OPTICAL CHARACTER RECOGNITION ON AN ANDROID DEVICE

BY

CHIDOZIE AGBAI

100701955

DECEMBER 12, 2011

SUPERVISED BY DR. ANTHONY WHITE

SCHOOL OF COMPUTER SCIENCE

CARLETON UNIVERSITY

HONORS PROJECT (COMP 4905)
Abstract

Having an archive of important documents is always a good idea; this could be done with a filing system which involves a lot of papers or it could be done digitally with the use of computers.

However, a filing system could be unpleasant to navigate though and time consuming. With the aid of Optical Character Recognition (OCR), which is a technology enables conversion different types of documents into searchable and editable data; this can be performed more efficiently. Documents which would normally take hours to digitize can be performed in a matter of minutes and with the resurgence of mobile technology, it would be fitting to have such an application on a mobile phone.

For instance, a student working on a project who would want to his/her work digitized could just use their smart phone to take photos of the image and have and OCR software convert these images into digitized documents.

My goal is this project is to create an application using The Android SDK that will enable me to do this. I would also like to have a good knowledge of developing applications using Android and hopefully, become an active developer.
Acknowledgement

I would like to show my appreciation to Dr. Anthony White for giving me a chance to work with him on this project; creating an application that could be beneficial to a lot of students in the future and hopefully migrate to the public.

I would also like to acknowledge the team at Lab Asprice for providing the tools necessary to complete this project, including the Asprice API and Asprice jar files.

Finally I would like to thank the members of the Carleton Community that were instrumental in the making of this application.
Table of Contents

Abstract .................................................................................................................................2

Acknowledgement ..................................................................................................................3

List of Figures ..........................................................................................................................6

Introduction & Background .....................................................................................................7

  Introduction .......................................................................................................................... 7
  Types of Recognition Engines .............................................................................................. 7

Motivation ...............................................................................................................................8

  Android SDK ....................................................................................................................... 9
  Activity ................................................................................................................................. 9
  Content Providers ............................................................................................................... 10

Figure 1: Activity lifecycle (Android 2011) .............................................................................11

Asprice OCR ...........................................................................................................................12

Design OF OCR .....................................................................................................................12

  Matrix Matching ............................................................................................................... 12
  Fuzzy Logic ....................................................................................................................... 12
  Feature Extraction ............................................................................................................. 12
  Structural Analysis ........................................................................................................... 13
  Artificial Neural Networks (ANN) ..................................................................................... 13

Application Design ..............................................................................................................13

  UML Diagrams ................................................................................................................... 14
  Capture Image Use case ..................................................................................................... 15
Connect to Server .................................................................................................................. 17
Send Image to Server ........................................................................................................... 18
Process Image from Client ................................................................................................... 19
Send Image to Client ............................................................................................................ 20
Android Client ..................................................................................................................... 20

Figure 3: When user loads the application ........................................................................... 21
Figure 4: User selects the menu option for the camera ......................................................... 22

Server ................................................................................................................................. 24

Communication ................................................................................................................... 24

Conclusion ............................................................................................................................ 25

Limitations ............................................................................................................................ 25

Contribution to Field ............................................................................................................ 26

Appendices ........................................................................................................................... 27

DVD Contents ....................................................................................................................... 27

References ............................................................................................................................ 27
List of Figures

Figure 1 ................................................................................................................................................. 11
Figure 2 ................................................................................................................................................. 14
Figure 3 ................................................................................................................................................. 14
Figure 4 ................................................................................................................................................. 21
Figure 5 ................................................................................................................................................. 22
Figure 6 ................................................................................................................................................. 22
Figure 7 ................................................................................................................................................. 23
Figure 8 ................................................................................................................................................. 23
Introduction & Background

Introduction
The engineering attempts at automated recognition of printed characters started prior to World War II. But it was not until the early 1950's that a commercial venture was identified that justified necessary funding for research and development of the technology. This impetus was provided by the American Bankers Association and the Financial Services Industry. They challenged all the major equipment manufacturers to come up with a "Common Language" to automatically process checks. After the war, Check processing had become the single largest paper processing application in the world. Although the banking industry eventually chose Magnetic Ink Recognition (MICR), some vendors had proposed the use of an optical recognition technology. However, OCR was still in its infancy at the time and did not perform as acceptably as MICR.

OCR (Optical Character Recognition) is a technology that enables you to convert different types of documents such as scanned paper documents, PDF files or images captured by a digital camera into editable and searchable data. Images captured by a digital camera differ from scanned documents or image-only PDFs. They often have defects such as distortion at the edges and dimmed light, making it difficult for most OCR applications, to correctly recognize the text.

Types of Recognition Engines

- Intelligent Character Recognition (ICR): reads images of hand-printed characters (not cursive) and converts them into machine-readable characters.
• **Optical Mark Recognition (OMR):** OMR technology detects the existence of a mark, not its shape.

• **Magnetic Ink Character Recognition (MICR):** MICR is a specialized character recognition technology adopted by the U.S banking industry to facilitate check processing.

• **Barcode Recognition:** A barcode is a machine-readable representation of information

This project will be focused on Optical Character Recognition. An Android client will capture an image with machine-print text and send it to the server over sockets using TCP protocol and the server reads the characters from the image and sends it back to the android client.

**Motivation**

The motivation of this project comes from the growing interest and development of mobile technology. Suppose you want to digitize an article or a printed work, hours could be spent retyping and correcting the misprints. Or you could convert the required materials into digital format in several minutes using a digital camera and Optical Character Recognition software. This brought the inspiration to develop a mobile application which will be able to perform this task. With the development of mobile phones this is now possible.

Thanks to the Open Handset Alliance, the number of smartphones that support Android is increasing rapidly. Android requires no special hardware requirement and is easily obtainable through the Android website.

To satisfy the motivation for this project, a few objectives must be successfully completed

- Design and Implement an Android application that is able to take a picture of machine-printed characters.
Design a server that is able to take the image, process the image and return the text to the client.

**Android SDK**

Android is a software stack for mobile devices that includes operating system, middleware and key applications. Application components are the essential building blocks of an android application. Each component is a different point through which the system can enter your application. Not all components are entry points as there are some components which depend on others.

There are four types of components:

- Activities
- Services
- Content Providers
- Broadcast Receivers

In this project, Activities and Content providers were used so I will elaborate on these two components.

**Activity**

An activity is an application component that provides a screen with which the user can interact with the application (Android 2011). A good example of an activity on an android device would be the window that shows up when the phone application is selected, email application or even a map view.

An application can consist of multiple activities which are loosely bound to each other; this can be done through intents. An *Intent* object is a bundle of information. It contains information of interest to the component that receives the intent plus information of interest to the Android system. However, one activity in the application is always considered as the main activity (the main point of entry into the application).

The most important callbacks (methods) in an activity are:
**onCreate()**

This method must be implemented. The system calls this when creating an activity. In here, the essentials of the components are initialized. Most importantly, this is where you must call `setContentView()` to define the layout for the activity.

**onPause()**

The method is called when the user is leaving the activity which does not mean the activity has been destroyed. This is where the user commits changes that should be persisted.

The lifecycle of an activity is described below in figure 1.

**Content Providers**

Content providers store and retrieve data and make it accessible to all applications. They are the only way to share data across application (Android 2011).

If you want to make your data public, you could either create your own content provider or add to an existing content provider. For this application, the “MediaStore” content provider was used. It is the main provider that dispenses image, audio and video data.

More information about Content providers can be found on the Android website.
Figure 1: Activity lifecycle (Android 2011)
**Asprice OCR**

Asprice OCR is a high performance OCR engine which offers APIs for Java, VB.NET, CSharp.NET, C, C++, Delphi, VB6.0, C (asprise.com). Asprice is a very useful tool as it enables you to equip your applications with OCR ability easily.

Computer systems equipped with OCR system can improve the speed of input operation and decrease human errors. Recognition of printed characters is itself a challenging problem since there is a variation of the same character due to change of font or introduction different type of noises. Therefore, a good character recognition approach must eliminate the noise after reading binary image data, smooth the image for better recognition, extract features efficiently, train the system and classify patterns.

**Design OF OCR**

Various approaches used for the design of OCR systems are discussed below:

**Matrix Matching**

Matrix matching converts each character into a pattern within a matrix, and compares the pattern with an index of known characters.

**Fuzzy Logic**

Fuzzy logic is a multi-valued logic that allows intermediate values to be defined between conventional evaluations like yes/no, true/false, black/white etc. An attempt is made to attribute a human-like way of logical thinking in the programming of computers. This is used when answers do not have distinct value.

**Feature Extraction**

This method defines character by the presence or absence of key features, including height, width density, loops and other character traits. Feature extraction is a perfect approach for OCR of magazines, laser print and high quality images.
Structural Analysis
Structural Analysis identifies characters by examining their sub features—shape of the image, sub-vertical and horizontal histograms. Its character repair capability is great for low quality text and newsprints.

Artificial Neural Networks (ANN)
This strategy simulates the way human neural system works. It samples the pixels in each image and matches them to a known index of character pixel patterns. A benefit of using Neural Network in OCR is extensibility of the system’s ability to recognize more character set that initially defined. Artificial Neural Networks are collections of mathematical model that represent some of the observed properties of biological nervous systems and draw on the analogies of adaptive biological learning. ANN consists of large number of highly interconnected processing elements (nodes) that are tied together with weighted connections (links). Originated in late 1950’s, neural networks did not gain much popularity until 1980s. Today, ANNs are mostly used for solution of complex real world problems. They are often good at solving problems which an algorithmic solution is too for conventional technologies and are often well suited to problems that people are good at solving but for which traditional methods are not.

Asprise OCR uses Matrix Matching.

Application Design
For this application, the Android client should be able to take a photo of an image, connect to a server (Java-based) and send the server the image for processing. The server processes the image, performs OCR on the image and sends the result back to the client as a strings
**UML Diagrams**

Figure 2: UML for server

Figure 3: UML for client
The project is divided into two sub-projects for convenience.

1. **OcrEye**: an Android based 2.1 (Eclair_Mr1) based client.
   
   An Android client that contains the custom view and Activity classes for the purpose of capturing an image for manipulation.

2. **OcrService**: A java based Server.
   
   A server which takes an image as an input and transforms the characters (if any) in the image to machine-readable characters.

   In the section that follows, a use case scenario will be used to describe the design of the application.

Use Case Models

There are four (4) use cases associated with this application.

- Capture Image
- Connect to Server
- Send Image to Server
- Process Image from Client
- Send Image to Server

Below is a descriptive scenario of each use case scenario.

**Capture Image Use case**

<table>
<thead>
<tr>
<th>Description</th>
<th>User tries to capture an image with the Android Client</th>
</tr>
</thead>
</table>
| **Actors**     | • Android Client  
|               | • Server         |
| **Trigger**   | This is triggered when the Android Client selects the camera option on the application |
| **Precondition** | The application is active |
| **Basic Flow of Events** | • Client captures an image |
| **Alternate Flow** | |
| **Post Condition** | • Image is successfully saved in the Android Client. |
### Connect to Server

<table>
<thead>
<tr>
<th>Description</th>
<th>User attempts to connect to server</th>
</tr>
</thead>
</table>
| **Actors**  | • Android Client  
              • Server |
| **Trigger** | This is triggered when the Android Client selects the send option on the application |
| **Precondition** | • The application is active  
                           • Image has been captured |
| **Basic Flow of Events** | • Client connects to server |
| **Alternate Flow** | • Unable to detect server  
                             • Client times out |
| **Post Condition** | Client connects server. |
## Send Image to Server

<table>
<thead>
<tr>
<th>Description</th>
<th>Client sends image to server</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors</strong></td>
<td>• Android Client</td>
</tr>
<tr>
<td></td>
<td>• Server</td>
</tr>
<tr>
<td><strong>Trigger</strong></td>
<td>This is triggered when the Android Client selects the send option on the application</td>
</tr>
<tr>
<td><strong>Precondition</strong></td>
<td>• The application is active</td>
</tr>
<tr>
<td></td>
<td>• Server is running</td>
</tr>
<tr>
<td></td>
<td>• Image has been captured</td>
</tr>
<tr>
<td></td>
<td>• Connection has been established</td>
</tr>
<tr>
<td><strong>Basic Flow of Events</strong></td>
<td>• Client sends the image to the server</td>
</tr>
<tr>
<td><strong>Alternate Flow</strong></td>
<td>• Connection is lost</td>
</tr>
<tr>
<td></td>
<td>• Image does not get to server</td>
</tr>
<tr>
<td><strong>Post Condition</strong></td>
<td>• Image is successfully saved into the Android Client.</td>
</tr>
</tbody>
</table>
**Process Image from Client**

<table>
<thead>
<tr>
<th>Description</th>
<th>Server processes image</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors</strong></td>
<td>• Server</td>
</tr>
<tr>
<td><strong>Trigger</strong></td>
<td>Server receives image from android client</td>
</tr>
<tr>
<td><strong>Precondition</strong></td>
<td>Server is running</td>
</tr>
<tr>
<td><strong>Basic Flow of Events</strong></td>
<td>• Server receives image, processes the data and gets the result</td>
</tr>
</tbody>
</table>
| **Alternate Flow** | • Server unable to process the data  
• Outputs ascii characters |
| **Post Condition** | • Server has successfully/unsuccessfully processed the image |
**Send Image to Client**

<table>
<thead>
<tr>
<th>Description</th>
<th>Server sends result back to client</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actors</strong></td>
<td>• Server</td>
</tr>
<tr>
<td></td>
<td>• Android Client</td>
</tr>
<tr>
<td><strong>Trigger</strong></td>
<td>Triggered when server sends image to client</td>
</tr>
<tr>
<td><strong>Precondition</strong></td>
<td>• The application is active</td>
</tr>
<tr>
<td><strong>Basic Flow of Events</strong></td>
<td>• Client receives the text and saves it</td>
</tr>
<tr>
<td><strong>Alternate Flow</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Post Condition</strong></td>
<td>• Text is successfully saved on the android client</td>
</tr>
</tbody>
</table>

**Android Client**

The client is composed of two (2) activity classes.

- ImageSenderActivity (IMS)
- Camera Activity (Called implicitly)
- Android should run at version 2.1 or higher.

Upon startup the user sees the IMS (our main point of entry). The ImageSenderActivity has an `ImageView` a two `MenuItem` on its menu; `Camera`, menu item for taking photos of an image and `Send`, the other for sending the image to the server.
IMS calls a takePicture() method which calls the default android’s Camera Activity and saves the image returned by the Camera Activity to the ImageView object. The user selects the send menu item, the IMS attempts to establish connection with the server. If the server is running, a connection is established and the IMS sends the image to the Server for processing and creates a Runnable, Receiver which acts as the channel for incoming messages from the server. The communication is made between server and client over sockets using TCP protocol.

Upon startup of the IMS, the user sees a blank screen because the user has yet to capture an image. The user has the option to select the camera menu item on the menu.

Figure 4: When user loads the application
If the user selects the camera option, the application calls the Camera Activity to take a shot of the image. After taking a photo of the image, the user can either send the image to the server for processing or take another photo, if the user decides the photo is not good enough for processing.
When the user has taken the photo of the image and the user decides to send the image to the server for processing, this can be done by selecting the send option.
Server

The server uses the reactor pattern to process the requests of the client. This is to make easier to add more functionality to the application in the future.

The server application has an Acceptor which listens for connection from clients and spawns a new Runnable, ClientWorker for a client when a connection has been established.

A ClientWorker passes its request to the reactor where it is handled by an appropriate EventHandler (right now just one EventHandler). The EventHandler reads the request of the client, processes it (in this case the image for translation) and sends the response to the client.

The following are the classes present in on the server application:

- Acceptor
- EventHandler
- Reactor
- Acceptor
- ClientWorker
- ServerMain

Communication

Communication between clients and the server is done with sockets and handled using the TCP protocol.

Clients initiate communication with server by sending a request to the server to perform a task. Clients send Objects to the server; OCRResource, which contains the name of the file being sent over the network, the type of request to be performed (called message) a byte array, containing the contents of the file to be processed, which is an image. The server translates an image into text. The server receives the clients request, sends decomposes the message using a reactor, performs the necessary operation for the request (in this case converting the image into text) and sends a String response to the client with the contents of the image.
Unfortunately, there are limitations to sending an image this way. The image might be distorted therefore making it difficult for the server to read it. A more efficient approach will be to pre process the image before sending to server.

**Conclusion**

Despite achieving the goals for this project, there were a lot of obstacles on the way. The means of reaching the target goal was changed after careful examination; converting the characters into text was first approached with the idea of implementing a code which performs the digitization of these photos.

Considerable amount of time was spent learning the different functionalities of the Android SDK. For example, spending a lot of time trying to understand how to build a Camera Activity class before coming to a realization that with the use of Intent objects, I can call the default camera of the android device.

However after spending a considerable amount of time learning the Android-specific java code, I will be looking into adding more functionality for the application.

**Limitations**

A significant limitation discovered during the implementation of the project. Most OCR software packages are used in scanning papers and reading from PDF files. Images captured with digital camera often have defects such as distortion at the edges and dimmed light, making it difficult for most OCR application to correctly recognize the text. Also most of the OCR softwares applications discovered were commercialized and considerable expensive for someone on a student budget.

**Future Improvements**

This is by no means a polished product, with plenty of room for improvement to be made. The User interface is too simple. The application only allows for the user to send the current images viewed on the Camera Activity.
I suggest a wider range of options to be made available for the user in sending images. For example, implementing a grid view which allows the user a number of options for images to process.

Also, as hinted above, OCR technology does not handle distorted images efficiently. Improving the quality of images sent to the OCR software will be a good way around this. This will involve having a high resolution digital camera, equipped with the following;

- The use of a high resolution Digital Camera, equipped with optical zoom, an anti-shake feature, auto focus assist beam or manual focusing, manual aperture control or aperture priority mode.
- Taking two to three shots of the same documents, to make sure the image does not come out blurry or corner of the document was not cut off.
- Artificial Neurone Network would be the best algorithm suited for this application.

A room with good lighting will be ideal for capturing an image. Natural light is always the best, in the absence of natural light, a camera with flash can be used but image should be captured about 20 inches from the document.

I would suggest for future developments, images should be processed before being sent to the server for better results.

**Contribution to Field**

Due to the nature and goals of the project, no significant contributions to the field in terms of algorithms have been made. The purpose of this project is to allow the author and users familiarize themselves with the Android platform and allow for future developments. Hopefully, readers of this document could be inspired to improve on this Android application and those unfamiliar with the Android platform might find this document (along with the provided source code) useful.
Appendices

**DVD Contents**

Provided with this report is a CD containing the following:

- OcrEye Source code
- OcrService source code
- PDF of project proposal
- PDF of project report

Development Environment

The Eclipse IDE was used for both the Android and Java development. The tools required to run the provided source code include.

- Java JDK
- Android SDK (version 2.1 or higher)
  
  http://developer.android.com/sdk/installing.html
- ADT plugin for Eclipse http://developer.android.com/sdk/eclipse-adt.html#installing
- Android device (Version 2.1 or higher)
  
  An emulator can be used but an android device is strongly recommended.
  
  http://developer.android.com/guide/developing/device.html#setting-up

References

http://finereader.abbyy.com/about_ocr/whatis_ocr/
http://developer.android.com/guide/topics/fundamentals.html