Augmented Reality Arabic Translation

Dr/ Walid Khedr

2013

Augmented Reality Arabic Translation

BY

- Sherif Ismail Hassan Alwelaily
- Mustafa Mohammed Mustafa Abo-Ghida
- Abd-elazeem Mohammed Abd-elazeem
- Abanoub Eed Labeeb Adeeb

A Graduation Project Report Submitted to
The Faculty of Computer and informatics at Zagazig University
In Partial Fulfillment of the Requirements for the
Degree of
Bachelor of Science
In Information technology

Supervisor

Dr. Walid Khedr

Faculty of Computer and informatics, Zagazig University

Zagazig, Egypt

ACKNOWLEDGEMENT

After giving thanks to Allah;

First and foremost, we would like to express our sincere gratitude and respect to our faculty staff members specially

Project supervisor: ***Dr. Walid Khedr***

for his advice, guidance, supervision and patience.

Special thanks go to ***Dr. Mahmoud Osman*** for his effort to help us.

Special thanks go to ***ENG. Hanaa Hamza*** for sharing her ideas to help us and for her advice and supervision.

Besides that, we would like to thank our families for their understanding, encouragement and support.

And last but not least thank our friends hope Allah Grace them.

FINALLY:

For any person read this documentation, we want you to know that all the ideas and algorithms that used at this project are product of our mind and effort. We still work to develop it for last moment.

And our goal is producing our best we have at this project to produce application is ready to use at the market.

Abstract

The problem that may face us when we want to translate any words is the time of search or typing the words if we translate by software program and the typing also consume time.

With the development in technologies we need to consume less time and without any effort.

We think if you have your mobile application that has ability to translate at live mode by using mobile's camera that is better, faster and helpful for user.

Our application translates form Arabic to English and vice versa without typing the words, Perform a live translation by using the mobile camera.

So this application can be used as a dictionary, where, everybody at everywhere can be helped using it, to translate any text from Arabic to English and vice versa.

Table of Contents

Chapter1:

Introduction

Project Field

- Mobile Application (android)

Description of the Problem

- The problem that may face us when we want to translate any words is the time of search or typing the words if we translate by software program and the typing also consume time.
- With the development in technologies we need to consume less time and without any effort.

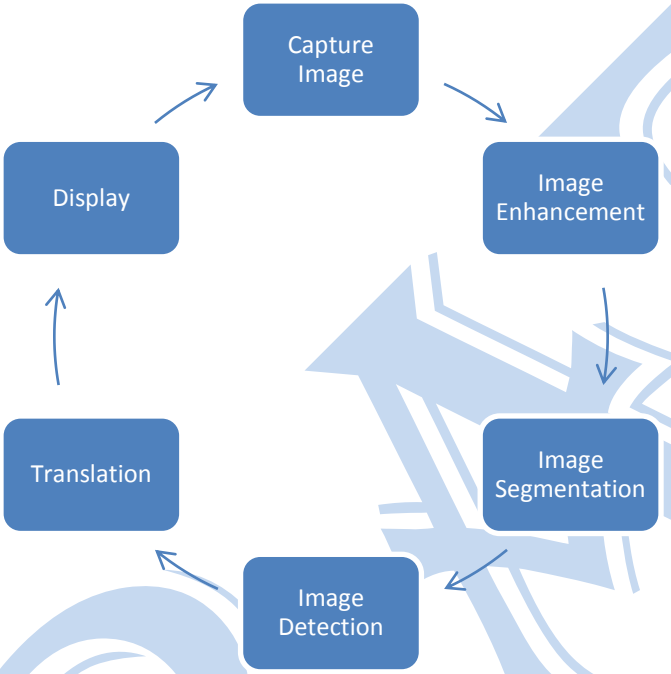
Project Objectives

- We think if you have your mobile application that has ability to translate at live mode by using mobile's camera that is better, faster and helpful for user.
- Our application translates form Arabic to English and vice versa without typing the words, Perform a live translation by using the mobile camera.

Importance and Relevance to people

- So this application can be used as a dictionary, where, everybody at everywhere can be helped using it, to translate any text from Arabic to English and vice versa at live manner.

The overall project



Chapter2:

Tools

1.1. Augmented Reality

What Is Augmented Reality?

Augmented Reality (AR) is a variation of *Virtual Environments* (VE), or Virtual Reality as it is more commonly called. VE technologies completely immerse a user inside a synthetic environment. While immersed, the user cannot see the real world around him. In contrast, AR allows the user to see the real world, with virtual objects superimposed upon or composited with the real world. Therefore, AR supplements reality, rather than completely replacing it. Ideally, it would appear to the user that the virtual and real objects coexisted in the same space, similar to the effects achieved in the film "Who Framed Roger Rabbit?" Figure 1 shows an example of what this might look like. It shows a real desk with a real phone. Inside this room are also a virtual lamp and two virtual chairs. Note that the objects are combined in 3-D, so that the virtual lamp covers the real table, and the real table covers parts of the two virtual chairs. AR can be thought of as the "middle ground" between VE (completely synthetic) and telepresence (completely real)

Figure 1: Real desk with virtual lamp and two virtual chairs. (Courtesy ECRC)

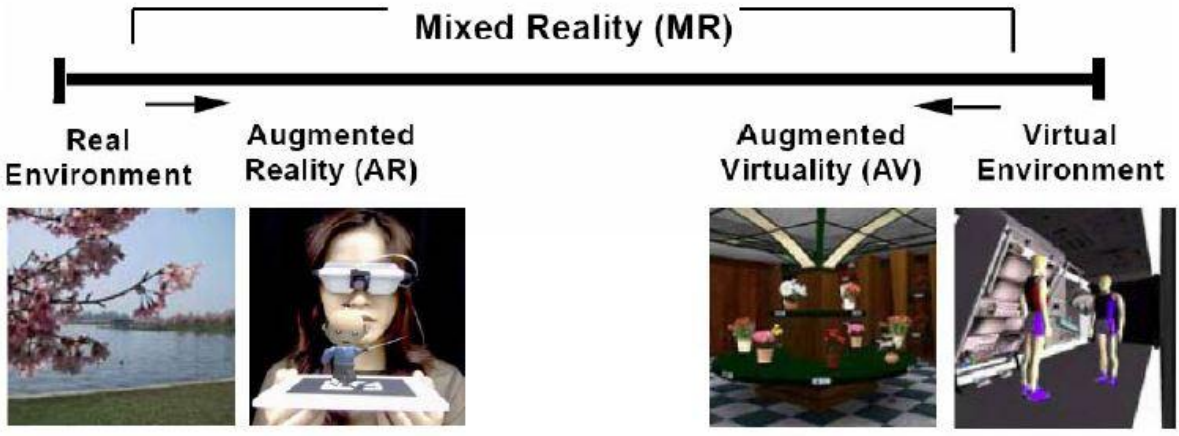
Some researchers define AR in a way that requires the use of Head-Mounted Displays (HMDs). To avoid limiting AR to specific technologies, this survey defines AR as systems that have the following three characteristics:

1) combines real and virtual
2) Interactive in real time
3) Registered in 3-D

This definition allows other technologies besides HMDs while retaining the essential components of AR. For example, it does not include film or 2-D overlays. Films like "Jurassic Park" feature photorealistic virtual objects seamlessly blended with a real environment in 3-D, but they are not interactive media. 2-D virtual overlays on top of live video can be done at interactive rates, but the overlays are not combined with the real world in 3-D. However, this definition does allow monitor based interfaces, monocular systems, see-through HMDs, and various other combining technologies.

Theory:

Augmenting natural feedback to the operator with simulated cues.



Motivation:

Why is Augmented Reality an interesting topic? Why is combining real and virtual objects in 3-D useful? Augmented Reality enhances a user's perception of and interaction with the real world. The virtual objects display information that the user cannot directly detect with his own senses. The information conveyed by the virtual objects helps a user perform real-world tasks. AR is a specific example of what Fred Brooks calls *Intelligence Amplification* (IA): using the computer as a tool to make a task easier for a human to perform.
At least six classes of potential AR applications have been explored: medical visualization, maintenance and repair, annotation, robot path planning, entertainment, and military aircraft navigation and targeting. The next section describes work that has been done in each area. While these do not cover every potential application area of this technology, they do cover the areas explored so far.

Characteristics:

This section discusses the characteristics of AR systems and design issues encountered when building an AR system. discusses their characteristics and relative strengths and weaknesses. Blending the real and virtual poses problems with focus and contrast (Section 3.3), and some applications require portable AR systems to be truly effective (Section 3.4). Finally, Section 3.5 summarizes the characteristics by comparing the requirements of AR against those for Virtual Environments.

Augmentation

Describes the basic characteristics of augmentation. There are two ways to accomplish this augmentation: optical or video technologies.
Besides *adding* objects to a real environment, Augmented Reality also has the potential to *remove* them. Current work has focused on adding virtual objects to a real environment. However, graphic overlays might also be used to remove or hide parts of the real environment from a user. Augmented Reality might apply to all senses, not just sight.

Optical vs. video

A basic design decision in building an AR system is how to accomplish the combining of real and virtual. Two basic choices are available: optical and video technologies.

Focus and contrast

Focus can be a problem for both optical and video approaches. Ideally, the virtual should match the real. In a video-based system, the combined virtual and real image will be projected at the same distance by the monitor or HMD optics. However, depending on the video camera's depth-of-field and focus settings, parts of the real world may not be in focus. In typical graphics software, everything is rendered with a pinhole model, so all the graphic objects, regardless of distance, are in focus. To overcome this, the graphics could be rendered to simulate a limited depth-of-field, and the video camera might have an autofocus lens.

In the optical case, the virtual image is projected at some distance away from the user. This distance may be adjustable, although it is often fixed. Therefore, while the real objects are at varying distances from the user, the virtual objects are all projected to the same distance. If the virtual and real distances are not matched for the particular objects that the user is looking at, it may not be possible to clearly view both simultaneously.

Contrast is another issue because of the large dynamic range in real environments and in what the human eye can detect. Ideally, the brightness of the real and virtual objects should be appropriately matched. Unfortunately, in the worst case scenario, this means the system must match a very large range of brightness levels. The eye is a logarithmic detector, where the brightest light that it can handle is about eleven orders of magnitude greater than the smallest, including both dark-adapted and light-adapted eyes. In any one adaptation state, the eye can cover about six orders of magnitude. Most display devices cannot come close to this level of contrast. This is a particular problem with optical technologies, because the user has a direct view of the real world. If the real environment is too bright, it will wash out the virtual image. If the real environment is too dark, the virtual image will wash out the real world. Contrast problems are not as

severe with video, because the video cameras themselves have limited dynamic response, and the view of both the real and virtual is generated by the monitor, so everything must be clipped or compressed into the monitor's dynamic range.

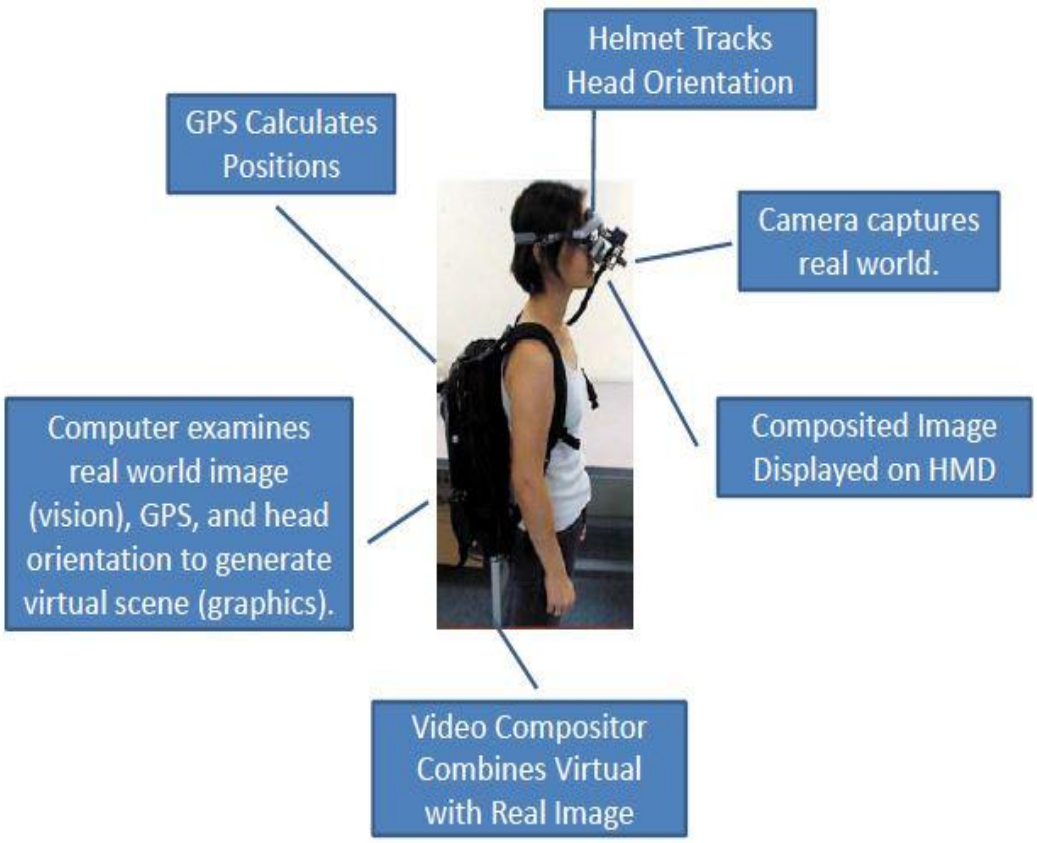
Portability

In almost all Virtual Environment systems, the user is not encouraged to walk around much. Instead, the user navigates by "flying" through the environment, walking on a treadmill, or driving some mockup of a vehicle. Whatever the technology, the result is that the user stays in one place in the real world.

Some AR applications, however, will need to support a user who will walk around a large environment. AR requires that the user actually be at the place where the task is to take place. "Flying," as performed in a VE system, is no longer an option. If a mechanic needs to go to the other side of a jet engine, she must physically move herself and the display devices she wears. Therefore, AR systems will place a premium on portability, especially the ability to walk around outdoors, away from controlled environments. The scene generator, the HMD, and the tracking system must all be self-contained and capable of surviving exposure to the environment. If this capability is achieved, many more applications that have not been tried will become available. For example, the ability to annotate the surrounding environment could be useful to soldiers, hikers, or tourists in an unfamiliar new location.

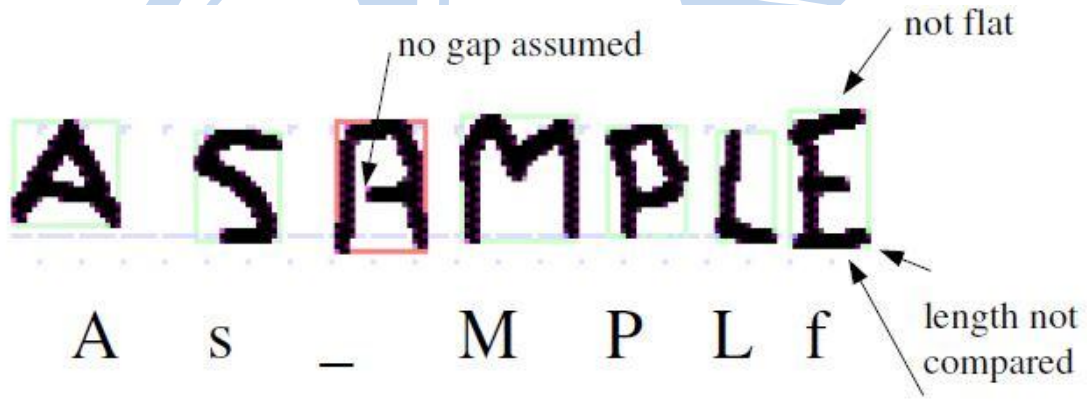
Building AR Systems

Building AR Systems is HARD.



1.2. OCR

Optical character recognition (**OCR)** is the mechanical or electronic conversion of scanned images of handwritten, typewritten or printed text into machine-encoded text. It is widely used as a form of data entry from some sort of original paper data source, whether documents, sales receipts, mail, or any number of printed records. It is a common method of digitizing printed texts so that they can be electronically searched, stored more compactly, displayed on-line, and used in machine processes such as machine translation, text-to-speech and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision. Early versions needed to be programmed with images of each character, and worked on one font at a time. "Intelligent" systems with a high degree of recognition accuracy for most fonts are now common. Some systems are capable of reproducing formatted output that closely approximates the original scanned page including images, columns and other non-textual components.



Pre-processing:

OCR software often "pre-processes" images to improve the chances of successful recognition. Techniques include:

- If the document was not aligned properly when scanned, it may need to be tilted a few degrees clockwise or counterclockwise in order to make lines of text perfectly horizontal or vertical.
- Remove positive and negative spots, smoothing edges.
- Convert an image from color or gray to black-and-white (called a “binary image" because there are two colors). In some cases, this is necessary for the character recognition algorithm; in other cases, the algorithm performs better on the original image and so this step is skipped
- Cleans up non-glyph boxes and lines
- Layout analysis or "zoning" - Identifies columns, paragraphs, captions, etc. as distinct blocks. Especially important in multi-Column layouts and tables.
- Line and word detection - Establishes baseline for word and character shapes, separates words if necessary.
- Character isolation or "segmentation" - For per-character OCR, multiple characters that are connected due to image artifacts must be separated; single characters that are broken into multiple pieces due to artifacts must be connected.
- Normalize aspect ratio and scale
- Segmentation of fixed-pitch fonts is accomplished relatively simply by aligning the image to a uniform grid based on where vertical grid lines will least often intersect black areas. For proportional fonts, more sophisticated techniques are needed because whitespace between letters can sometimes be greater than that between words, and vertical lines can intersect more than one character.

Post-processing:

OCR accuracy can be increased if the output is constrained by a lexicon - a list of words that are allowed to occur in a document. This might be, for example, all the words in the English language, or a more technical lexicon for a specific field. This technique can be problematic if the document contains words not in the lexicon, like proper nouns. Tesseract uses its dictionary to influence the character segmentation step, for improved accuracy.

The output stream may be a plain text stream or file of characters, but more sophisticated OCR systems can preserve the original layout of the page and produce, for example, an annotated PDF that includes both the original image of the page and a searchable textual representation.

"Near-neighbor analysis" can make use of co-occurrence frequencies to correct errors, by noting that certain words are often seen together. For example, "Washington, D.C." is generally far more common in English than "Washington DOC".

Knowledge of the grammar of the language being scanned can also help determine if a word is likely to be a verb or a noun, for example, allowing greater accuracy

Application-specific optimizations

In recent years, the major OCR technology providers began to tweak OCR systems to better deal with specific types of input. Beyond an application-specific lexicon, better performance can be had by taking into account business rules, standard expression, or rich information contained in color images. This strategy is called "Application-Oriented OCR" or "Customized OCR", and has been applied to OCR of license plates, business cards, invoices, screenshots, ID cards, driver licenses, and automobile manufacturing

1.3. SQL Light

is a relational database management system contained in a small (~350 KB) C programming library. In contrast to other database management systems, SQLite is not a separate process that is accessed from the client application, but an integral part of it.

SQLite is ACID-compliant and implements most of the SQLstandard, using a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity.

SQLite is a popular choice as embedded Database for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating system, embedded systems and, among others. SQLite has many bindings to programming languages.

Design:

Unlike client–server database management systems, the SQLite engine has no standalone processes with which the application program communicates. Instead, the SQLite library is linked in and thus becomes an integral part of the application program. (In this, SQLite follows the precedent of Informix SE of c. 1984) The library can also be called dynamically.

The application program uses SQLite's functionality through simple function calls, which reduce latency in database access: function calls within a single process are more efficient than inter-process communication. SQLite stores the entire database (definitions, tables, indices, and the data itself) as a single cross-platform file on a host machine. It implements this simple design by locking the entire database file during writing. SQLite read operations can be multitasked, though writes can only be performed sequentially.

Features:

SQLite implements most of the SQL-92 standard for SQL but it lacks some features. For example it has partial support for triggers, and it can't write to views (however it supports INSTEAD OF triggers that provide this functionality). While it supports complex queries, it still has limited ALTER TABLE support, as it can't modify or delete columns.

SQLite uses an unusual type system for an SQL-compatible DBMS. Instead of assigning a type to a column as in most SQL database systems, types are assigned to individual values; in language terms it is *dynamically typed*. Moreover, it is *weakly typed* in some of the same ways that Perl is: one can insert a string into an integer column (although SQLite will try to convert the string to an integer first, if the column's preferred type is integer). This adds flexibility to columns, especially when bound to a dynamically typed scripting language. However, the technique is not portable to other SQL products. A common criticism is that SQLite's type system lacks the data integrity mechanism provided by statically typed columns in other products. The SQLite web site describes a "strict affinity" mode, but this feature has not yet been added. However, it can be implemented with constraints like

```
CHECK (typeof(x) ='integer')
```

Several computer processes or threads may access the same database concurrently. Several read accesses can be satisfied in parallel. A write access can only be satisfied if no other accesses are currently being serviced. Otherwise, the write access fails with an error code (or can automatically be retried until a configurable timeout expires). This concurrent access situation would change when dealing with temporary tables. This restriction is relaxed in version 3.7 when WAL is turned on enabling concurrent reads and writes.

A standalone program called sqlite3 is provided that can be used to create a database, define tables within it, insert and change rows, run

queries and manage an SQLite database file. This program is a single executable file on the host machine. It also serves as an example for writing applications that use the SQLite library.

SQLite is a popular choice for local/client SQL storage within a web browser and within a rich internet application framework; most notably the leaders in this area (Google Gears, Adobe AIR and Firefox) embed SQLite.

SQLite full Unicode support is optional.

SQLite also has bindings for a large number of programming language, including BASIC,C,C++,C# ,Curl ,Free Pascal ,Java , PHP, Python , Visual Basic and Java Script. An ADO.NET adapter, initially developed by Robert Simpson, is maintained jointly with the SQLite developers since April 2010. An ODBS driver has been developed and is maintained separately by Christian Werner. Werner's ODBC driver is the recommend connection method for accessing SQLite from Open Office. There is also a COM (ActiveX) wrapper making SQLite accessible on Windows to scripted languages such as JScript and VBScript. This adds database capabilities to HTML Application (HTA).

SQLite has automated regression testing prior to each release. Over 2 million tests are run as part of a release's verification. Starting with the August 10, 2009 release of SQLite 3.6.17, SQLite releases have 100% branch test coverage, one of the components of code coverage.

1.4. Android

What is android?

Android is a mobile operating system that is based on a modified version of Linux. It was originally developed by a startup of the same name, Android, Inc. In 2005, as part of its strategy to enter the mobile space, Google purchased Android and took over its development work (as well as its development team).

Google wanted Android to be open and free; hence, most of the Android code was released under the open-source Apache License, which means that anyone who wants to use Android can do so by downloading the full Android source code. Moreover, vendors (typically hardware manufacturers) can add their own proprietary extensions to Android and customize Android to differentiate their products from others. This simple development model makes Android very attractive and has thus piqued the interest of many vendors. This has been especially true for companies affected by the phenomenon of Apple's iPhone, a hugely successful product that revolutionized the smartphone industry.

Such companies include Motorola and Sony Ericsson, which for many years have been developing their own mobile operating systems. When the iPhone was launched, many of these manufacturers had to scramble to find new ways of revitalizing their products. These manufacturers see Android as a solution — they will continue to design their own hardware and use Android as the operating system that powers it.

The main advantage of adopting Android is that it offers a unified approach to application development. Developers need only develop for Android, and their applications should be able to run on numerous different devices, as long as the devices are powered using Android. In the world of smartphones, applications are the most important part of the success chain. Device manufacturers therefore see Android as their best hope to challenge the onslaught of the iPhone, which already commands a large base of applications.

Android application version

Android is software for mobile devices that includes an operating system and key applications. Google Inc. purchased the initial developer of the software, Android Inc., in 2005. Android's mobile operating system is based on the Linux kernel. Google and other members of the Open Handset Alliance (business alliance of 80 firms to develop standards for mobile devices) collaborated on Android's development and release. The Android Open Source Project (AOSP) is tasked with the maintenance and further development of Android. The Android operating system is the world's best-selling Smartphone platform. Android has a large community of developers writing applications ("apps") that extend the functionality of the devices. There are currently over 200,000 apps available for Android. Android Market is the online app store run by Google, though apps can also be downloaded from third-party sites.

Developers write primarily in the Java language, controlling the device via Google-developed Java libraries. The Android operating system includes the Linux kernel. The unveiling of the Android distribution on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 80 hardware, software and telecom companies devoted to advancing open standards for mobile devices.

The Android operating system is used on Smartphones, notebooks, tablets, Google TV and other devices.

Version history

- Android has seen a number of updates since its original release. These updates to the base operating system typically fix bugs and add new features. Past updates included Cupcake and Donut. The code names are in alphabetical order (Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, and the upcoming Ice Cream Sandwich). The most recently released versions of Android are:
- Android 1.5
- Android 1.6
- 2.0/2.1 (Eclair), which revamped the user interface and introduces HTML5, W3C Geolocation API and Exchange ActiveSync 2.5 support.
- 2.2 (Froyo), which introduced speed improvements with JIT optimization and the Chrome V8 JavaScript engine, and added Wi-Fi hotspot tethering and Adobe Flash support.
- 2.3 (Gingerbread), which refined the user interface, improved the soft keyboard and copy/paste features, and added support for Near Field Communication.
- 3.0/3.1 (Honeycomb), a tablet-oriented release which supports larger screen devices and introduces many new user interface features, and supports multi-core processors and hardware acceleration for graphics.
- 4.0 (Ice Cream Sandwich), a combination of Gingerbread and Honeycomb. It was announced on May 10, 2011 at Google I/O that it will be released in Q4 2011 [10]. The following pie chart is based on the number of Android devices that have accessed Android Market within a 14-day period ending on the data collection date noted below.

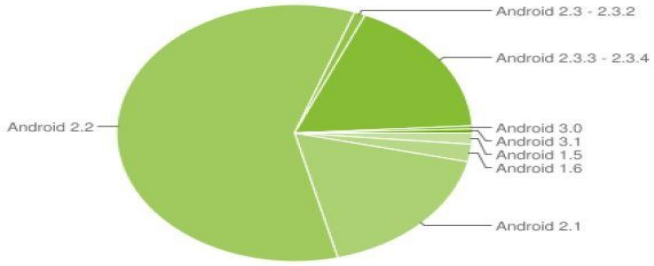


Figure 2.1 pie chart of the number of Android devices that have accessed Android Market within a 14-day period

Features of Android

As Android is open source and freely available to manufacturers for customization, there are no fixed hardware and software configurations. However, Android itself supports the following features:

Storage: Uses SQLite, a lightweight relational database, for data storage.

Connectivity: Supports GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth (includes A2DP and AVRCP), WiFi, LTE, and WiMAX. Chapter 8 discusses networking in more detail.

Messaging: Supports both SMS and MMS.

Web browser: Based on the open-source WebKit, together with Chrome's V8 JavaScript engine

Media support: Includes support for the following media: H.263, H.264 (in 3GP or MP4 container), MPEG-4 SP, AMR, AMR-WB (in 3GP container), AAC, HE-AAC (in MP4 or 3GP container), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP

Hardware support — Accelerometer Sensor, Camera, Digital Compass, Proximity Sensor, and GPS

Multi-touch: Supports multi-touch screens

Multi-tasking: Supports multi-tasking applications

Flash support — Android 2.3 supports Flash 10.1.

Tethering: Supports sharing of Internet connections as a wired/wireless hotspot.

Android Devices in the Market

Android devices come in all shapes and sizes. As of late November 2010, the Android OS can be seen powering the following types of devices:

- Smartphones
- Tablets
- E-reader devices
- Netbooks
- MP4 players
- Internet TVs

Chances are good that you own at least one of the preceding devices. Figure 1-2 shows (clockwise) the Samsung Galaxy S, the HTC Desire HD, and the LG Optimums One smartphones.

Another popular category of devices that manufacturers are rushing out is the *tablet*. Tablet sizes typically start at seven inches, measured diagonally. Figure 1-3 shows the Samsung Galaxy Tab and the Dell Streak, which is a five inch phone tablet.

FIGURE 1-2

Besides smartphones and tablets, Android is also beginning to appear in dedicated devices, such as e-book readers. Figure 1-4 shows the Barnes and noble's NOOK color, which is a color e-Book reader running the Android OS.

FIGURE 1-3

FIGURE 1-4

In addition to these popular mobile devices, Android is also slowly finding its way into your living room. People of Lava, a Swedish company, has developed an Android-based TV, call the Scandinavia Android TV (see Figure 1-5).

Google has also ventured into a proprietary smart TV platform based on Android and co-developed with companies such as Intel, Sony, and Logitech. Figure 1-6 shows Sony's Google TV.

FIGURE 1-5

FIGURE 1-6

OBTAINING THE REQUIRED TOOLS

Now that you know what Android is and its feature set, you are probably anxious to get your hands dirty and start writing some applications! Before you write your first app, however, you need to download the required tools and SDKs.

For Android development, you can use a Mac, a Windows PC, or a Linux machine. All the tools needed are free and can be downloaded from the Web. Most of the examples provided in this book should work fine with the Android emulator, with the exception of a few examples that require access to the hardware.

So, let the fun begin!

JAVA JDK

The Android SDK makes use of the Java SE Development Kit (JDK). Hence, if your computer does not have the JDK installed, you should start by downloading the JDK from www.oracle.com/technetwork/java/javase/downloads/index.html and installing it prior to moving to the next section.

Eclipse

The first step towards developing any applications is obtaining the integrated development environment (IDE). In the case of Android, the recommended IDE is Eclipse, a multi-language software development environment featuring an extensible plug-in system. It can be used to develop various types of applications, using languages such as Java, Ada, C, C++, COBOL, Python, etc.

For Android development, you should download the Eclipse IDE for Java EE Developers (www.eclipse .org/downloads/packages/eclipse-ide-java-eedevelopers/heliossr1). Six editions are available: Windows (32 and 64-bit), Mac OS X (Cocoa 32 and 64), and Linux (32 and 64-bit). Simply select the relevant one for your operating system. All the examples in this book were tested using the 32-bit version of Eclipse for Windows.

Once the Eclipse IDE is downloaded, unzip its content (the eclipse folder) into a folder, say C:\Android\.

Android SDK

The next important piece of software you need to download is, of course, the Android SDK. The Android SDK contains a debugger, libraries, an emulator, documentation, sample code, and tutorials.
You can download the Android SDK from http://developer.android.com/sdk/index.html.

Once the SDK is downloaded, unzip its content (the android-sdk windows folder) into the C:\Android\ folder, or whatever name you have given to the folder you just created.

Anatomy of an Android Application

Now that you have created your first Hello World Android application, it is time to dissect the innards of the Android project and examine all the parts that make everything work.
First, note the various files that make up an Android project in the Package Explorer in Eclipse.
The various folders and their files are as follows:

- src: Contains the .java source files for your project.
- Android library: This item contains one file, android.jar, which contains all the class libraries needed for an Android application.
- gen: Contains the R.java fi le, a compiler-generated file that references all the resources found in your project. You should not modify this file.
- assets: This folder contains all the assets used by your application, such as HTML, text files, databases, etc.
- res: This folder contains all the resources used in your application. It also contains a few other subfolders: drawable-<*resolution*>, layout, and values.
- AndroidManifest.xml: This is the manifest file for your Android application. Here you specify the permissions needed by your application, as well as other features (such as intent-filters, receivers, etc.).

Chapter3:

Image Capture

The Android framework includes support for various cameras and camera features available on devices, allowing you to capture pictures and videos in your applications.

Considerations

Before enabling your application to use cameras on Android devices, you should consider a few questions about how your app intends to use this hardware feature.

Camera Requirement - Is the use of a camera so important to your application that you do not want your application installed on a device that does not have a camera? If so, you should declare the camera requirement in your manifest.

Storage - Are the images or videos your application generates intended to be only visible to your application or shared so that other applications such as Gallery or other media and social apps can use them? Do you want the pictures and videos to be available even if your application is uninstalled?

The Basics

The Android framework supports capturing images and video through the Camera API or camera Intent. Here are the relevant classes:

Camera:

This class is the primary API for controlling device cameras. This class is used to take pictures or videos when you are building a camera application.

SurfaceView:

This class is used to present a live camera preview to the user.

Our project depend on captured images by camera that contain words need to be translated.

Android SDK provides a full control on Android devices' hardware, so that you can use any component of your device in your applications. Thanks to this feature, you can use your device's camera to obtain live camera preview in your application. So that, you can build your own camera application with additional features.

Basically in a camera application, you need a camera surface and a button to take photos. First we start with getting required permission for Camera component. Also put a control into the activity tag to keep screen in landscape mode.

```
<uses-permission android:name = "android.permission.CAMERA"/>
 <activity .......
    android:screenOrientation="landscape" ...... >
```

Then we create a simple layout includes a SurfaceView and a Button.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal" >
    <SurfaceView
        android:id="@+id/preview"
        android:layout_width=" wrap_content "
        android:layout_height="fill_parent" >
    </SurfaceView>
    <Button
        android:id="@+id/buttonTakePhoto"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center"
        android:visibility="invisible" >
    </Button>
</LinearLayout>
```

Steps of camera

- Then we create our Camera, Surface and Button fields which will be initialized when our activity is started.
- After this fields we need to declare which actions will be performed when user takes a photo:

  Saving the captured images at SDcard.
- Now we can implement the onCreate method of the Activity to initialize our surface and Button.
- Then we must provide some methods for our surface to create first preview, change preview, and stop preview.

Challenges

- This is a simple camera application, we do to capture images with a button and that what we want, so we made the button be invisible as

```
android:layout_width="0dp"
android:visibility="invisible"
```

and make SurfaceView take its size but **how to click it to start capture?**

We use a method `performClick()` to do the action of clicking.

- Camera capture one image at a time and we need a continuous stream of images to update the positions of words, so we do a new thread for one goal “capture images”.

- The captured images are not clear, contains a much noise.

  So we apply the Autofocus for every captured image and that reduced noise.

Chapter4:

Image Enhancement

In this chapter we will talk about some operations that must be performed on the image to make it closer to typical form before performing any OCR's operations on it.

- The captured image contains noise and we reduced it by using Autofocus but the result is not empty of noise, so we applied two filters of Camera:

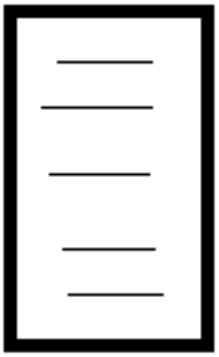
  First, make borders for white and black colors.

```
setWhiteBalance()
```

  Second, make borders for every color at the image.

```
setColorEffect()
```

- We assume that the paper that contains the text that is required to be translated has a black border as the following:



- We want to convert the RGB image to binary image. So, we want to divide the RGB color's range into two sets (black range and white range). The value of the white color in RGB image is "-1" and the value of the black color in RGB image is

  "-16,777,216". So, we will divide the RGB color's range with 3:7 ratio, where "3" for black range and "7" for white range. Obviously, we give the white color the higher priority by giving it the biggest range. We do this to remove any noise from the background wall that the paper is sticked on and to convert the background wall's color to white color to make the process of determining the paper's black border easier, but the text color will be blurred. We will treat the problem of blurred text later.

Dealing with the background wall's color:

Now, we will colorize the area that surrounds the paper's black border with the black color as the following:

It is obviously that the paper that contains the text doesn't have a rectangular shape, but it is wrong. This phenomenon appears according to the third dimension (z-axis) in our life and we will treat it later.

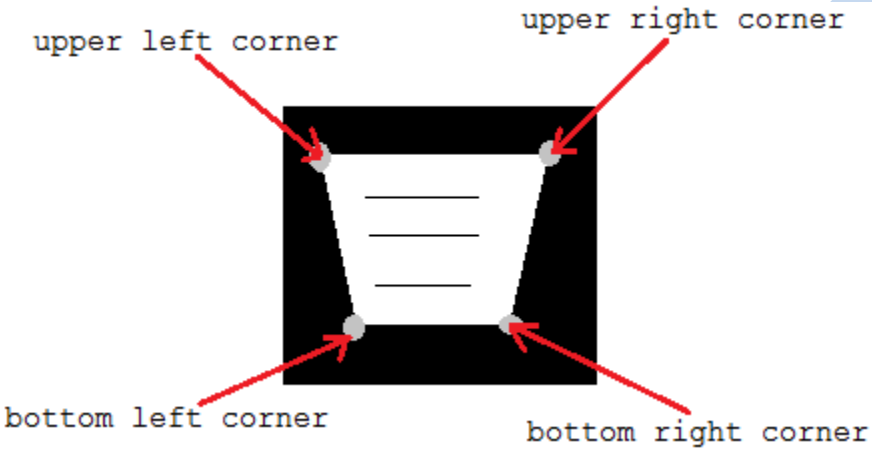
Treating the blurred text problem:

Then, we begin to treat the blurred text problem by generate a new image, and colorize the pixels which we colorized them with the black color previously (background wall and the black border) with the actual RGB's black color "-16,777,216". After that, we will colorize the other pixels with their original RGB colors in the original captured RGB image. Then, we will convert this new RGB image into binary image again by dividing the range of the RGB colors that are contained in this new image into two sets (black range and white range) by 9:11 ratio, where "11" for black range and "9" for white range. Obviously, we give the black color the higher priority by giving it the biggest range. We do this to make the text clear.

Note that: to convert this new RGB image into binary image we will work on the RGB colors that are contained in this new RGB image, don't work on the whole RGB colors' range to give accurate results. The following operations will be performed on this new binary image.

Determining the paper's four corners:

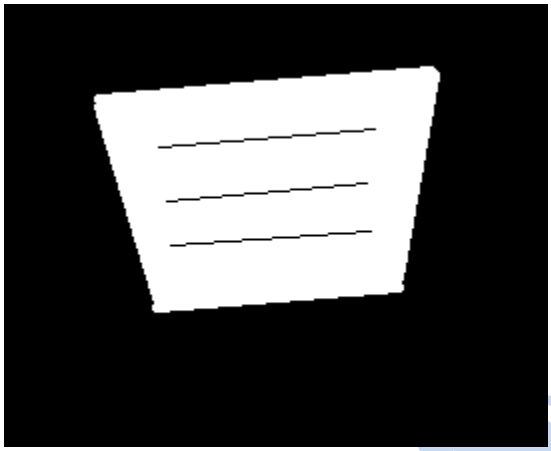
Now, we want to determine the four corners (upper left corner, upper right corner, bottom left corner and bottom right corner) of the paper that contains the text, but it is a hard task because the paper doesn't have a rectangular shape or any regular geometric shape in the captured image.



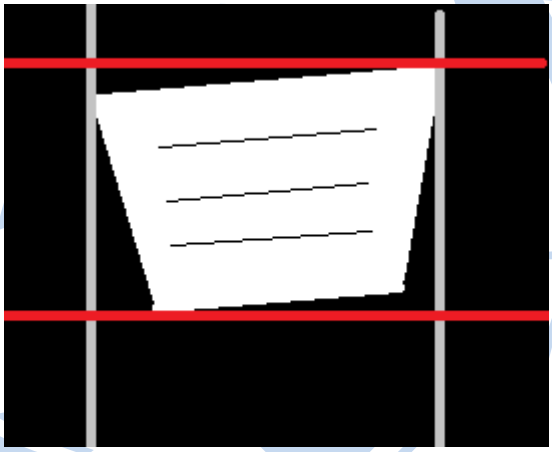
But we invent an algorithm to solve this problem. We will scan the final binary image vertically from left to right until we find at least one white pixel, then we set this column as left border. Similarly, we will scan the final binary image vertically from right to left until we find at least one white pixel, then we set this column as right border. Also, we will scan the final binary image horizontally from up to down until we find at least one white pixel, then we set this row as upper border. Similarly, we will scan the final binary image horizontally from down to up until we find at least one black pixel, then we set this row as bottom border. Then we will crop the area between these four borders (left border, right border, upper border and bottom border).

For example:

If we have this image:

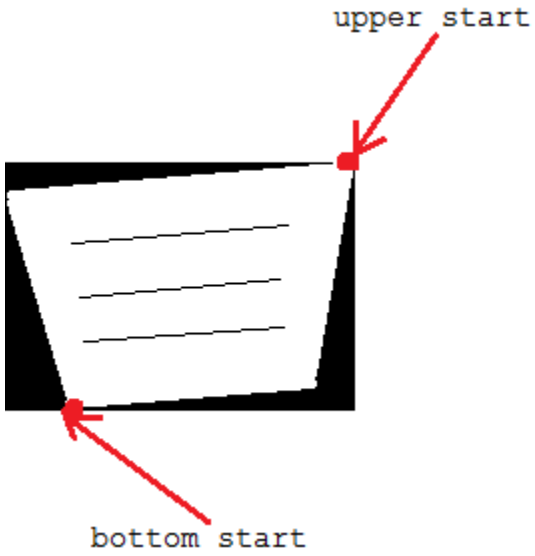


- Determine the four borders as the following:



- Crop the area between these four borders as the following:

Now, we start to determine the upper start point which is used as a start point to determine the upper left corner and the upper right corner. Also, we want to determine the bottom start point which is used as a start point to determine the bottom left corner and the bottom right corner.



Upper start point: is the white point which touches the upper edge of the image.

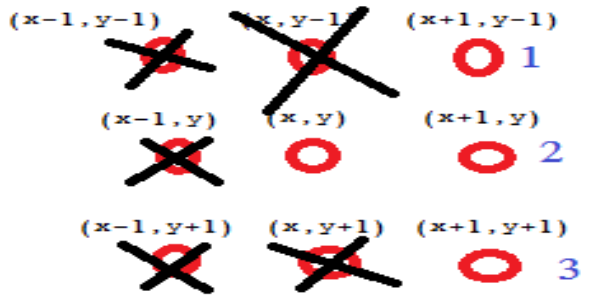
Bottom start point: is the white point which touches the bottom edge of the image.

We can determine the upper start point by scanning the image vertically starting with the image's upper edge and counting the number of black pixels until we reach a white pixel. We stop if the number of black pixels in any column is equal to zero, which means that the white pixel touches the image's upper edge. Similarly, we can determine the bottom start point by scanning the image vertically starting with the image's bottom edge and counting the number of black pixels until we reach a white pixel. We stop if the number of black pixels in any column is equal to zero, which means that the white pixel touches the image's bottom edge.

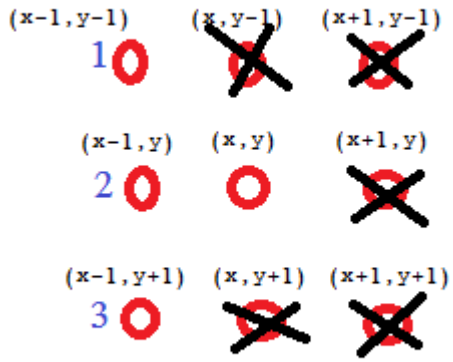
Now, we will start to determine the four corners (upper left corner, upper right corner, bottom left corner and bottom right corner) of the white paper that contains the text. To determine the upper right corner, we will go on white pixels in the right direction starting with the upper start point. There are three points that must be tested if there are white or not. The point number "1" has the highest priority which means that this point must be tested first. The point number "2" has a Medium priority which means that this point must be tested second. The point number "3" has lowest priority which means that this point must be tested last. If a one point of these points is white then we will stop testing the other remaining points and set this point as a current point and begin testing the next three points from this current point until there is no point of these three points is white, then set the current point as the upper right corner. Similarly, if we want to determine the upper left corner, but starting with the upper start point and go in the left direction.

Also, if we want to determine the bottom right corner, we starting with the bottom start point and go in the right direction. Similarly, if we want to determine the bottom left corner, we starting with the bottom start point and go in the left direction.

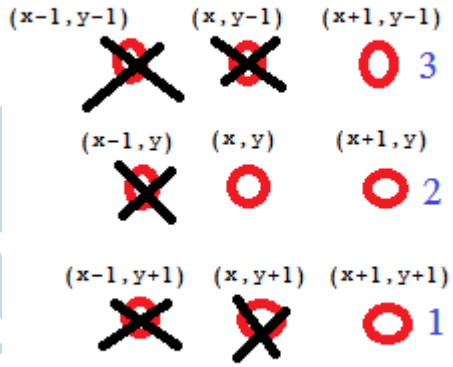
The three points that must be checked when determining the upper right corner are derived from the eight neighbors points of the current point as the following:



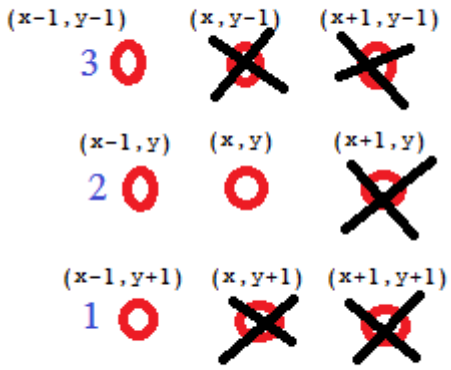
The three points that must be checked when determining the upper left corner are derived from the eight neighbors points of the current point as the following:



The three points that must be checked when determining the bottom right corner are derived from the eight neighbors points of the current point as the following:



The three points that must be checked when determining the bottom left corner are derived from the eight neighbors' points of the current point as the following:

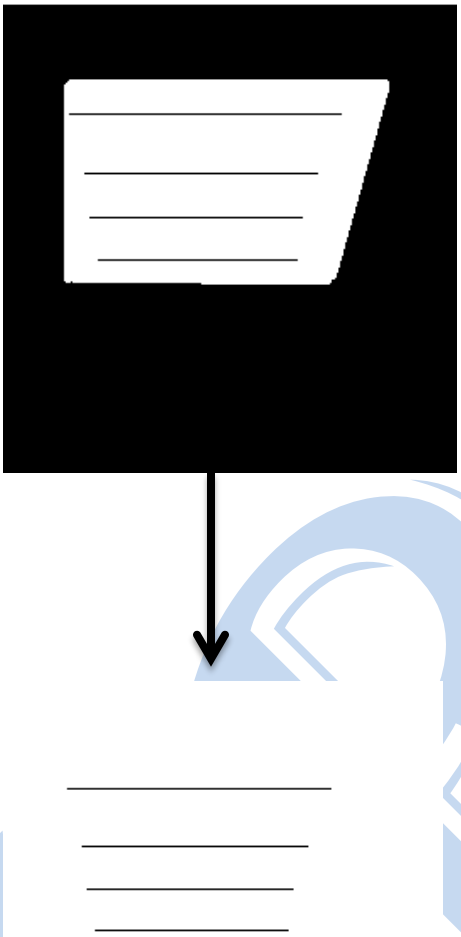


Adjust the paper's slope:

Now, we want to adjust the paper's slope. So, we will check the upper left corner and upper right corner. If upper left corner is higher than the upper right corner, then the image is rotated in the clock direction. If the upper right corner is higher than the upper left corner, then the image is rotated in anti-clock wise direction. If the height of the upper left corner is equal to the height of the upper right corner, then the image is in the typical form. To adjust the image, you can calculate the sloped angle and rotate the image in the reverse direction by the same angle, but it doesn't give accurate results. So, I prefer to make this rotation manually by shifting the image's pixels in the vertical direction until the upper right corner becomes on the upper left corner alignment. Then, start to shift the image's pixels in the horizontal direction until the upper right corner becomes on the bottom right corner alignment if the image is sloped in the clock direction or until the upper left corner becomes on the bottom left corner alignment if the image is sloped in anti-clock wise direction.

Treating the problem of the third dimension:

Now, we will treat the problem of the third dimension by replace all black pixels that surround the white paper that contains the text with white pixels to convert the shape of the paper to rectangular shape as the following:



After that we will move to the segmentation algorithm.

Chapter5:

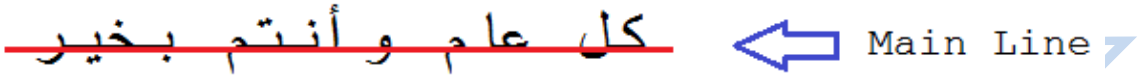
Image Segmentation

Introduction:

In this chapter, we will talk about our own algorithms of Arabic and English Segmentation.

Arabic Segmentation

First, we will talk about our own algorithm of Arabic segmentation. Our idea of segmentation is very simple. It depends on the properties of the Arabic language, where each writing line has main line where each letter in this writing line is connected to these main line.



So, we depend on this basis. First, we will state the steps of our algorithm quickly, and then, we will explain each step in details.

Arabic Segmentation steps:

1- Determine the upper start of the writing line.

2- Determine the bottom end of the writing line.

3- Calculate the width of the main line.

4- Isolate each word in the line.

5- Determine the main line for each word.

6- Extract each character.

Determine the upper start of the writing line:

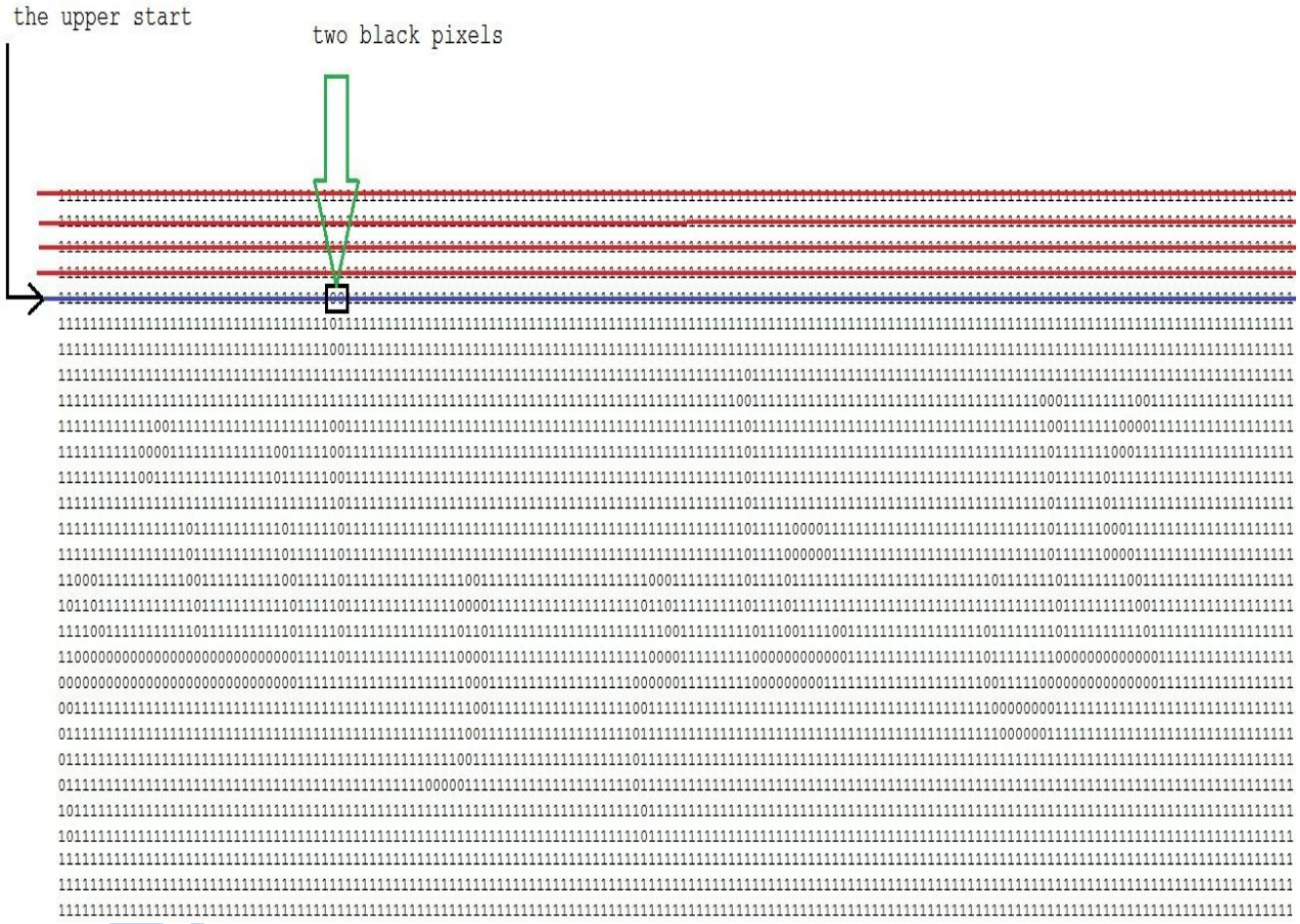
To determine the upper start of the writing line, we will scan the image horizontally starting with the first image line of pixels until we find at least one black pixel, and then we set this line as the upper start of the writing line.

For example: if we have this image:

كل عام و أنتم بخير

رمضان كريم

Then the corresponding binary image is (we will take a part of the binary image):



2- Determine the bottom end of the writing line:

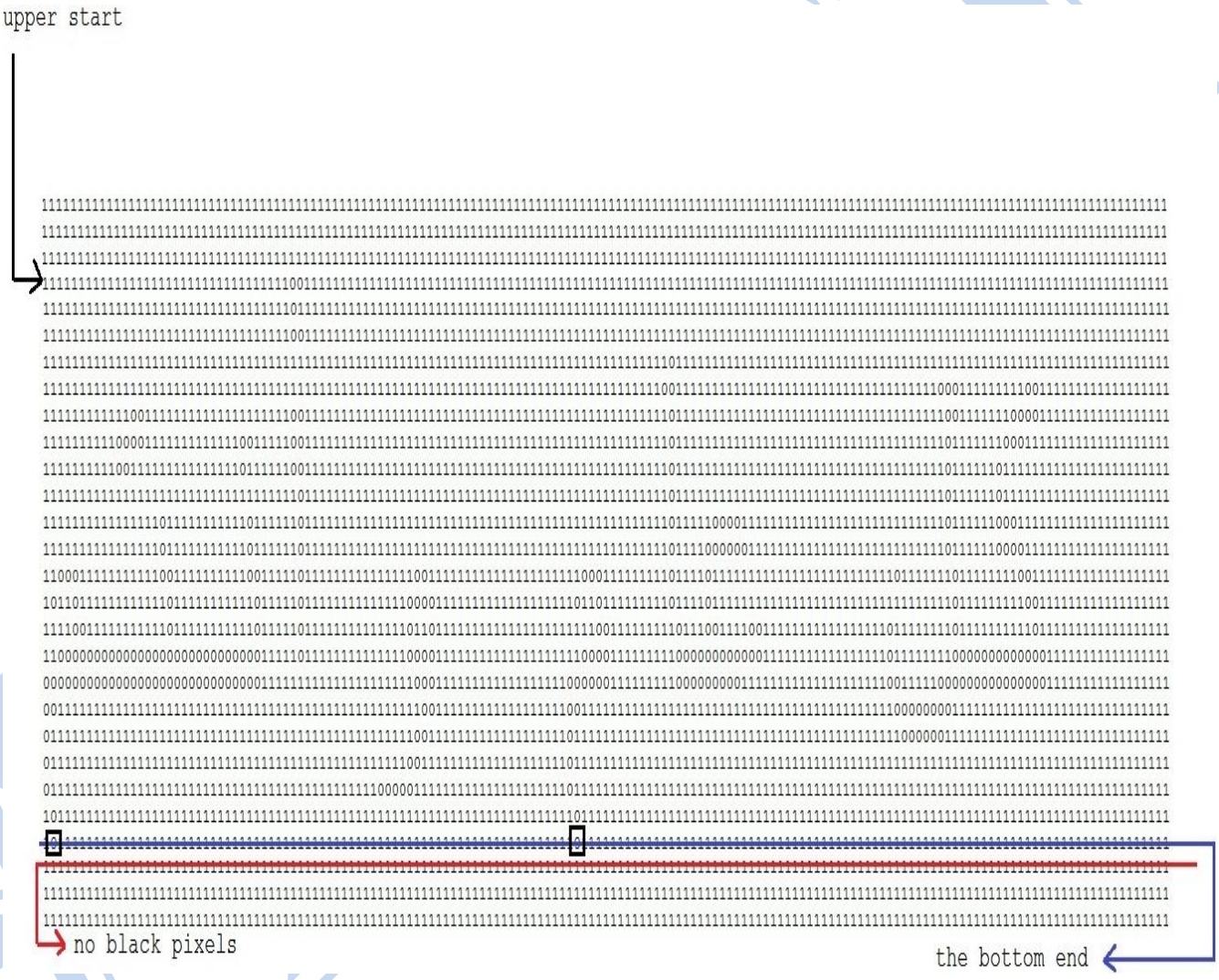
To determine the bottom end of the writing line, we will scan the image horizontally starting with the current upper line + 7 (to skip any dots above the letters, and the number 7 is a constant, and we determine it by trial) until we find no black pixels, and then, we set the line that is located above this line as the temporary bottom end of the writing line, and then, calculate the width of the main line (will come later), then resume the scanning for the bottom end of the writing line starting with the temporary bottom end + main line's width + 4 (to skip any dots under the letters, and this equation is formulated by trial) until we find no black pixels, and then, we set the line that is located above this line as the bottom end of the writing line.

For example: if we have this image:

كل عام و أنتم بخير

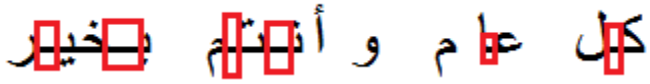
رمضان كريم

Then the corresponding binary image is (we will take a part of the binary image):



3- Calculate the width of the main line:

We calculate the width of the main line before completing the process of determining the bottom end of the writing line because we don't need to enter the dots under the letters in our account to get more accurate results. We scan the image vertically starting with the upper start of the writing line to the temporary bottom end of the writing line and calculate the number of black pixels in each column, and put the number of black in each column in a list, and then search for the most frequently number in this list, and set it as the width of the main line.



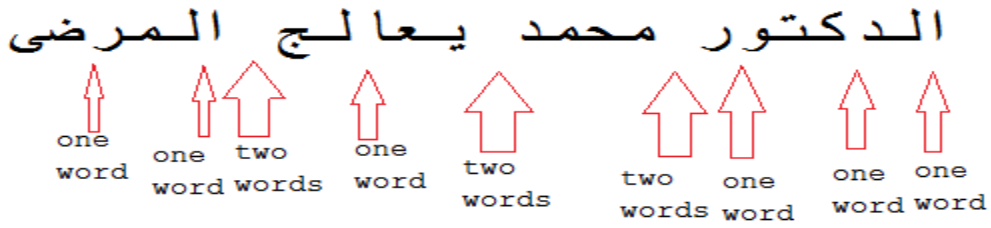
These marked regions have the same number of vertical black pixels. These regions will contribute in the process of determining the width of the main line.

4- Isolate each word in the line

To isolate each word in the line, we will search about the spaces between the words by calculating the number of white pixels between the absence of the black pixels and the appearance of the black pixels again in the same line.

If the number of white pixels between the two groups of black pixels is larger than ((2 * the width of the main line) + 11), this equation is determined by trial to determine the space length between the words according to the size of the font, then it indicate that there is two words separated by a space. This equation is formulated to differentiate between the spaces between two words and the space between one letter and another one of the same word.

For example:

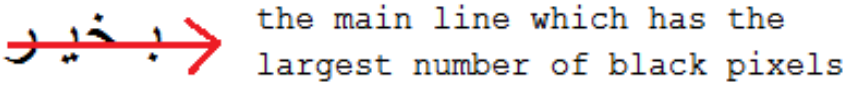


5- Determine the main line for each word:

To determine the main line for each word, we scan the word horizontally and count the number of black pixels in each row, and put this number in a list, and then search in this list for largest values (the number of these values is equal to the width of the main line) of black pixels. Then take the line corresponding to the first value (in order not for value) of these large vales as the main line, since the main line has the largest number of black pixels.

Note: the main line isn't a single line, but it can be a group of lines, so, we want to determine the upper line of these lines.

For example:



But it isn't the final main line. Because of the noise that has been added by the imaging system and the noise that is generated because of hand sloping or hand shake during image capturing, the main line has a need to be recalculated at every column during extracting each letter (will be explained in the next step).

6- Extract each character:

- In each time, before performing any calculations to extract a single character, we will recalculate the main line in the current column according to the writing line.

Note: the main line must be continuous.

According to the previous note, we have "(2 * the width of the main line) + 1" possible positions of the new main line as the following:

If we have a main line of width "2", then we have" 5" possible positions of the new main line as the following:

1-

```
0000000000
0000000000
```

2-

```
00000
0000000000
     00000
```

3-

```
00000
00000
     00000
     00000
```

4-

```
     00000
0000000000
00000
```

5-

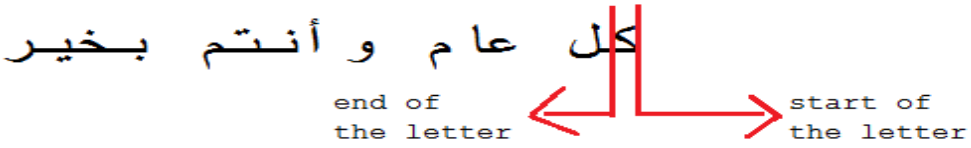
```
     00000
     00000
00000
00000
```

So, we can check these positions to recalculate the main line. If we use an image that is generated by the computer that is in typical form, we can skip this step.

Then, we will make a small filter operation for thinning the main line, because there may be a noise around the main line according to hand sloping or hand shake during image capturing or according to the resolution of the camera that is used for capturing the image. If there are no black pixels above the main line except the one pixel that is exactly located above the main line, we consider is as a noise and we will colorize this pixel by the white color. If we use an image that is generated by the computer that is in typical form, we can skip this step.

- Then, we start to extract each character by counting the number of black pixels that is located above the main line and the number of black pixels that is located under the "main line + the width of the main line – 1" (note: this equation to find the black pixels under the main line group, since the main line isn't a single line) starting with the right side of the writing line. If the number of pixels above the main line or the number of pixels under the main line is greater than "0", then it is indicate that there is an emergence of the main line which mean that there is a start of new letter. Then, we resume counting the number of black pixels above and below the main line until the number of black pixels above and under the main line is equal to zero, which indicate to the end of this letter. Now, the start and the end of the letter are determined and we can split this letter from the writing line. Then we resize the image of each letter to 16 * 16 images, which is useful in detection.

For example:



But there are some special cases with some letters such as:

-"س"

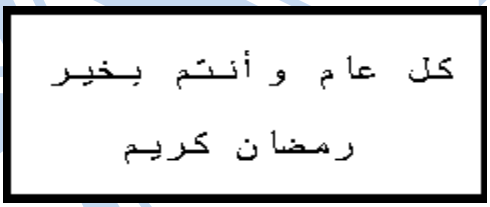
Because it consists of three sections and my algorithm will isolate each section separately as follows:

- And the letter "ب" that consists of three sections as follows:

And any other letter that consists of more than one section has the same problem.

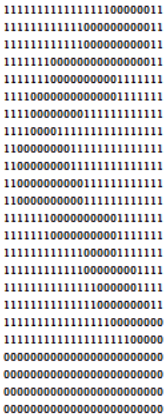
To solve this problem, we can train our detection algorithm to recognize each section of the letters that consists of more than one section and put these sections in one letter and then begin to recognize the original letter again.

After applying our segmentation algorithm on this image:

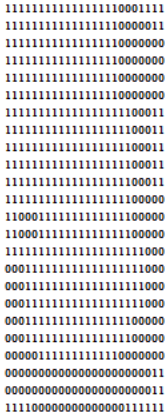


We have this result:

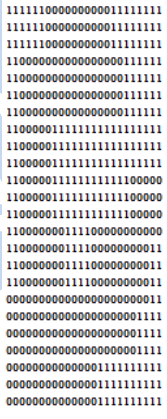
1- كـ



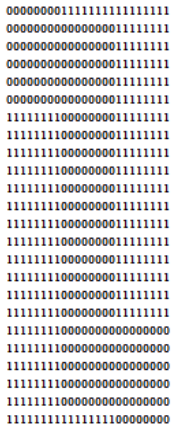
2- ل



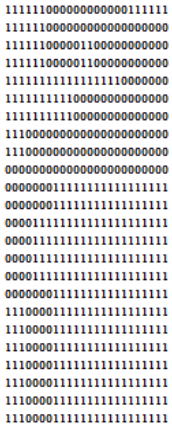
3- عـ



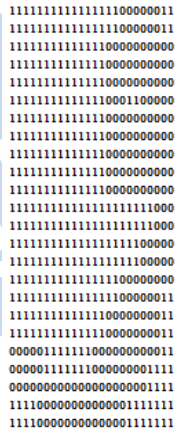
4- ا



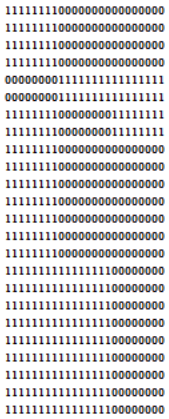
5- م



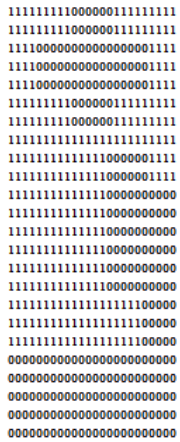
6- و



7- أ



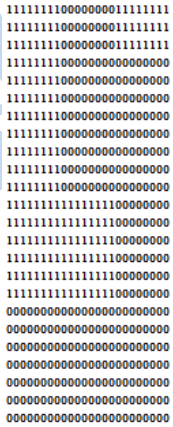
8- ـنـ



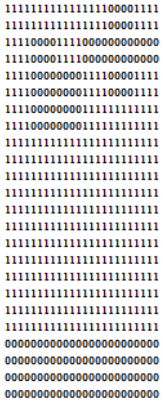
9- ـتـ

This letter consists of two sections:

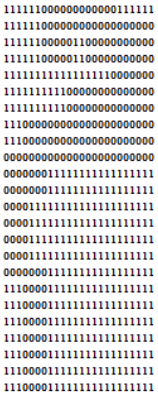
The start tooth of the letter:



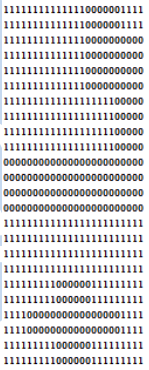
The upper dots of the letter:



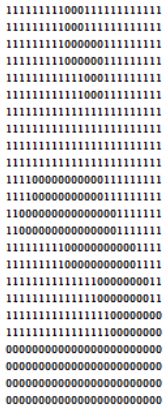
10- م



11- بـ

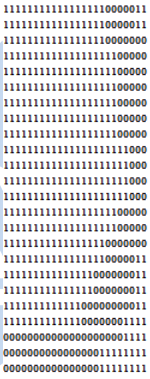


12- خ



13- پ

14- ر



15- ر

```
11111111111111110000011
11111111111111110000011
11111111111111110000000
11111111111111111100000
11111111111111111100000
11111111111111111100000
11111111111111111100000
11111111111111111100000
11111111111111111100000
11111111111111111111000
11111111111111111111000
11111111111111111111000
11111111111111111111000
11111111111111111100000
11111111111111111100000
11111111111111110000000
11111111111111110000011
11111111111111000000011
11111111111111000000011
11111111111110000000011
11111111111100000001111
00000000000000000001111
00000000000000000011111
00000000000000000011111
```

16- مـ

```
11111111111111000111111
11111111111111000111111
11111111111111000111111
11111111111111000111111
11111111100000000000111
11111111100000000000111
11111111100000000000111
11111111100000000000111
11111110000000000111000000
11111110000000000111000000
11111110000000000111000000
11111110000000000111000000
11100000000000000111000000
11100000000000000111000000
11100000000000000111000000
11100000000000000111000000
00000000000000000000000000
00000000000000000000000000
00000000000000000000000000
00000000000000000000000000
00000011111110000000000000
00000011111110000000000000
00000011111110000000000000
00000011111110000000000000
```

17- ضـ

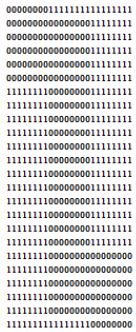
This letter consists of two sections:

Curve of the letter:

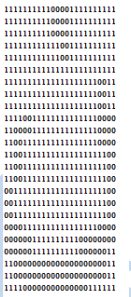
```
11110000001111111111111
11110000001111111111111
11110000001111111111111
11110000001111111111111
11110000001111111111111
11111110001111111111111
11111110001111111111111
11111111111111111111111
11111111111111111111111
11111111111111111111111
11111111111100000000011
11111111111100000000011
11111111111100000000011
11111111110000000000000
11111111110000000000000
11110000000000000000000
11110000001111110000000
11000000001111110000000
11000000111111111100000
00000000000000000000000
00000000000000000000000
00000000000000000000000
00000000000000000000000
00000000000000000000000
```

The tooth of the letter:

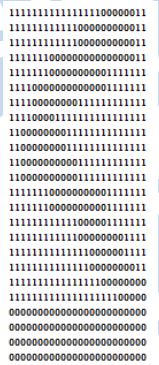
18- ا



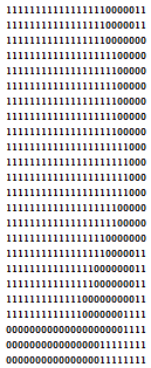
19- ن



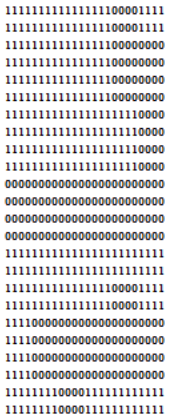
20- ـكـ



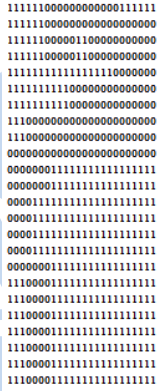
21- ر



22- ږ



23- م



English Segmentation:

Second, we will talk about our own algorithm of English segmentation. Our idea of segmentation is very simple. It depends on the on the properties of the English language, where there are spaces between the letters. English segmentation has the same of almost steps of the Arabic segmentation, but with small different details.

English Segmentation steps:

1- Determine the upper start of the writing line.

2- Determine the bottom end of the writing line.

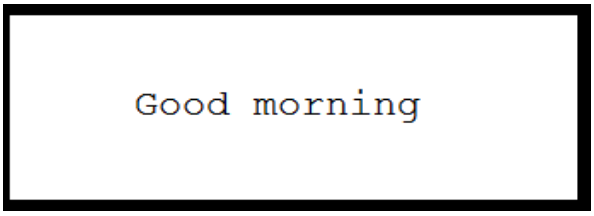
3- Isolate each word in the line.

4- Extract each character.

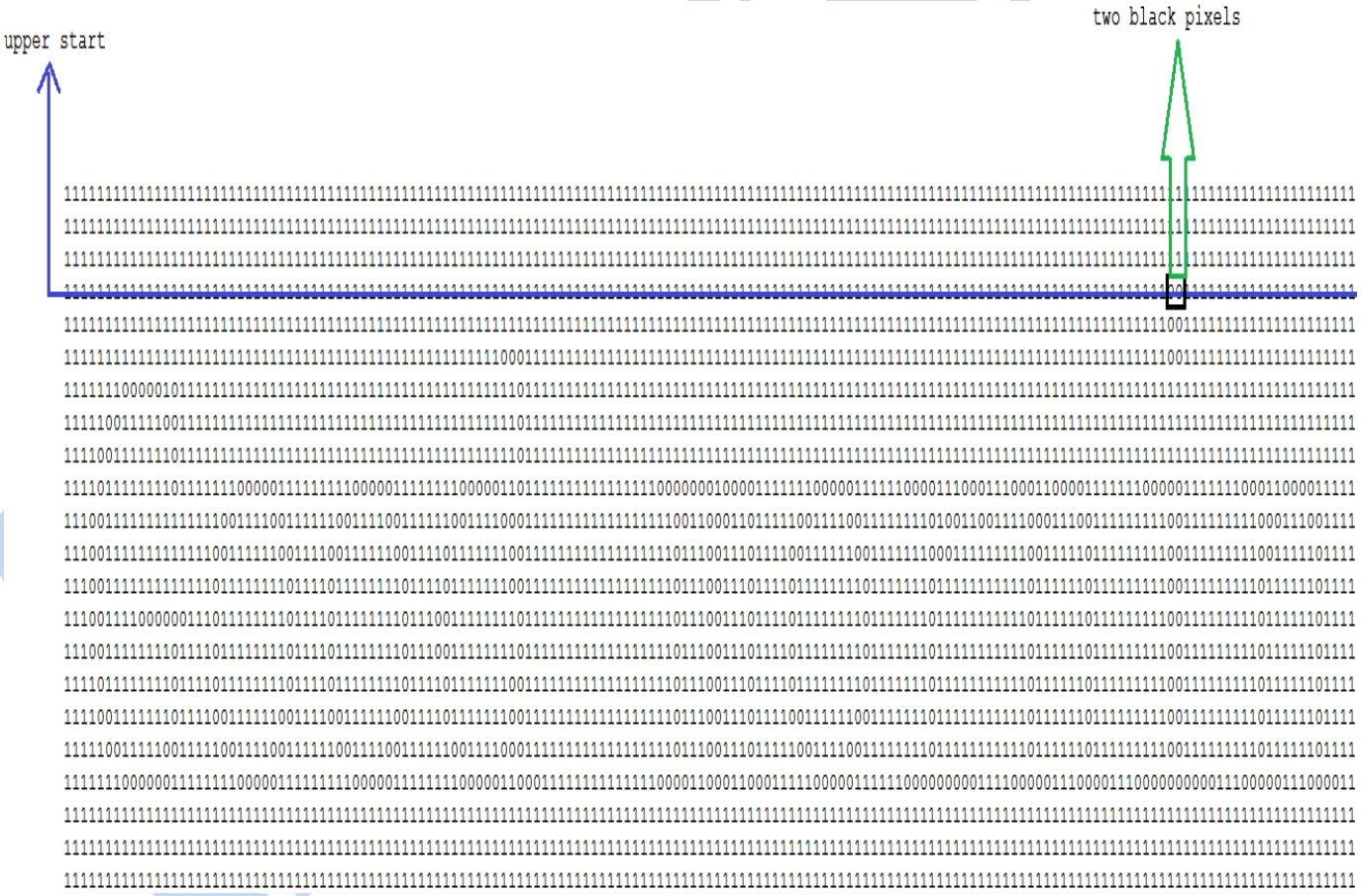
1- Determine the upper start of the writing line:

Previously explained in Arabic segmentation.

For example: if we have this image:



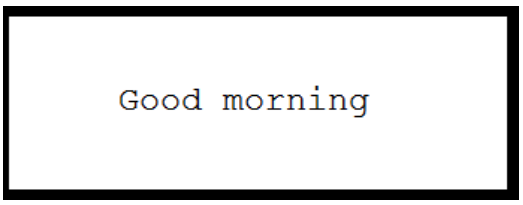
Then the corresponding binary image is (we will take a part of the binary image):



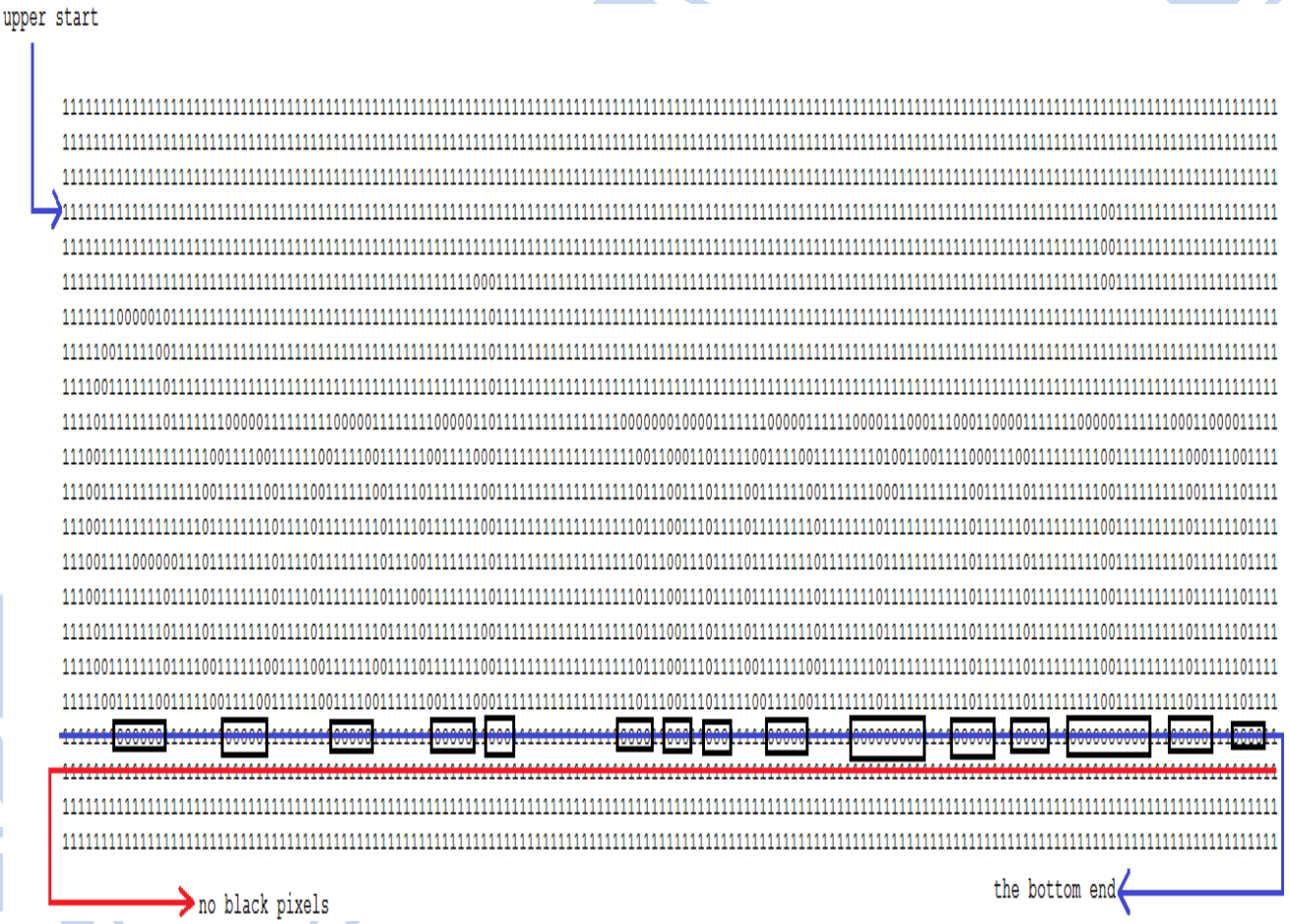
2- Determine the bottom end of the writing line:

Previously explained in Arabic segmentation.

For example: if we have this image:



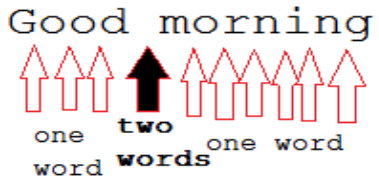
Then the corresponding binary image is (we will take a part of the binary image):



3- Isolate each word in the line:

Previously explained in Arabic segmentation, but the space length between two words is larger than "10".This constant is determined by trial.

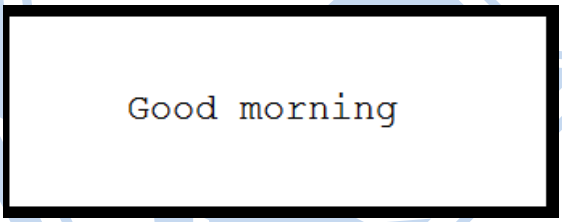
For example:



4- Extract each character:

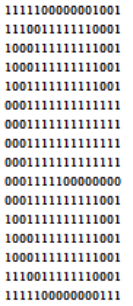
To extract each letter, we will scan the image vertically until we find at least one black pixel, which indicates to the start of the letter. Then, we continuo scanning until we find no black pixels, which indicate to the end of the letter. Now, we have the start and the end of the letter and we can split this letter from the writing line. Then we resize the image of each letter to 16 * 16 images, which is useful in detection.

After applying our segmentation algorithm on this image:

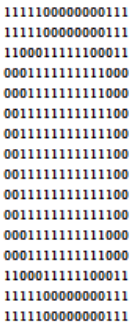


We have this result:

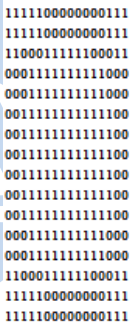
1- G



2- o



3- o



4- d

```
1111111111000011
1111111111110011
1111111111110011
1111111111110011
1111111111110011
1111000000110011
1100011111000011
1011111111100011
1011111111100011
0011111111110011
0011111111110011
1011111111100011
1011111111100011
1011111111100011
1100011111000011
1111000000110000
```

5- m

```
0000000010000011
0000000010000011
1100011000111011
1101111001111011
1101111001111011
1101111001111011
1101111001111011
1101111001111011
1101111001111011
1101111001111011
1101111001111011
1101111001111011
0000011000111000
0000011000111000
```

6- o

```
1111100000000111
1111100000000111
1100011111100011
0001111111111000
0001111111111000
0011111111111100
0011111111111100
0011111111111100
0011111111111100
0011111111111100
0011111111111100
0001111111111000
0001111111111000
1100011111100011
1111100000000111
1111100000000111
```

7- r

```
0000001111000001
0000001111000001
1111001000111000
1111000001111111
1111000001111111
1111001111111111
1111001111111111
1111001111111111
1111001111111111
1111001111111111
1111001111111111
1111001111111111
1111001111111111
0000000000000111
0000000000000111
```

8- n

```
0000111000001111
0000111000001111
1110000111100111
1110011111110111
1110011111110111
1110111111110111
1110111111110111
1110111111110111
1110111111110111
1110111111110111
1110111111110111
1110111111110111
0000000111100000
0000000111100000
```

9- i

```
1111110000111111
1111110000111111
1111110000111111
1111111111111111
1111111111111111
1111111111111111
1100000000111111
1111110000111111
1111110000111111
1111110000111111
1111110000111111
1111110000111111
1111110000111111
1111110000111111
0000000000000000
```

10- n

```
0000111000001111
0000111000001111
1110000111100111
1110011111110111
1110011111110111
1110111111110111
1110111111110111
1110111111110111
1110111111110111
1110111111110111
1110111111110111
1110111111110111
0000000111100000
0000000111100000
```

11- g

```
1111000000010000
1110111110000111
1001111111100111
1001111111110111
0001111111110111
0001111111110111
1001111111110111
1001111111100111
1001111111100111
1110111110000111
1111000000010111
1111111111110111
1111111111110111
1111111111110111
1111111111100111
1111000000011111
```

Chapter6:

Image Detection

In this chapter, we will talk about our own algorithm in detection. Our algorithm is very simple. The idea of our algorithm is depend on storing number of binary images for each character in all possible locations in the text.

Arabic detection:

For Arabic characters, we will store four binary images for each character, because each character has "4" possible positions in the text as the following:

1- Start of the word.

2- Middle of the word.

3- End of the word.

4- Isolated letter.

For example:

If we take the letter "س" as example:

Start of the word:

سـ

Middle of the word:

ـسـ

End of the word:

ـس

Isolated letter:

س

Another example: If we take the letter "م" as example:

Start of the word:

مـ

Middle of the word:

ـمـ

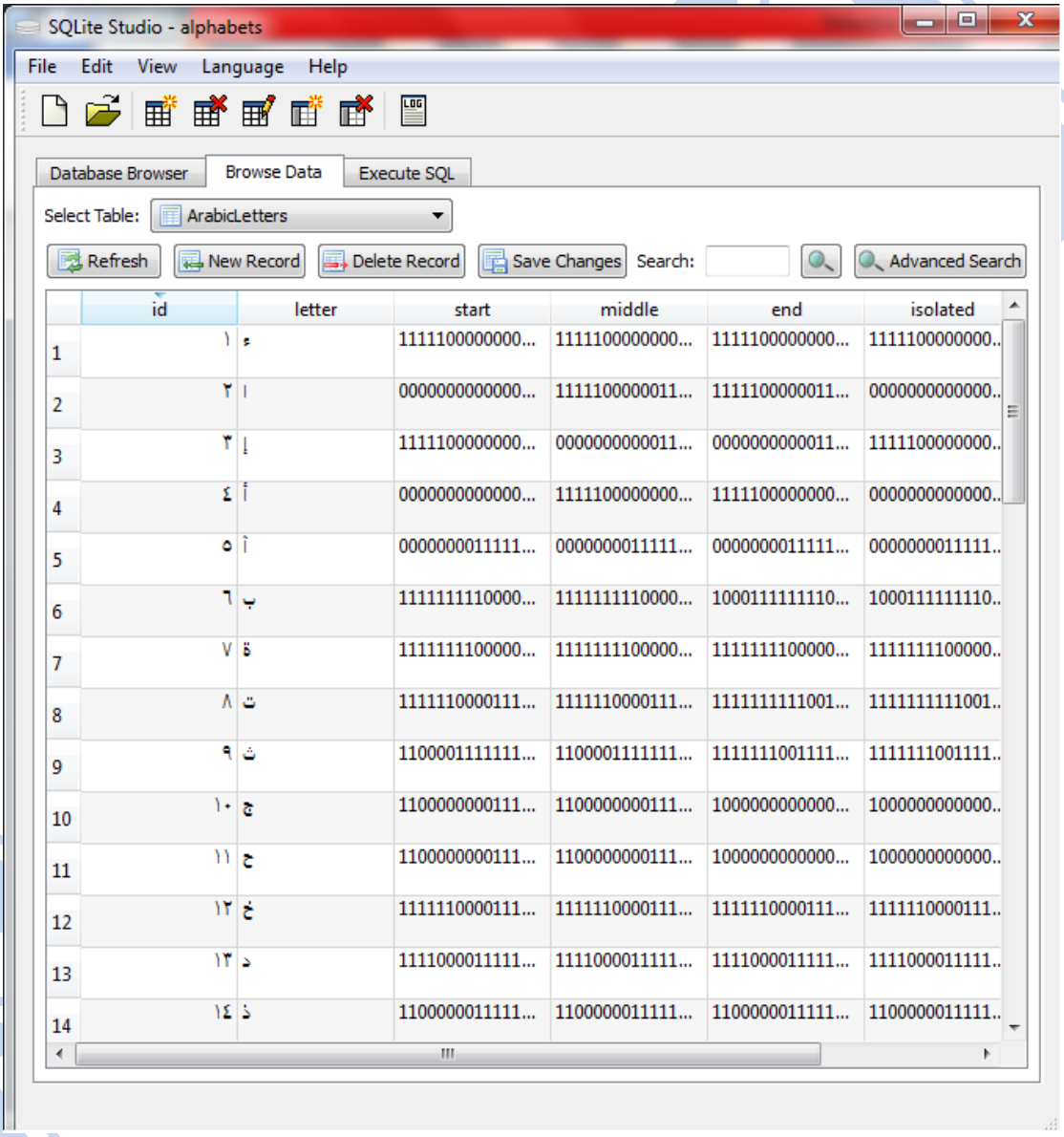
End of the word:

ـم

Isolated letter:

م

So, we create a table by the SQLite as the following:



This table consists of six columns as the following:

- The first column "id" is used as an index.
- The second column "letter" is used to retrieve the original letter after the recognition process.
- The third column "start" is used to store a 16 * 16 image for each letter when the letter is at the start of the word.
- The fourth column "middle" is used to store a 16 * 16 image for each letter when the letter is at the middle of the word.
- The fifth column "end" is used to store a 16 * 16 image for each letter when the letter is at the end of the word.
- The sixth column "isolated" is used to store a 16 * 16 image for each letter when the letter is isolated.

These 16 * 16 images are binary images that are consists of two colors "0 – black" and "1 – white". So, we convert these images to strings of length 256 each. Then, we extract each character by our own segmentation algorithm as a 16 * 16 binary image, and then convert this image to a string of length "256".Then; we generate a list with the same length of our database table. After that, we take the resulting string from our segmentation algorithm and compare it with the four cells (start, middle, end and isolated) of each row in the database table, and compute the percent of similarity with each cell. Then, we take the maximum percent of these four percent and store it in the corresponding cell in the list that we create. After finishing the comparison with all letters in the database table, we search in the list, which we create, about the maximum percent, and then take the letter that is corresponding to this percent as a recognized letter.

During database construction, if you find a letter that doesn't have the four states (start, middle, end, isolated), you can fill the empty cells with the isolated state but not leave it empty to avoid any exceptions.

For example:

The letter "و" has only to states (end and isolate) as the following:

End state:

ـو

Isolated state:

و

But it doesn't have "start" and "middle" state. So, we can put in the cells of "start" state and the "middle" state the same value as the cell of "isolated" state.

We can store in the database table some images of number of sections of the letters that consists of more than one section to inform the program that the current image that is retrieved from the segmentation algorithm doesn't represent actual letter. So, the program will combine the current image that is retrieved from the segmentation algorithm with the next image that is also retrieved from the segmentation algorithm until get an actual letter. We can store these sections in the database table as the following:

44	٤٤	-	1111100000011...	1111100000011...	1111100000011	1111100000011
45	٤٥	-	1111111000000...	1110000011111...	1110000011111	1110000011111
46	٤٦	-	1110000111111...	1100000000111...	1111100000011	1111100000011
47	٤٧	-	1111111100111...	1111001111111...	1111111110000	1111111110000

- The first column "id" is used as an index.
- The second column "letter" stores any special character such as "-" to inform the program that it isn't actual letter when retrieving the recognized letter.

- The third column "start", the fourth column "middle", the fifth column "end" and the sixth column "isolated" are used to store 16 * 16 images for number of sections of letters that is consists of more than one section.

English detection:

Similarly, we perform this process with English characters, but with a small different. Each English character has only two states instead of four states in Arabic characters. These two states are:

- Capital.

- Small.

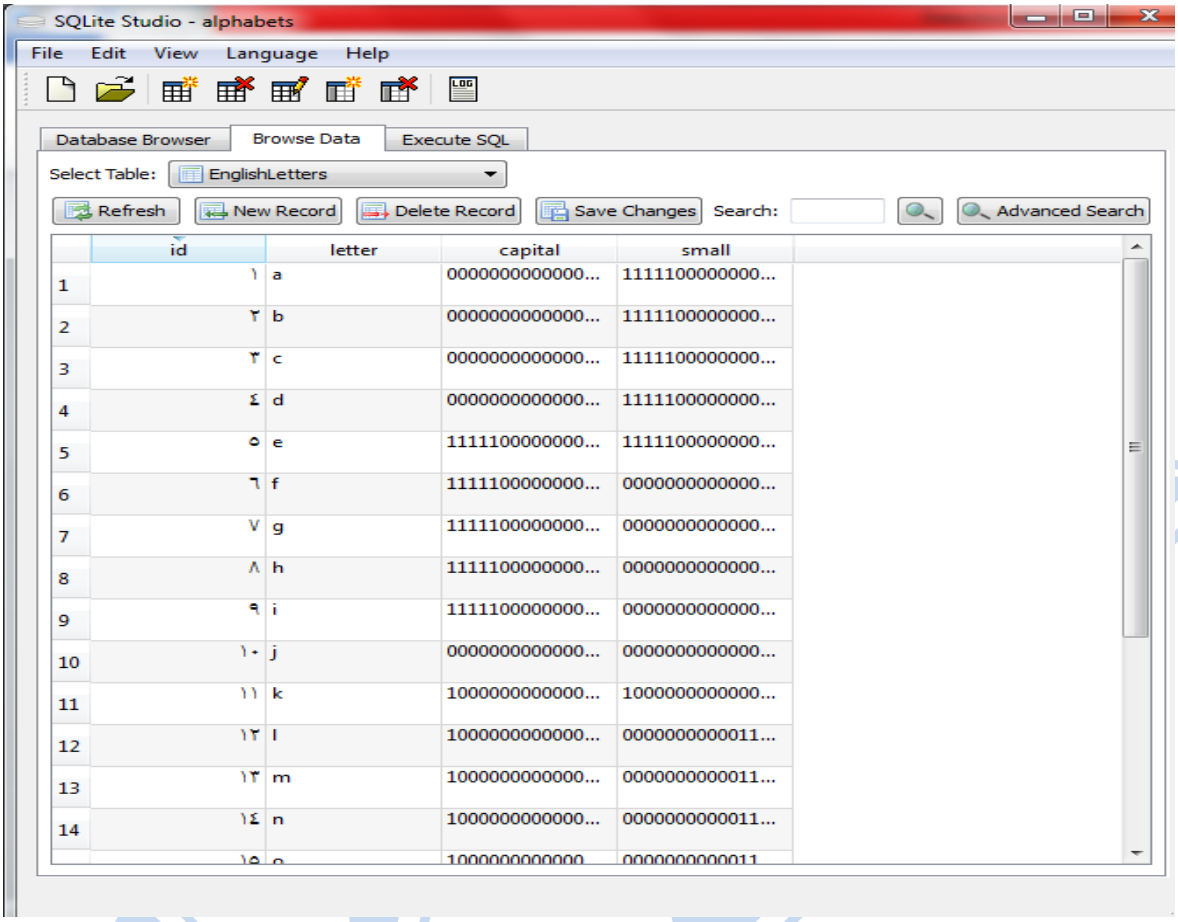
For example: if we take a letter "A" as an example:

It has two states as the following:

- Capital: "A"

- Small: "a"

So, the only difference between the English and Arabic detection is in database table, where we will create a database table with four columns instead of six columns in Arabic as the following:



This table consists of four columns as the following:

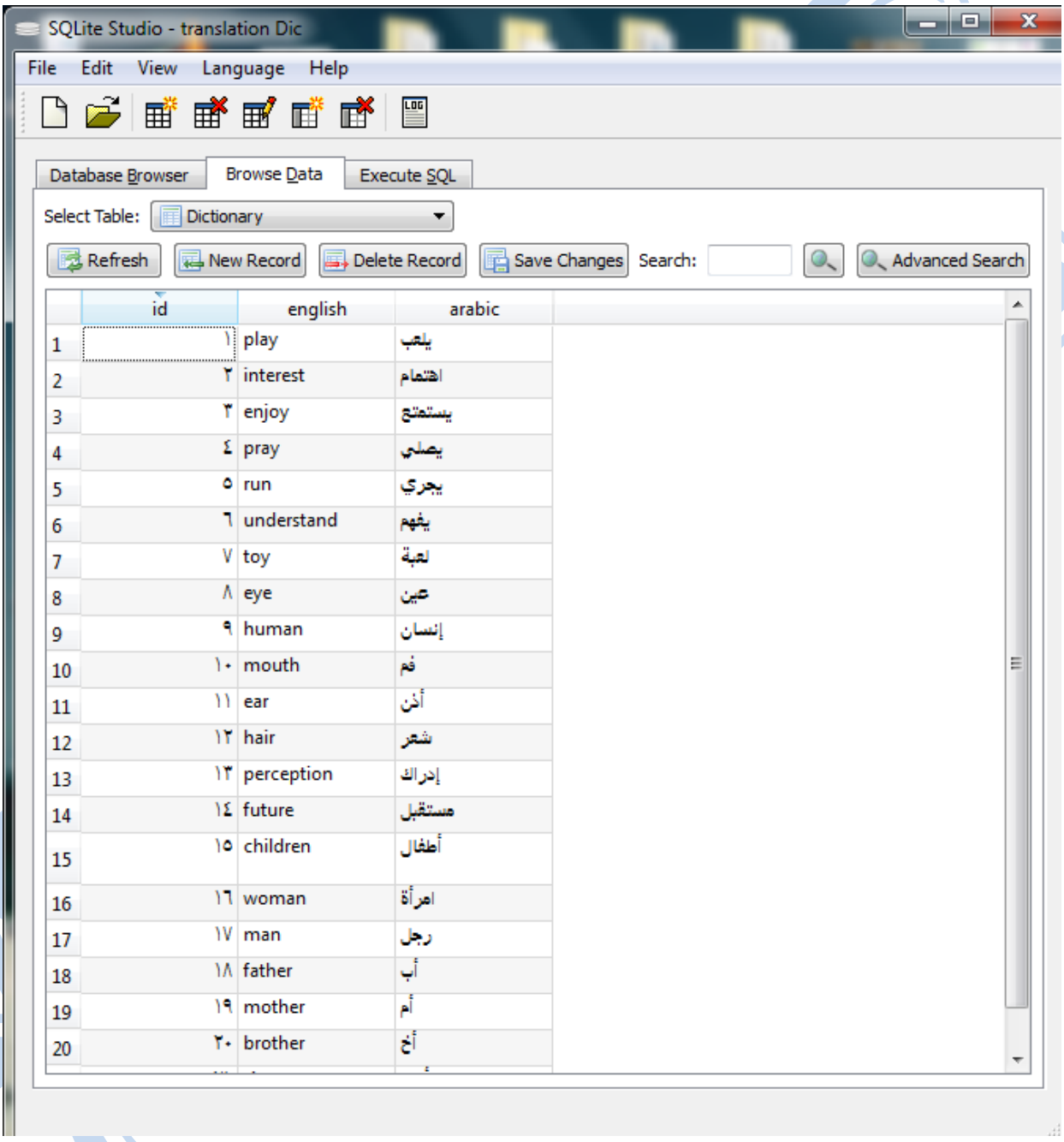
- The first column "id" is used as an index.
- The second column "letter" is used to retrieve the original letter after the recognition process.
- The third column "capital" is used to store a 16 * 16 image for each letter in the capital case.
- The fourth column "small" is used to store a 16 * 16 image for each letter in the small case.

  Then perform the same steps that are performed on Arabic letters to retrieve the original letter.

Chapter7:

Image Translation

In this chapter, we will talk about the translation process. Simply, we generate a small database table using SQLite that consists of some Arabic words and the corresponding English words as the following:



	id	english	arabic
1	١	play	يلعب
2	٢	interest	اهتمام
3	٣	enjoy	يستمتع
4	٤	pray	يصلي
5	٥	run	يجري
6	٦	understand	يفهم
7	٧	toy	لعبة
8	٨	eye	عين
9	٩	human	إنسان
10	١٠	mouth	فم
11	١١	ear	أذن
12	١٢	hair	شعر
13	١٣	perception	إدراك
14	١٤	future	مستقبل
15	١٥	children	أطفال
16	١٦	woman	امرأة
17	١٧	man	رجل
18	١٨	father	أب
19	١٩	mother	أم
20	٢٠	brother	أخ

Our database table consists of three columns as the following:

- The first column "id" is used as an index.

- The second column "english" contains number of English words.
- The third column "arabic" contains number of Arabic words.

To translate the recognized text:

- If the translation type is from Arabic to English, then we extract each word from the recognized text and Looking for it in the database table in the "arabic" column and retrieve the corresponding English word from "english" column as a translation.

- If the translation type is from English to Arabic, then we extract each word from the recognized text and Looking for it in the database table in the "english" column and retrieve the corresponding Arabic word from "arabic" column as a translation.

We can replace our database table with any other dictionary or we can use the online translation. So, the translation process isn't a problem.

Chapter8:

Display

Using of the water mark in our application:

We use the water mark in our application to write the translated text on image instead of the original text.

Steps of water mark:

1- We colorize the white paper that contains the original text in the original image with the white color to remove the original text.
2- Then we write each translated writing line on the original image at the same position of the original text (between the topline and the bottom line) using the water mark technique.

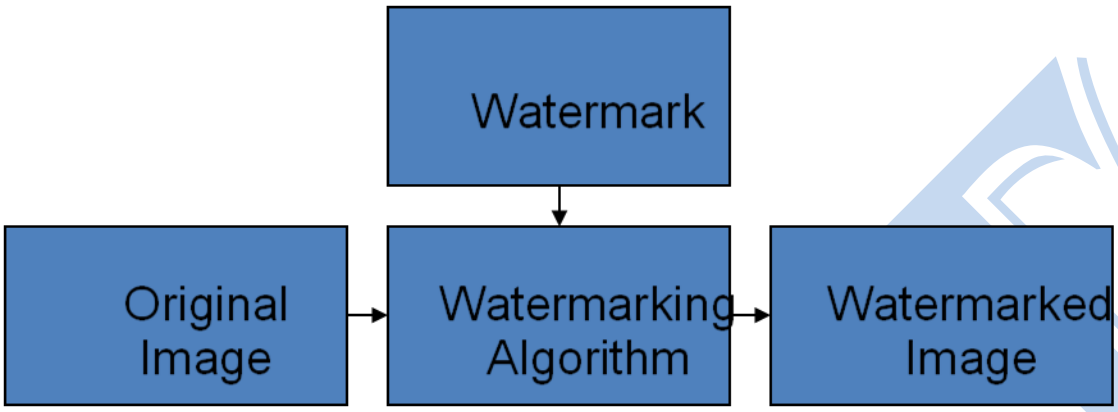
What is a water mark?

- A watermark is a recognizable image or pattern in image that appears as various shades of lightness/darkness when viewed by transmitted light
  Or when viewed by reflected light, atop a dark background caused by thickness or density variations in the image.
- Watermarking is a technique with which you can sign your name on the photo. It is a proof that you own the photo. Now anyone who tries to copy cannot do so because the watermark will be there in the photo.

The watermarked sign will show the rightful owner of the photo

It is typically used to identify ownership of the copyright for any person.

Watermarking is the process of hiding digital information in a carrier image



Block Diagram of image watermarking

The needed properties of a digital watermark depend on the use case in which it is applied

Digital watermarks may be used to

1. Verify the authenticity or integrity of the carrier image or to show the identity of its owners. It is prominently used for tracing copyright infringements and for authentication.

2. are only perceptible under certain conditions after using some algorithms

3. Does not change the size of the carrier image

Common water marking methods:

1. Spatial domain: modification made to the luminance values

2. Transformed domain: data compression techniques work well against most forms of image modification

Watermarks advantages

1. Watermarks vary greatly in their visibility; while some are obvious on casual inspection, others require some study to pick out. Various aids have been developed, such as watermark fluid that wets the paper without damaging it

2. Watermarks are often used as security features of banknotes, passports, postage stamps, and other documents to prevent counterfeiting (see security paper).

3. Watermarks is very useful in the examination of paper because it can be used for dating, identifying sizes, mill trademarks and locations, and the quality of a paper.

4. Encoding an identifying code into digitized music, video, picture or other file is known as a digital watermark.

Watermarking in spatial domain:

1. Computationally efficient and blind digital image watermarking in spatial domain. Embedded watermark is meaningful and recognizable and recovery process needs only one secret image.

2. Watermark insertion process exploits average brightness of the homogeneity regions of the cover image.

3. Spatial mask of suitable size is used to hide data with less visual impairments. Experimental results show resiliency of the proposed scheme against large blurring attack like mean and Gaussian filtering, nonlinear filtering like median,

4. Almost as discreetly as the technology itself, digital watermarking has recently made its debut on the geo-imaging stage.

Watermarking in Transformed domain:

1. This innovative technology is proving to be a cost-effective means of deterring copyright theft of mapping data and of ensuring the authenticity and integrity of rasterized image data

2. In the field of e-commerce, digital watermarking has already established itself as an effective deterrent against copyright theft of photographs and illustrations.

3. Now digital watermarking software is finding uses within national mapping agencies and others working with rasterized images or map data.

4. Current applications range from protecting valuable map data against copyright theft to securing photographic survey or reconnaissance images against tampering.

Digital watermarking applications:

Digital watermarking may be used for a wide range of applications, such as

1. Copyright protection

2. Source tracking different recipients gets differently watermarked content.

3. Broadcast monitoring: television news often contains watermarked video from international agencies

Digital watermarking evaluation

The evaluation of digital watermarking schemes may provide detailed information for a watermark designer or for end-users.

Displaying the translated Image:

We convert the bitmap object to a drawable object then display it on the surfaceView of the camera.

```
Drawable drawable = BitmapDrawable.createFromPath(imagePath);
mSurfaceView.setBackground(drawable);
```

After each translation process we display the resulting translated image to keep the live view.

Chapter9:

Conclusion & Future work

Conclusion:

Till know we reached the mobile application to captures images and scan it to get the words and detect the letters. Then translate them and display at mobile but the desired words to translate should be surrounding by borders.

Future work:

1. Translate more languages.
2. Enable more small sizes for fonts.
3. Decrease the time for the process of translation.
4. Search about the translated words.
5. Enable speaking for translated words.

Glossary

GP:	Graduation Project.
ARAT:	Augmented Reality Arabic Translation
IT:	Information Technology department
CS:	Computer Science department
IS:	Information Systems department
VR:	virtual reality
AR:	Augmented Reality
MR:	Mixed Reality
HMD:	Head-Mounted Display
IA:	Intelligence Amplification
OCR:	Optical character recognition

Faculty:	Permanent faculty member with a PhD degree.
Supervisor:	A fulltime faculty member in the College of Computer and Information Systems responsible for the supervision of a group of **GP**.
External Supervisor:	In case of an industrial project, a person assigned by the external organization as a supervisor.
Examiner:	A professor or an expert of the relevant area chosen from respective department, other departments of the College or outside the Zagazig University.
Coordinator:	A faculty member appointed by each department to coordinate the GP tasks, and prepare GP ABET course files.
Student:	A student registered for **GP** in **CS/IT /IS** at Zagazig University, Egypt.
Group/Team:	A group of students formed as a team to work on the **GP**.

At last moment for us at this place,
We would like to thank our friends that help us to spend our best days at these four years.