Textor: A Mobile Application that Extracts Text from Image Format

A Thesis Presented to

the Faculty of the College of Computer Studies

Lyceum of the Philippines

University - Batangas

In Partial Fulfilment

of the Requirements for the Degree

Bachelor of Science in Computer Science

Specialized in Mobile Application Development

By:

James Israel V. Camarillas

Ryan John Michael Conway

Jhon Carlo M. Fernando

Jose Simao F. Jones

John Michael F. Tangile

March 2017

APPROVAL SHEET

In partial fulfilment of the requirements for the degree Bachelor of Science in Computer Science (Specialized in Mobile Application Development), this thesis entitled **"Textor: A Mobile Application that Extracts Text from Image Format"** has been prepared and submitted by James Israel V. Camarillas, Ryan John Michael Conway, Jhon Carlo M. Fernando, Jose Simao F. Jones, and John Michael F. Tangile and is hereby recommended for oral defense.

Mrs. Roselie B. Alday, MCS, PhD.Cand.
Adviser

Defended in an oral examination before a duly constituted panel with a grade of

_______.

Miss Mischelle A. Esguerra, MIS
Chairman

Mr. Aris Gail S. Mendoza
Member

Mrs. Janice E. Velasquez, MSCS
Member

Mrs. Chona Andal, MAEd
Member

Accepted in partial fulfilment of the requirements for the degree Bachelor of Science in Computer Science (Specialized in Mobile Application Development).

Mrs. Roselie B. Alday
Dean, College of Computer Studies

Acknowledgement

The researchers would like to extend their deepest appreciation to the people who greatly contributed to the success of this thesis.

To our parents who gave their moral support and untiring guidance. To our adviser, Mrs. Roselie B. Alday for her continuous support, patience and motivation. And above all, to Almighty God for providing us the opportunity and granting us the capability to successfully complete our research.

James Israel V. Camarillas

Ryan John Michael Conway

Jhon Carlo M. Fernando

Jose Simao F. Jones

John Michael F. Tangile

Dedication

We dedicate our thesis to our family, friends and mentors whose words and action stimulated us in the course of finishing this research. We recognize the worth of their support in uplifting our spirits while we struggled to look for means and ways to take on this project. We will always appreciate their given time and effort to help us develop our technological and secular skills.

James Israel V. Camarillas

Ryan John Michael Conway

Jhon Carlo M. Fernando

Jose Simao F. Jones

John Michael F. Tangile

Table of Contents

Title Page i
Approval Sheet ii
Acknowledgement ii
Dedication iv
Table of Contents v
List of Tables and Figures vi
Abstract 1
1.0 Introduction 1
__1.1 Objectives 2
2.0 Review of Literature 2
__2.1 Text detection 2
__2.2 Optical Character Recognition 3
_____2.2.1 Scanning and Thresholding 3
_____2.2.2 Segmentation 4
_____2.2.3 Pre-Processing 4
_____2.2.4 Feature Extraction 4
_____2.2.5 Recognition 4
___2.3 Applications That Use OCR 4
___2.4 Tesseract OCR Engine 5
______2.4.1 Architecture of Tesseract 5
___2.5 Algorithm 6
___2.6 Flow of the Application 7
3.0 Methods 8
4.0 Results and Discussions 9
5.0 Conclusion 10
6.0 Recommendation 10
References 10
Appendices 12
A. Screenshots 12
B. Codes 15
C. Compatible Devices 22
D. Sample Test Results 23

List of Tables and Figures

Figure 1.0 OCR Components ……………………………………………….. 3

Figure 2.0 Tesseract Architecture ……………………………………………... 6

Figure 3.0 Tesseract Algorithm ………………………………………………... 7

Figure 4.0 Application Flow ……………………………………………………. 7

Figure 5.0 Launcher Icon ………………………………………………………. 12

Figure 6.0 Splash Screen ……………………………………………………. 12

Figure 7.0 Main Screen ………………………………………………………… 13

Figure 8.0 Instructions Screen ………………………………………………… 13

Figure 9.0 About Screen ……………………………………………………….. 14

Figure 10.0 Extracted Text Screen …………………………………………… 14

Table 1.0 Results of Testing ………………………………………………... 9

Table 2.0 Accuracy of Font Groups …………………………………………... 9

Table 3.0 List of Compatible Devices ……………………………………… 22

Table 4.0 Samsung Galaxy S7 Test Results ……………………………….. 23

Textor: A Mobile Application that Extracts Text from Image Format

James Israel V. Camarillas
Lyceum of the Philippines University
Capitol Site Batangas City
james.camarillas@gmail.com
ORCID No. 0000-0002-9340-1932

Ryan John Michael Conway
Lyceum of the Philippines University
Capitol Site Batangas City
rjmconway@hotmail.co.uk
ORCID No. 0000-0002-3698-5484

Jhon Carlo M. Fernando
Lyceum of the Philippines University
Capitol Site Batangas City
jcfernando.ccs@gmail.com
ORCID No. 0000-0003-4365-0815

Jose Simao F. Jones
Lyceum of the Philippines University
Capitol Site Batangas City
jonesjose.ccs@gmail.com
ORCID No. 0000-0001-9835-2255

John Michael F. Tangile
Lyceum of the Philippines University
Capitol Site Batangas City
michaeltangile.ccs@gmail.com
ORCID No. 0000-0001-9398-2848

Abstract

Textor is a mobile application developed using the Tesseract algorithm and engine to convert text in image form into a digital format that can be easily edited and shared through various means. The application allows users to capture an image of a line of text, detects the characters and outputs them to the user. The output of the application can then be copied to the user's clipboard or shared through various means such as through E-mail or Twitter. It was developed using the Java programming language and incorporates a modified version of the Tesseract engine that was created to run on mobiles devices.

Keywords: *Android, Java, Mobile application, Tesseract, Optical Character Recognition*

1.0 **Introduction**

According to Sami L. (2002) Optical Character Recognition has long been used by libraries and government agencies to make lengthy documents quickly available electronically. Advances in OCR technology have spurred its increasing use by enterprises. OCR is the most cost-effective and speedy method available in digitalizing various documents. "Older OCR systems match these images against stored bitmaps based on specific fonts. The hit-or-miss results of such

pattern-recognition systems helped establish OCR's reputation for inaccuracy.” (Sami 2002). With the fast advance in technology, books are becoming of lesser use and are becoming dormant sources of information. The time will come that books will no longer need to
exist physically but will be available electronically. The need for OCR by individuals willing to digitalize books led to the development of such application

The study focuses on the application of the Tesseract Algorithm and Engine in order to create an OCR application. The mobile application incorporates Optical Character Recognition in order to recognize and extract text from images. These images may be taken by the user or chosen from a different source through the camera on their smartphone or from the pictures already present on the device.

The application is limited only to a single line of text and requires proper lighting when capturing images with text. It also cannot recognize punctuation and symbols when detecting text. The application requires a smart phone that has a back camera that is 5 megapixels or more and an Android version that is equal to or higher than v.4.0, also known as Android Ice Cream Sandwich.

The researchers recommend that the application be improved by learning to recognize more than a single line of text, improving the recognition capabilities of the tesseract algorithm, such as allowing for the recognition of punctuation and symbols, and be able to recognize hand written text.

1.1 Objectives

This paper aims to:

1. Develop a mobile application that will extract text from images and printed documents.
2. Use Tesseract Algorithm to extract text from images.
3. To use the Java programming language in the development of the application.

2.0 Review of Literature

The researchers made use of different references to define the problem and its solution with regards to extracting text from images into an editable format.

2.1 Text detection

Text detection employs the use of various methods to extract text from journal images, camera captured images, video images, printed document, degraded document images, handwritten historical document, graphical and colour document images, low resolution images, book cover and web pages. In general, text extraction methods use the colours of the foreground and the background as the main information source to separate them. That is, they divide the colour space in groups, and each of them is classified as foreground or background, so are the pixels which contain those colours. The first version of text extraction methodologies performed the segmentation using grey-scale images, assuming that the backgrounds were clean, and the degradations on the image were small enough not to be considered. (Canedo-Rodriguez, 2012)

2.2 Optical Character Recognition

Optical Character Recognition, or OCR, is a technology that allows different types of documents, such as scanned paper documents, PDF files or images captured by a digital camera to be converted into editable and searchable data (“What is OCR and OCR Technology”, n.d.). OCR employs the use of automatic recognition of patterns, the first principle of which is to teach the machine what classes of patterns may occur and what they look like. These patterns are usually letters, numbers and certain special symbols such as punctuation marks, with each class corresponding to a different character (Eikvil, 1993).

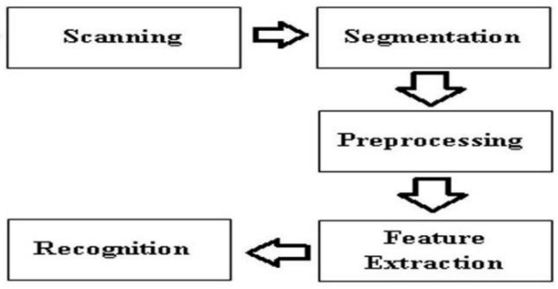


Figure 1.0 OCR Components

The Figure 1.0 given above illustrates the overall functions of Optical Character Recognition (OCR). It contains the steps needed to recognize text. These steps are scanning, segmentation, pre-processing, feature extraction and recognition. Here, the input image is any printed text such as the pages of a book, magazine, newspaper, etc. Such input is given to the OCR formula. First, the text is scanned using a smartphone’s camera. This allows for the digitization of the analogue document. Text regions within the newly digitized image are located, symbols are then extracted through segmentation, each one is pre-processed, has its features extracted and finally the text is recognized.

2.2.1 Scanning and Thresholding

Scanning is used to capture a digital image of a document or piece of text. In general, this is performed using a camera, which captures the image of the document. We can thus say that scanning takes an original document and transforms it into a digital image. Generally, original documents are made up of a black coloured text print overlaid on a white coloured background. The scanning process comes with thresholding which converts the digital image to grey scale. (Lian, 2009)

Thresholding is the process which converts a multi-level image into a bi-level image, i.e. a black and white image. A fixed threshold level is defined in the thresholding process. If the grey levels are below the threshold level, the pixels will be identified as black. Likewise, if the grey level is above the threshold value, the pixels will be identified as white. This results in saving memory space and computational efforts.

2.2.2 Segmentation

Segmentation is the process of locating regions of printed or handwritten text. Segmentation differs text from figures and graphics. When segmentation is applied to text, it isolates characters and words. The most commonly occurring problem in segmentation is that it can become confused between text and graphics in the case of joined and split characters. Usually, splits and joints in the characters are caused during the scanning phase. If the document is a dark photocopy or if it is scanned at low threshold, joints between characters can occur. Similarly, splits in characters can occur if the document is a light photocopy or is scanned at a high threshold. An OCR system may also get confused during segmentation when characters are connected to graphics.

2.2.3 Pre-Processing

Some noise may occur during the scanning process. This results in poor recognition of characters due to the fuzziness. This problem can usually be overcome through the use of pre-processing. Pre-processing consists of the smoothing and normalization of the image. In smoothing, certain rules are applied to the contents of image with the help of filling and thinning techniques. Normalization is responsible for handling the uniform size, slant and rotation of scanned characters.

2.2.4 Feature Extraction

Feature extraction, as the name implies, extracts the features of symbols. Features are the specific characteristics of each symbol. In this, symbols are characterized and unimportant attributes are discarded. The feature extraction technique does not make use of concrete character patterns, but rather makes note of abstract features present in a character such as intersections, open spaces, lines, etc., and extrapolates as to what character those features represent. The Tesseract algorithm is used to implement feature extraction. Feature extraction is concerned with the correct representation of the symbols. The character image is mapped to a higher level by extracting special characteristics of the image in the feature extraction phase. (Eikvil, 2009)

2.2.5 Recognition

The OCR system works with the Tesseract algorithm which recognizes characters. Tesseract identifies characters in foreground pixels, called blobs, and then it finds lines. Word by word recognition of characters is done throughout the lines. Recognition involves converting these images to character streams representing letters of recognized words. In short, recognition extracts text from images of documents.

2.3 Applications That Use OCR

While there are several applications currently on the android market that make use of OCR, most of these do not truly extract text, only enhance the inputted image to make the text more readable. An example of this would be OCR – Text Scanner by Rishi Apps. This application makes use of the Tesseract Algorithm, but does not perform well in most situations. It also requires the user to download an additional language pack before they can first use the application and includes intrusive, full screen advertisements.

However, there are OCR applications that perform exceedingly well, such as Google Goggles or Microsoft's Office Lens. Unfortunately, these applications also have their own shortcomings. Office Lens, for example, requires the user to have a Microsoft account in order to save the extracted text to a Word document or OneNote, as the save location for these formats is in the user's cloud storage. The user may still save the image taken to their device's gallery, but no text extraction will take place.

On the other hand, Google Goggles does not require the user to sign up for anything and offers many more features than any other OCR application, but all the computing is performed on Google's servers, meaning that the application requires an internet connection at all times. This can be problematic for users without a mobile data plan or access to Wi-Fi.

2.4 Tesseract OCR Engine

Tesseract is an open-source OCR engine that was developed by Hewlett-Packard (HP) between 1984 and 1994. It began as a research project by HP Labs as an add-on for their line of flatbed scanners, motivated by the fact that most OCR engines at the time failed to detect text in all but the highest quality prints. Since HP had independently-developed page layout analysis technology that was used in products, (and therefore not released for open-source) Tesseract never needed its own page layout analysis. The engine therefore assumes that its input is a binary image with optional polygonal text regions defined. (Smith, 2007) In August of 2006, Google began funding and supporting the research and development of Tesseract after HP abandoned the project in 1995 and subsequently rereleased it as an open source project in the early 2000's to allow other developers to continue its research. Much of the early bugs were fixed by the Information Science Research Institute at the University of Nevada, Las Vegas (UNLV), whose researchers eventually asked Google to help with other more advanced complications within the code. (Vincent, 2006)

2.4.1 Architecture of Tesseract

Tesseract works with independently developed Page Layout Analysis Technology. Hence Tesseract accepts input images as a binary image. Tesseract can handle both the traditional black on white text and as well as the inverse of white on black text. Outlines of components are stored in connected Component Analysis. Nesting of outlines is done which gathers the outlines together to form a blob. These blobs are then organized into text lines, which are analysed for fixed pitch and proportional text.

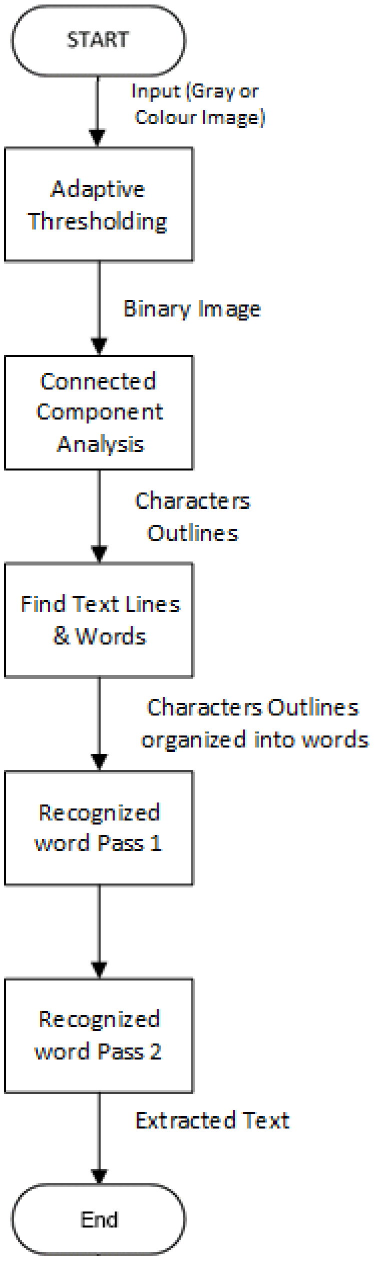


Figure 2.0 Tesseract Architecture

Then the lines are broken into words by analysis according to the character spacing. Fixed pitch is chopped in character cells and proportional text is broken into words based on definite spaces and fuzzy spaces. Tesseract then performs its function to recognize words. This recognition activity consists of two passes. The first pass attempts to recognize the words. Then satisfactory words are passed to the Adaptive Classifier as training data, which is used to recognize subsequent text with greater accuracy. During second pass, the words which were not recognized well during first pass are analyzed again and recognized through a second run over the page. Finally, Tesseract resolves fuzzy spaces within the image. To differentiate between small and capital text, Tesseract checks alternative hypotheses for text height.

2.5 Algorithm

The Tesseract algorithm was used to detect the text within the inputted images.

The algorithm starts by first selecting an image to be scanned, either directly from the user's camera or from another source. Adaptive thresholding is then applied to the image in order to produce a black and white, or binary, image. The algorithm then identifies the foreground pixels of the image.

The image is then segmented into "blobs", with each blob then being filtered to increase its potential readability. The algorithm then estimates the positions of the base lines where the text is located, where the filtered blobs are then merged. The fixed pitch for the image is then found. The Tesseract algorithm then chops the words detected into individual characters and measures the gaps between them. This process will repeat until a certain result is met.

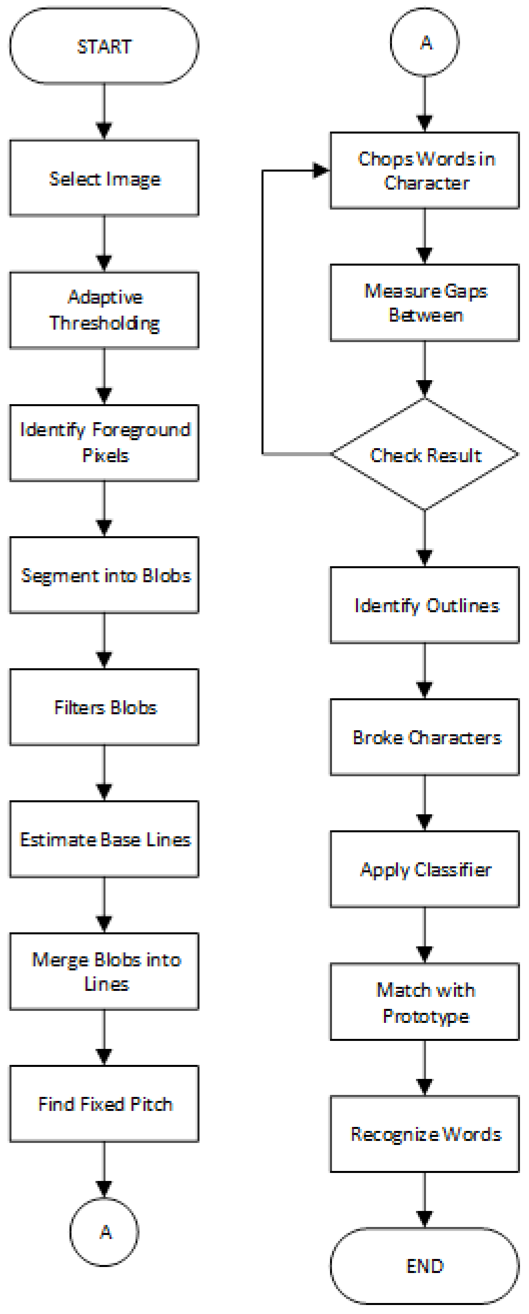


Figure 3.0 Tesseract Algorithm

The outlines of the individual characters are then identified along with the broken or "incomplete" characters. The results are then classified based on their shapes and relative distances between the "broken" segments, the results of which are matched with the prototypes of the "learned" characters. This then leads to proper words being recognized by the system, ending the algorithm's work.

2.6 Flow of the Application

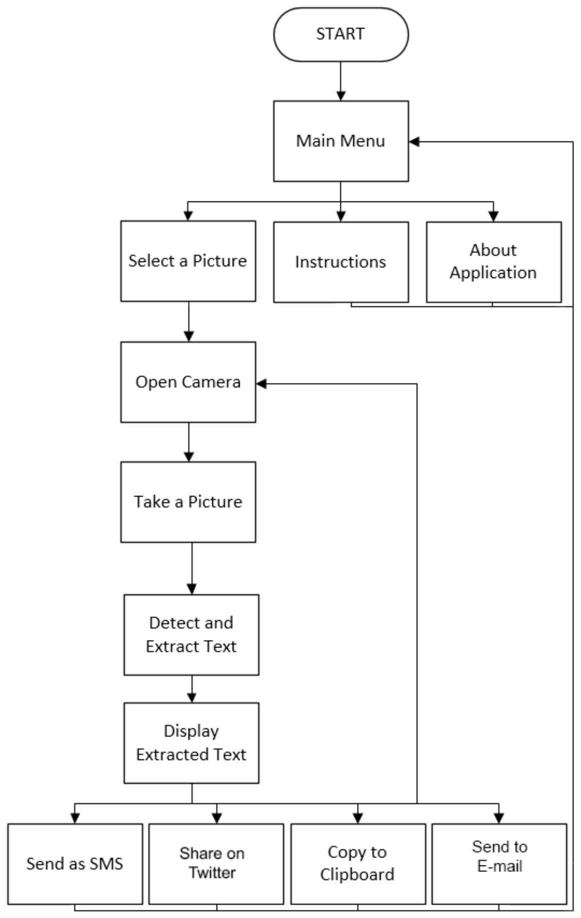


Figure 4.0 Application Flow

When the user starts the application a main menu screen will appear holding the options for select picture, instructions and about application. The first option allows the user to take a picture of a piece of text using the phones built-in camera. It then detects the text in the image and extracts it into a text format. After extracting the text from the image, the application will provide several choices for the user: Sending the text via Short Message Service (SMS), sharing the text on Twitter, copying the text to the user's clipboard, sending the text via E-mail, or allowing the user to capture another image, which will add the next line of detected text to what has already been detected and extracted. The second option on the main menu displays the instructions on how to properly use the application, while the third option provides information about the application.

3.0 Methods

The programming language used by the researchers is Java, which is a general purpose, high-level programming language developed by Sun Microsystems (Beal, n.d.). Java is an object-oriented language similar to C++, but simplified to eliminate language features that cause common programming errors. Java source code files (files with a .java extension) are compiled into a format called bytecode (files with a .class extension), which can then be executed by a Java interpreter. (Rouse, 2007)

The authors made use of the Eclipse as the programming environment. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages through the use of plugins. The researchers also made use of Android Development Tools (ADT), a Google-provided plugin for the Eclipse Integrated Development Environment (IDE) that is designed to provide an integrated environment in which to build Android applications. ADT extends the capabilities of Eclipse to let developers set up new Android projects, create an application UI, add packages based on the Android Framework API, debug their applications using the Android Software Development Kit (SDK) tools, and export signed (or unsigned) apk files in order to distribute their applications. It is free to download. It was the official IDE for Android but was replaced by Android Studio (based on IntelliJ IDEA Community Edition).

In order to accomplish the study, the researchers made use of various procedures and methods to gather the information needed to develop the mobile application. Various ways that Optical Character Recognition can be implemented were investigated by the researchers. After combing through the different approaches for the most effective method, the researchers decided to make use of the Tesseract Engine and usage of the Tesseract algorithm. The engine and its algorithm were integrated into a mobile application. Students, researchers and people who deal with a great deal of text in their day to day lives will benefit from this mobile application. It can help to reduce the time it takes to duplicate a written or printed work into a digital format for quick and easy editing. This will in turn reduce the workload of the user as they will usually only need to make minor changes to the scanned text, with the best case scenario involving no editing at all.

4.0 Results and Discussions

The researchers tested the accuracy of the application and the Tesseract algorithm by scanning a predetermined piece of text printed using several commonly used font styles. The text used was “the quick brown fox jumps over the lazy dog”, as it includes the letters from A to Z, both in upper and lower case, as well as the numbers from 0 to 9. The styles were divided into four categories: Serif, sans serif, script and decorative. The text was scanned using several font styles that were chosen from each category and the averages of their accuracy was calculated to obtain a final accuracy rating. The results of the testing are as follows:

Font Style	*Lower case*	*Upper case*	*Num*	*Ave*
Serif	93.72	87.67	73.00	84.80
Sans Serif	95.00	89.24	64.25	82.83
Script	59.28	45.12	50.67	51.69
Decor	43.53	43.07	30.80	39.13

Table 1.0
Results of testing

Table 1.0 above displays the results of testing the tesseract algorithm on the different font categories. The results above are the averages of the different fonts from each category. The most easily recognized font type is the Serif style, with an overall accuracy of 84.80%. This is closely followed by Sans Serif fonts at 82.83%. Script fonts were difficult to recognize, with the algorithm only able to correctly extract 51.69% of the text. Finally, the algorithm could not accurately recognize decorative fonts, with an accuracy rating of only 39.13%.

With this information, three subsequent results can be computed.

Font Group	*Accuracy*
Serif and Sans Serif	83.81
Script and Decorative	45.41
Overall Accuracy	64.61

Table 2.0
Accuracy of font groups

When looking at the overall accuracy of the algorithm using all four categories of fonts, the accuracy is 64.61%, as seen above in Table 2.0. However, when looking at only Serif and Sans serif fonts, the most commonly used types in printed text, the accuracy rises to 83.81%. This can be attributed to the low rating of the Script and Decorative fonts, which is at 45.41%.

5.0 Conclusion

From the overall assessment of the application, it can be noted that there are many aspects that affect the accuracy of the application’s ability to detect text. Different font sizes, styles, lighting conditions, angular positioning of the camera, the specifications of the device used and the quality and quantity of the text to be converted are among the identified factors from the testing performed.

Concerning font size, the researchers found that the larger the size of the printed text, the more accurate the detection was likely to be. With regards to the lighting conditions, natural light provided the best conditions for detection, either direct sunlight or having the text in close proximity to a window. Ample light is required in order to reduce the amount of noise in the image without affecting the quality of the text when in image form. As for the angular positioning of the camera, a straight image captured with the camera parallel to the text produced the most accurate results, with the accuracy falling as the angle increases. Camera quality also played a role in the accuracy of the detection, with higher specifications providing better results, especially the size of the camera’s pixels, with larger sensors providing the greatest boost in accuracy. Furthermore, the application developed was found to be useful and accurate when the text image uses the Serif and Sans serif font styles. The quality of the printed text also played a role, with sharper text being easier to detect.

Finally, the amount of text that is to be detected has a large impact on the application’s accuracy. With single lines of text, the application can accurately detect the text assuming all other variables are at suitable conditions. However, when multiple lines of text are captured in a single image, the accuracy of the detection falls dramatically, often producing high amounts of garbage characters as the algorithm attempts to process several lines of text together as a single line.

The researchers also discovered that the application is unable to accurately recognize handwritten text, either producing garbage results or not detecting any text at all.

6.0 Recommendation

The researchers recommend that further research and development be done to improve the accuracy of the application and to enhance its current set of features. This includes the ability to detect multiple lines of text accurately, reducing the amount of garbage text that is found within the images taken, and allowing for the recognition of punctuation marks and special symbols.

References

Beal, V. (n.d.). What is Java? A Webopedia Definition. Retrieved from http://www.webopedia.com/TERM/J/Java.html

Canedo-Rodríguez, A., Kim, J. H., Kim, S. H., Kelly, J., Kim, J. H., Yi, S., ... & Blanco-Fernández, Y. (2012). Efficient text extraction algorithm using color clustering for language translation in mobile phone.

Lais, Sami. (2002, May). Optical Character Recognition. Retrieved from http://www.computerworld.com/article/2577868/app-development/optical-character-recognition.html

Lian, Wing-Soon Wilson, and Diane P. Pozefsky. Heuristic-Based OCR Post-Correction for Smart Phone Applications. Diss. Thesis. Chapel Hill: The University of North Crolina, 2009.

Line Eikvil. (1993). OCR – Optical Character Recognition. Retrieved from https://www.nr.no/~eikvil/OCR.pdf

Mithe, R., Indalkar, S., & Divekar, N. (2013). Optical character recognition. International Journal of Recent Technology and Engineering (IJRTE) Volume, 2, 72-75.

Rouse, M. (2007, May). What is Java? – Definition from WhatIs.com. Retrieved from http://searchsoa.techtarget.com/definition/Java

Smith, R. (2007, September). An overview of the Tesseract OCR engine. Inicdar (pp. 629-633). IEEE.

Vincent, L. (2006, August 30). The Official Google Code Blog. Retrieved from Blogspot: http://googlecode.blogspot.com/2006/08/announcing-tesseract-ocr.html

What is OCR and OCR Technology? (n.d.). Retrieved from http://www.abbyy.com/finereader/about-ocr/what-is-ocr/

Appendices

A. Screenshots

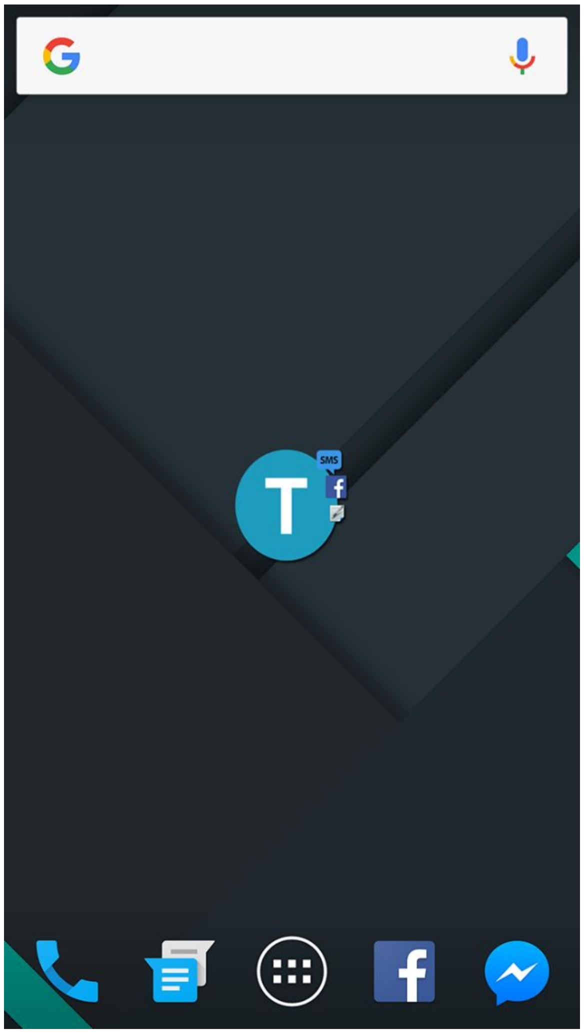


Figure 5.0 Launcher Icon

The launcher icon of the Textor application, this is what users will see on their home screen or app drawer before launching the application.

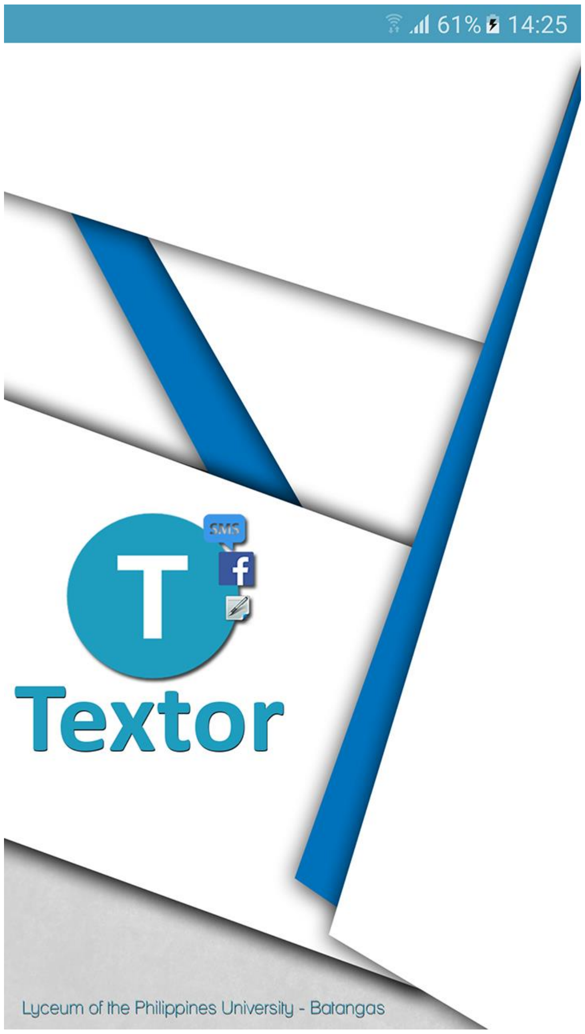


Figure 6.0 Splash Screen

The splash screen of the application, users will be presented with this image before proceeding to the main screen.

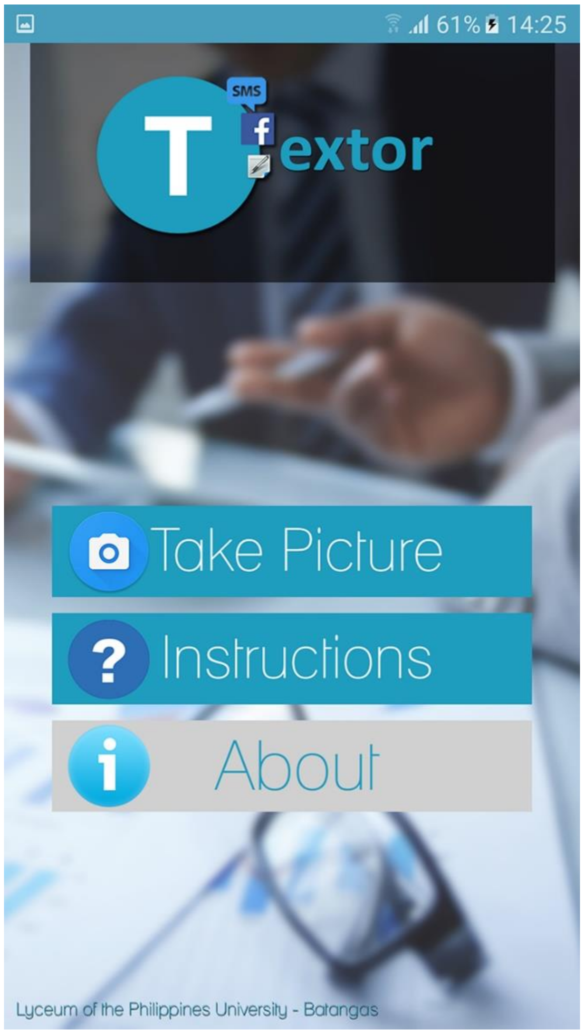


Figure 7.0 Main Screen

The main screen of the application, this screen will allow users to choose whether to take a picture, view the instructions or learn more about the application and its authors.

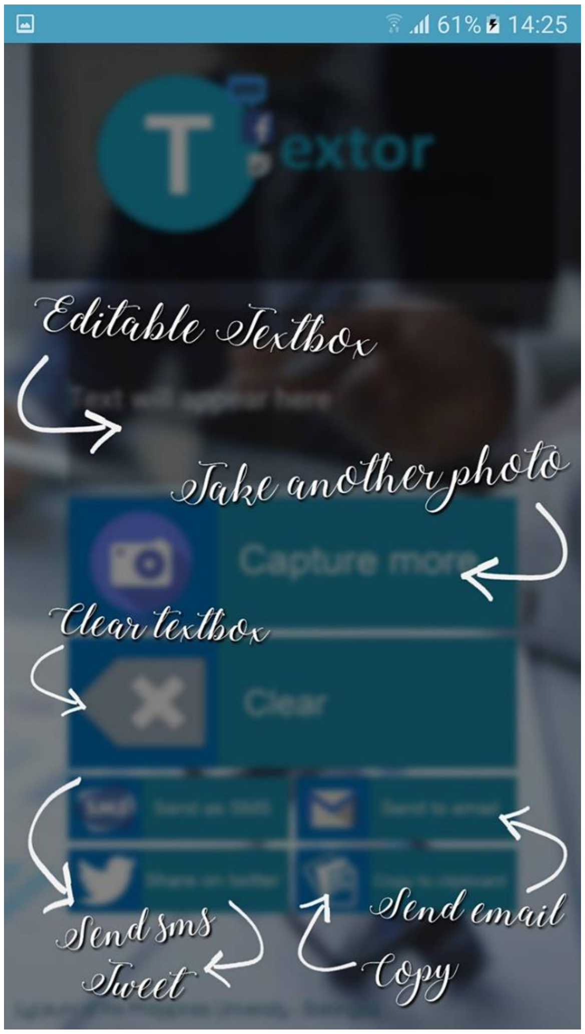
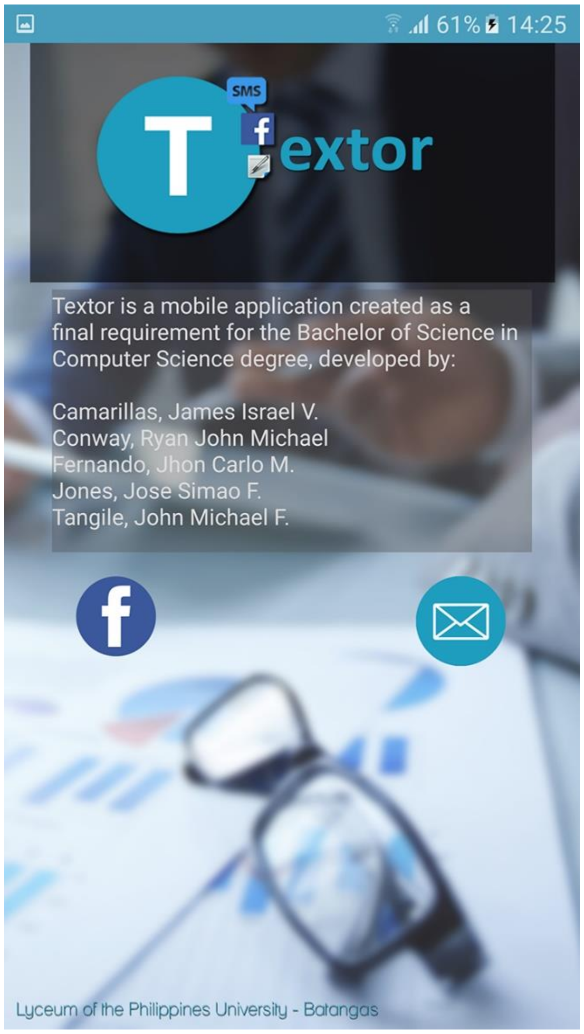


Figure 8.0 Instructions Screen

This screen shows the different functions of each button on the extracted text screen.



Textor is a mobile application created as a final requirement for the Bachelor of Science in Computer Science degree, developed by:

Camarillas, James Israel V.
Conway, Ryan John Michael
Fernando, Jhon Carlo M.
Jones, Jose Simao F.
Tangile, John Michael F.

Figure 9.0 About Screen

This screen displays information about the application and its authors as well as a link to contact them through Facebook and E-mail.

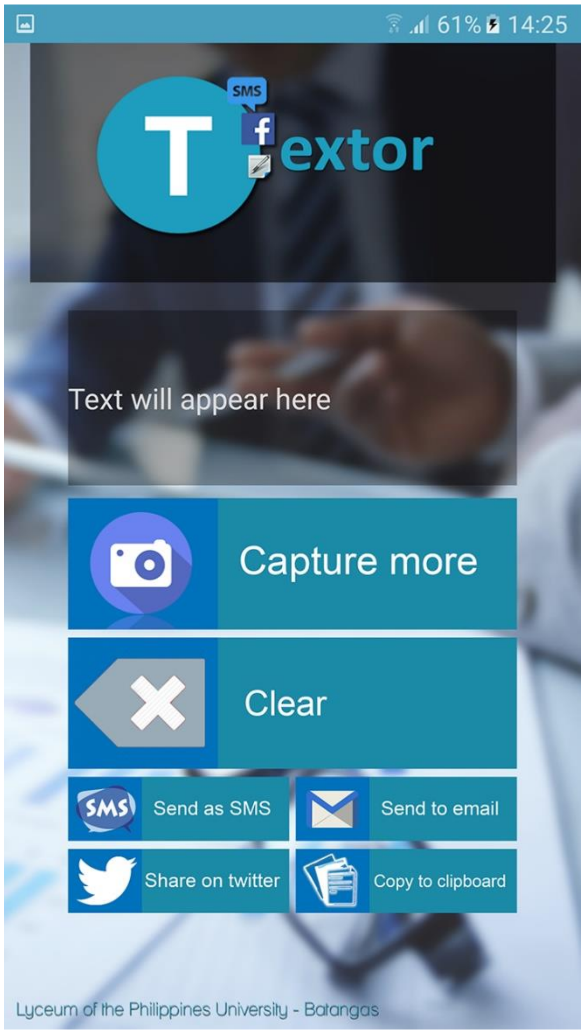


Figure 10.0 Extracted Text Screen

This is the extracted text screen where the detected text is displayed to the user. The user may then capture more text, clear the text field, or share the text via SMS, E-mail, and Twitter or copy it to the clipboard.

C. Compatible Devices

Phone Model	Compatibility
Asus Zenfone 2	✓
Asus Zenfone 3 Max	✓
Asus Zenfone 5	✓
Huawei Honor 4X	✓
Huawei P9 Lite	✓
LG G3	✓
LG G4	✓
Samsung Galaxy Grand	✓
Samsung Galaxy J2	✓
Samsung Galaxy S3	✓
Samsung Galaxy S4	✓
Samsung Galaxy S5	✓
Samsung Galaxy S7	✓
Samsung Galaxy S7 Edge	✓
Sony Xperia C3	✓
Sony Xperia Z2	✓
Sony Xperia M	✓

Table 3.0
List of Compatible Devices

D. Sample Test Results

Font	R1p1		R1p2		R1p3		R1p4		R1p5		R2p1		R2p2		R2p3		R2p4		R2p5		R3p1		R3p2		R3p3		R3p4		R3p5		R1 Average	R2 Average	R3 Average	Average
Arial	43	100.00	43	100.00	43	100.00	43	100.00	43	100.00	43	100.00	43	100.00	35	81.40	43	100.00	41	95.35	7	70.00	10	100.00	10	100.00	7	70.00	7	70.00	100.00	95.35	82.00	92.45
Bell MT	32	74.42	43	100.00	38	88.37	43	100.00	43	100.00	42	97.67	43	100.00	0	0.00	43	100.00	32	74.42	9	90.00	6	60.00	8	80.00	6	60.00	7	70.00	92.56	74.42	72.00	79.66
Bookman Old St	39	90.70	43	100.00	43	100.00	40	93.02	43	100.00	42	97.67	37	86.05	43	100.00	43	100.00	43	100.00	7	70.00	7	70.00	8	80.00	9	90.00	0	0.00	96.74	96.74	62.00	85.16
Bradley hand	0	0.00	17	39.53	28	65.12	34	79.07	23	53.49	31	72.09	23	53.49	34	79.07	20	46.51	30	69.77	0	0.00	5	50.00	3	30.00	0	0.00	6	60.00	47.44	64.19	28.00	46.54
Calibri	31	72.09	40	93.02	25	58.14	33	76.74	42	97.67	33	76.74	42	97.67	42	97.67	34	79.07	23	53.49	0	0.00	9	90.00	0	0.00	7	70.00	0	0.00	79.53	80.93	32.00	64.16
Courier new	40	93.02	31	72.09	35	81.40	41	95.35	40	93.02	35	81.40	39	90.70	35	81.40	33	76.74	31	72.09	10	100.00	7	70.00	9	90.00	7	70.00	8	80.00	86.98	80.47	82.00	83.15
Chiller	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	4	9.30	28	65.12	23	53.49	23	53.49	28	65.12	2	20.00	4	40.00	2	20.00	0	0.00	0	0.00	0.00	49.30	16.00	21.77
Eras medium	43	100.00	43	100.00	43	100.00	43	100.00	43	100.00	19	44.19	39	90.70	43	100.00	42	97.67	41	95.35	9	90.00	9	90.00	8	80.00	9	90.00	8	80.00	100.00	85.58	86.00	90.53
Forte	22	51.16	41	95.35	25	58.14	18	41.86	0	0.00	23	53.49	1	2.33	2	4.65	0	0.00	18	41.86	0	0.00	0	0.00	8	80.00	7	70.00	0	0.00	49.30	20.47	30.00	33.26
Franklin gothic	41	95.35	42	97.67	29	67.44	41	95.35	41	95.35	38	88.37	0	0.00	40	93.02	43	100.00	43	100.00	9	90.00	5	50.00	6	60.00	6	60.00	2	20.00	90.23	76.28	56.00	74.17
Gadugi	38	88.37	42	97.67	41	95.35	41	95.35	43	100.00	43	100.00	43	100.00	43	100.00	28	65.12	43	100.00	0	0.00	8	80.00	7	70.00	7	70.00	7	70.00	95.35	93.02	58.00	82.12
Harrington	39	90.70	0	0.00	30	69.77	0	0.00	38	88.37	0	0.00	17	39.53	32	74.42	29	67.44	30	69.77	0	0.00	6	60.00	8	80.00	8	80.00	8	80.00	49.77	50.23	60.00	53.33
Impact	25	58.14	0	0.00	0	0.00	2	4.65	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	1	10.00	5	50.00	0	0.00	5	50.00	0	0.00	12.56	0.00	22.00	11.52
Jokerman	27	62.79	28	65.12	34	79.07	31	72.09	33	76.74	37	86.05	36	83.72	36	83.72	16	37.21	36	83.72	6	60.00	5	50.00	4	40.00	6	60.00	7	70.00	71.16	74.88	56.00	67.35
Kristen ITC	38	88.37	40	93.02	40	93.02	38	88.37	38	88.37	42	97.67	28	65.12	23	53.49	41	95.35	30	69.77	7	70.00	5	50.00	3	30.00	8	80.00	6	60.00	90.23	76.28	58.00	74.84
Lucida calligrap	25	58.14	40	93.02	31	72.09	20	46.51	21	48.84	1	2.33	1	2.33	4	9.30	31	72.09	11	25.58	7	70.00	9	90.00	9	90.00	7	70.00	7	70.00	63.72	22.33	78.00	54.68
Lucida console	43	100.00	42	97.67	42	97.67	43	100.00	41	95.35	35	81.40	43	100.00	33	76.74	36	83.72	43	100.00	1	10.00	4	40.00	0	0.00	1	10.00	8	80.00	98.14	88.37	28.00	71.50
Monotype corsi	0	0.00	0	0.00	0	0.00	0	0.00	28	65.12	21	48.84	22	51.16	1	2.33	1	2.33	17	39.53	10	100.00	8	80.00	2	20.00	8	80.00	7	70.00	13.02	28.84	70.00	37.29
MV Boli	42	97.67	42	97.67	41	95.35	41	95.35	39	90.70	6	13.95	24	55.81	25	58.14	33	76.74	34	79.07	9	90.00	9	90.00	8	80.00	7	70.00	5	50.00	95.35	56.74	76.00	76.03
OCR A extended	26	60.47	40	93.02	41	95.35	35	81.40	39	90.70	23	53.49	2	4.65	41	95.35	12	27.91	10	23.26	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	84.19	40.93	0.00	41.71
Papyrus	24	55.81	23	53.49	29	67.44	0	0.00	23	53.49	3	6.98	7	16.28	6	13.95	22	51.16	4	9.30	6	60.00	5	50.00	5	50.00	5	50.00	6	60.00	46.05	19.53	54.00	39.86
Sogoe print	36	83.72	43	100.00	39	90.70	40	93.02	40	93.02	42	97.67	43	100.00	36	83.72	43	100.00	43	100.00	8	80.00	6	60.00	8	80.00	3	30.00	4	40.00	92.09	96.28	58.00	82.12
Tahoma	43	100.00	41	95.35	43	100.00	40	93.02	42	97.67	43	100.00	43	100.00	31	72.09	43	100.00	43	100.00	10	100.00	6	60.00	10	100.00	10	100.00	10	100.00	97.21	94.42	92.00	94.54
Times new rom	43	100.00	43	100.00	43	100.00	43	100.00	40	93.02	42	97.67	43	100.00	42	97.67	43	100.00	43	100.00	7	70.00	8	80.00	7	70.00	8	80.00	8	80.00	98.60	99.07	76.00	91.22
Verdana	43	100.00	43	100.00	43	100.00	42	97.67	43	100.00	43	100.00	43	100.00	43	100.00	43	100.00	43	100.00	8	80.00	5	50.00	10	100.00	7	70.00	10	100.00	99.53	100.00	80.00	93.18
Viner hand ITC	7	16.28	20	46.51	20	46.51	20	46.51	11	25.58	0	0.00	19	44.19	17	39.53	0	0.00	10	23.26	2	20.00	0	0.00	0	0.00	0	0.00	0	0.00	36.28	21.40	4.00	20.56

Legend:	
	🟧 serif
	🟦 sans serif
	🟨 script
	🟩 decorative

Table 4.0
Samsung Galaxy S7 Test Results

PERSONAL INFORMATION

Name:	James Israel V. Camarillas
Age:	20
Gender:	Male
Date of Birth:	November 22, 1996
Address:	Sto. Cristo San Jose, Batangas
Contact Number:	09772479615
Email Address:	james.camarillas@gmail.com
Nationality:	Filipino

FAMILY BACKGROUND

Father's Name:	Roel D. Camarillas	**Contact Number:**	09086334325
Occupation:	Pastor		
Mother's Name:	Adelita V. Camarillas	**Contact Number:**	09498560085
Occupation:	Housewife		

EDUCATIONAL BACKGROUND

Primary:	Christ Outreach Christian Academy
Secondary:	Sovereign Shepherd School of Values and Learning
Tertiary:	Lyceum of the Philippines University – Batangas

PERSONAL INFORMATION

Name:	Conway, Ryan John Michael
Age:	20
Gender:	Male
Date of Birth:	October 4, 1996
Address:	Gulod Labac, Batangas City
Contact Number:	09063970410
Email Address:	rjmconway@hotmail.co.uk
Nationality:	British

FAMILY BACKGROUND

Father's Name: Peter Conway **Contact Number:** N/A

Occupation: Maintenance Superintendent

Mother's Name: Melanie Conway **Contact Number:** 09355440705

Occupation: Businesswoman

EDUCATIONAL BACKGROUND

Primary:	Victoria English School
Secondary:	Stonyhurst Southville International School
Tertiary:	Lyceum of the Philippines University - Batangas

PERSONAL INFORMATION

Name:	Jhon Carlo M. Fernando
Age:	20
Gender:	Male
Date of Birth:	July 06, 1996
Address:	96 P. Hererra Street, Barangay 06, Batangas City
Contact Number:	09178795480
Email Address:	jhoncarlofernando.ccs@gmail.com
Nationality:	Filipino

FAMILY BACKGROUND

Father's Name:	Willy B. Stellander	**Contact Number:**	N/A
Occupation:	Deceased		
Mother's Name:	Jocelyn M. Fernando	**Contact Number:**	09175421399
Occupation:	Housewife		

EDUCATIONAL BACKGROUND

Primary:	Puerto Galera Central School
Secondary:	Puerto Galera National High School
Tertiary:	Lyceum of the Philippines University - Batangas

PERSONAL INFORMATION

Name:	Jose Simao F. Jones
Age:	24
Gender:	Male
Date of Birth:	September 17, 1992
Address:	Saint Peter Subd., Batangas
Contact Number:	09279092184
Email Address:	jonesjose.ccs@gmail.com
Nationality:	Angolan

FAMILY BACKGROUND

Father's Name: Lucas Jones **Contact Number:** +244923505997

Occupation: Military

Mother's Name: Gabriela Rosa Caprego **Contact Number:** +244923605874

Occupation: Businesswoman

EDUCATIONAL BACKGROUND

Primary:	Angolan Elementary School
Secondary:	Central Puniv Angolan
Tertiary:	Lyceum of the Philippines University – Batangas

PERSONAL INFORMATION

Name:	John Michael F. Tangile
Age:	22
Gender:	Male
Date of Birth:	June 02, 1994
Address:	Maraykit San Juan, Batangas
Contact Number:	09162032576
Email Address:	michaeltangile.ccs@gmail.com
Nationality:	Filipino

FAMILY BACKGROUND

Father's Name:	Jessie A. Tangile	**Contact Number:**	09258020359
Occupation:	Chief Mechanic		
Mother's Name:	Marina F. Tangile	**Contact Number:**	09053342655
Occupation:	Businesswoman		

EDUCATIONAL BACKGROUND

Primary:	San Juan West Elementary School
Secondary:	Joseph Marello Institute
Tertiary:	Lyceum of the Philippines University – Batangas