emptypages.blogspot.com/2011/06/analysis-of-shutdown-announcements-of.html
10. https://developers.google.com/translate/?hl=ar
11. http://www.microsoft.com/web/post/using-the-free-bing-translation-apis
12. http://msdn.microsoft.com/en-us/library/hh456380.aspx
13. http://www.jiahaoliuliu.com/2011/03/android-activity-life-cycle-from.html
14. http://developer.android.com/guide/topics/ui/declaring-layout.html
15. http://mindtherobot.com/blog/452/android-beginners-ndk-setup-step-by-step/
16. http://www.leptonica.com/
17. Ray Smith," An Overview of the Tesseract OCR Engine".
18. Brian M. Gonzalez," A Solution for Executing Handwritten Code" Spring 2012.
19. R. Plamondon and S. Srihari, "Online and off-line handwriting recognition: a comprehensive survey," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 22, pp. 63 –84, jan 2000.
20. R. Plamondon and S. Srihari, "Online and off-line handwriting recognition: a comprehensive survey," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 22, pp. 63 –84, jan 2000.
21. N. Senthilkumaran, R. Rajesh, "Edge Detection Techniques for Image Segmentation and A Survey of Soft Computing Approaches", International Journal of Recent Trends in Engineering, Vol. 1, No. 2, PP.250-254, May 2009.